Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 21, 2016

P. Saint-Andre
&yet
S. Ibarra
AG Projects
E. Ivov
Jitsi

July 20, 2015

Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Media Sessions
draft-ietf-stox-media-07

Abstract

This document defines a bidirectional protocol mapping for use by gateways that enable the exchange of media signaling messages between systems that implement the Session Initiation Protocol (SIP) and systems that implement the Jingle extensions to the Extensible Messaging and Presence Protocol (XMPP).

Table of Contents

1. Introduction

The Session Initiation Protocol [RFC3261] is a widely-deployed technology for the management of media sessions (such as voice and video calls) over the Internet. SIP itself provides a signaling channel via TCP [RFC0793] or UDP [RFC0768], over which two or more parties can exchange messages for the purpose of negotiating a media session that uses a dedicated media channel such as the Real-time Transport Protocol (RTP) [RFC3550]. The Extensible Messaging and Presence Protocol (XMPP) [RFC6120] also provides a signaling channel, typically via TCP (although bindings for HTTP [XEP-0124] and WebSocket [RFC7395] also exist).

Given the significant differences between XMPP and SIP, traditionally it was difficult to combine the two technologies in a single user agent (although nowadays such implementations are not uncommon, as

described in [RFC7081]). Thus in 2005 some developers wishing to add media session capabilities to XMPP clients defined a set of XMPP-specific session negotiation protocol extensions called Jingle (see especially [XEP-0166], [XEP-0167], and [XEP-0176]).

Jingle was designed to easily map to SIP for communication through gateways or other transformation mechanisms. Nevertheless, given the significantly different technology assumptions underlying XMPP and SIP, Jingle is different from SIP in several important respects:

o Base SIP messages and headers use a plaintext format similar in some ways to the Hypertext Transport Protocol [RFC7230], whereas Jingle messages are pure XML. Mappings between SIP headers and Jingle message syntax are provided below.

o SIP payloads for session semantics use the Session Description Protocol [RFC4566], whereas the equivalent Jingle payloads use XML child elements of the Jingle <content/> element. However, the Jingle specifications defining such child elements specify mappings to SDP for all Jingle syntax, making the mapping relatively straightforward.

o SIP messages have historically often been transported over UDP, whereas the signaling channel for Jingle is XMPP over TCP. Mapping between the transport layers typically happens within a gateway using techniques below the application level, and therefore is not addressed in this specification.

Consistent with existing specifications for mapping between SIP and XMPP (see [RFC7247]), this document describes a bidirectional protocol mapping for use by gateways that enable the exchange of media signaling messages between systems that implement SIP and systems that implement the XMPP Jingle extensions.

It is important to note that SIP and Jingle sessions could be gatewayed in a very simple way if all media were always routed and potentially even transcoded through the same gateway used for signaling. By contrast, this specification defines a mapping that allows gateways to only intervene at the signaling level, thus letting user agents exchange media in an end-to-end or peer-to-peer manner without intervention by a specialized gateway (naturally, a media relay that supports TURN [RFC5766] might be used). Such signaling-only gateways focus on handling session establishment and control within the context of what users would perceive as "calls". This document is hence primarily dealing with calling scenarios as opposed to generic media sessions or specialized sessions for functionality such as file transfer (see [RFC5547] and [XEP-0234]).

2. Intended Audience

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

- RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551]

- The Secure Real-time Transport Protocol (SRTP) [RFC3711]

- SDP: Session Description Protocol [RFC4566]

- Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols [RFC5245]

- Jingle [XEP-0166]

- Jingle RTP Sessions [XEP-0167]

- Jingle ICE-UDP Transport Method [XEP-0176]

- Jingle Raw UDP Transport Method [XEP-0177]

3. Terminology

A number of technical terms used here are defined in [RFC3261], [RFC6120], [XEP-0166], and [XEP-0167]. The term "JID" is short for "Jabber Identifier".

In flow diagrams, SIP traffic is shown using arrows such as "***>" whereas XMPP traffic is shown using arrows such as "...>".

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

4. Compatibility with Offer/Answer Model and Interactive Connectivity Establishment

Even if Jingle semantics have many similarities with those used in SIP, there are some use cases that cannot be handled in exactly the same way due to the Offer/Answer model used in SIP in conjunction with SDP.

More specifically, mapping SIP and SDP Offer/Answer to XMPP is often complicated due to the difference in how each handles backward compatibility. Jingle, as most other XMPP extensions, relies heavily on the XMPP extension for service discovery [XEP-0030], which implies that XMPP entities are able to verify the capabilities of their intended peer before attempting to establish a session with it.

SDP Offer/Answer, on the other hand, uses a "least common denominator" approach where every SDP offer needs to be comprehensible by legacy endpoints. Newer, unsupported aspects in this offer can therefore only appear as optional, or their use needs to be limited to subsequent Offer/Answer exchanges once their support has been confirmed.

In particular, many older SIP endpoints do not support Interactive Connectivity Establishmen (ICE) [RFC5245]. A signaling gateway from Jingle to SIP has two primary alternatives for dealing with such endpoints on the SIP side:

o Require the use of ICE and otherwise fail the call by including the "Require: ice" SIP option tag [RFC5768] in the SIP INVITE that it sends on behalf of the Jingle initiator.

o Send an initial SIP INVITE for an ICE connection and, if the SIP endpoint indicates that it cannot handle ICE, send a re-INVITE for a non-ICE connection to the SIP endpoint and a Jingle transport-replace for a Raw UDP connection to the Jingle endpoint. (This will introduce a potentialy large delay and might not result in a much higher percentage of calls succeeding unless the signaling gateway also offers a TURN [RFC5766] service for NAT traversal.) See Section 6 for further discussion.

Use of "Trickle ICE" is one significant example where this issue occurs. From the beginning, Jingle supported the trickling of candidates (via Jingle messages of type 'transport-info'), and only years later was this behavior generalized [I-D.ietf-mmusic-trickle-ice] and then ported to SIP [I-D.ietf-mmusic-trickle-ice-sip]. Therefore SIP endpoints need to always behave like so-called "vanilla ICE" agents when sending their first offer and make sure they gather all candidates before sending a

SIP INVITE. This is necessary because otherwise ICE agents with no support for trickling of candidates can prematurely declare failure. Jingle endpoints, on the other hand, can verify support for trickling of candidates prior to engaging in a session and adapt their behavior accordingly (and, as noted, trickling of candidates is standard operating procedure in Jingle).

In order to work around this disparity in relation to communication of transport candidates, the Jingle RTP transport method [XEP-0176] defines a mode for supporting traditional Offer/Answer interactions through the "urn:ietf:rfc:3264" feature tag. When an XMPP entity such as a client (or, significantly, a gateway to a SIP system) advertises support for this feature, the entity indicates that it needs to receive multiple transport candidates in the initial offer, instead of receiving them trickled over time. Although implementations conforming to this specification MUST support the Offer/Answer model with Jingle, such endpoints SHOULD NOT actually declare support for the "urn:ietf:rfc:3264" service discovery feature since this would mean that they too would be reachable only through Offer/Answer semantics and not also through trickle-ICE semantics.

The difference in handling of transport candidates also has an impact on ICE restarts (see Section 9.1.1.1 of [RFC5245]). Because Jingle endpoints can send candidates at any time, when communicating directly with other Jingle endpoints they would not initiate an ICE restart simply in order to send a candidate that, for example, changes the media target. However, as part of support for the Offer/Answer model a Jingle endpoint would instead need to initiate an ICE restart when communicating with a SIP endpoint or gateway that does not support trickle ICE. Similarly, a Jingle endpoint needs to support the 'generation' attribute (used to signal an ICE restart) when communicating with a SIP endpoint or gateway that does not support trickle ICE. See also the syntax discussion under Section 5.4.

5. Syntax Mappings

5.1. Generic Jingle Syntax

Jingle is designed in a modular fashion, so that session description data is generally carried in a payload within high-level Jingle elements, i.e., the <jingle/> element and its <content/> child. The following example illustrates this structure, where the XMPP stanza is a request to initiate an audio session (via the <content/> and <description/> elements) using a transport of RTP over raw UDP (via the <transport/> element).

Example 1: Structure of a Jingle session initiation request

```
| <iq from='romeo@example.net/v3rsch1kk3l1jk'
|     id='ne91v36s'
|     to='juliet@example.com/t3hr0zny'
|     type='set'>
|   <jingle xmlns='urn:xmpp:jingle:1'
|           action='session-initiate'
|           initiator='romeo@example.net/v3rsch1kk3l1jk'
|           sid='a73sjjvkla37jfea'>
|     <content creator='initiator'
|              media='audio'
|              name='this-is-the-audio-content'
|              senders='both'>
|       <description xmlns='urn:xmpp:jingle:app:rtp:1'>
|         <payload-type id='101' name='opus' clockrate='48000'/>
|         <payload-type id='18' name='G729'/>
|         <payload-type channels='2'
|                       clockrate='16000'
|                       id='103'
|                       name='L16'/>
|         <payload-type id='98' name='x-ISAC' clockrate='8000'/>
|       </description>
|       <transport xmlns='urn:xmpp:jingle:transport:raw-udp'>
|         <candidate id='v3c18fgg'
|                    ip='10.1.1.104'
|                    port='13540'
|                    generation='0'/>
|       </transport>
|     </content>
|   </jingle>
| </iq>
```

The syntax and semantics of the <jingle/> and <content/> elements are defined in the core Jingle specification [XEP-0166], the syntax and semantics of the <description/> element qualified by the 'urn:xmpp:jingle:app:rtp:1' namespace are defined in the Jingle RTP specification [XEP-0167], and the syntax and semantics of the <transport/> element qualified by the 'urn:xmpp:jingle:transport:raw-udp' namespace are defined in the Jingle Raw UDP specification [XEP-0177]. Other <description/> elements are defined in specifications for the appropriate application types (see for example [XEP-0234] for file transfer) and other <transport/> elements are defined in the specifications for appropriate transport methods (see for example [XEP-0176], which defines an XMPP profile of ICE [RFC5245]).

At the core Jingle layer, the following mappings are defined.

Table 1: High-Level Mapping from XMPP to SIP

Jingle	SIP
<jingle/> 'action'	[see next table]
<jingle/> 'initiator'	[no mapping]
<jingle/> 'responder'	[no mapping]
<jingle/> 'sid'	local-part of Dialog ID
local-part of 'initiator'	<username> in SDP o= line
<content/> 'creator'	[no mapping]
<content/> 'name'	no mandatory mapping (1)
<content/> 'senders' value of both, initiator, responder, or none	a= line of sendrecv, recvonly, sendonly, or inactive

1. In can be appropriate to map to the a=mid value defined in [RFC5888].

The 'senders' attribute is optional in Jingle, with a default value of "both"; thus in case the attribute is absent the SDP direction value MUST be considered as 'sendrecv'.

The 'action' attribute of the <jingle/> element has 15 allowable values. In general they should be mapped as shown in the following table, with some exceptions as described below.

Table 2: Mapping of Jingle Actions to SIP Methods

Jingle Action	SIP Method
content-accept	INVITE response (1xx or 2xx)
content-add	INVITE request
content-modify	re-INVITE request
content-reject	unused in this mapping
content-remove	INVITE request
description-info	unused in this mapping
security-info	unused in this mapping
session-accept	INVITE response (1xx or 2xx)
session-info	see note (1) below
session-initiate	INVITE request
session-terminate	BYE
transport-accept	unused in this mapping
transport-info	see note (2) below
transport-reject	unused in this mapping
transport-replace	unused in this mapping

1. The Jingle session-info action can be used for multiple purposes, such as putting the session on hold or sending a ringing indication. In particular, a session-info action of type 'ringing' SHOULD be mapped to a 180 SIP provisional response. The use of session-info for the purpose of session hold is described in Section 7.

2. In Jingle the transport-info action is used to exchange transport candidates after the initial offer, as documented in [XEP-0176]. This usage has been generalized as "Trickle ICE" [I-D.ietf-mmusic-trickle-ice] and has also been extended to SIP [I-D.ietf-mmusic-trickle-ice-sip]. Therefore a Jingle action of

transport-info SHOULD be mapped to a SIP INFO request, but only in cases where it is reasonable to assume that the SIP endpoint or gateway supports trickle ICE. See Section 4 for further discussion.

5.2. Application Formats

Jingle application formats for audio and video exchange via RTP are specified in [XEP-0167]. These application formats effectively map to the "RTP/AVP" profile specified in [RFC3551] and the "RTP/SAVP" profile specified in [RFC3711], where the media types are "audio" and "video" and the specific mappings to SDP syntax are provided in [XEP-0167].

(As stated in [XEP-0167], future versions of that specification might define how to use other RTP profiles such as "RTP/AVPF" and "RTP/SAVPF" as defined in [RFC4585] and [RFC5124] respectively.)

5.3. Raw UDP Transport Method

A basic Jingle transport method for exchanging media over UDP is specified in [XEP-0177]. This "Raw UDP" transport method involves the negotiation of an IP address and port only. It does not provide NAT traversal, effectively leaving the task to intermediary entities (which might be a media relay associated with but functionally independent of a signaling gateway). The Jingle 'ip' attribute maps to the connection-address parameter of the SDP c= line and the 'port' attribute maps to the port parameter of the SDP m= line. Use of SIP without ICE would generally map to use of Raw UDP on the XMPP side of a session.

5.4. ICE-UDP Transport Method

A more advanced Jingle transport method for exchanging media over UDP uses Interactive Connectivity Establishment and is specified in [XEP-0176]. By following the ICE methodology specified in [RFC5245], ideally this transport method provides NAT traversal for media.

The relevant SDP mappings are provided in [XEP-0176]. However, those who implement signaling gateways need to be aware of a few syntax incompatibilities that need to be addressed by gateways conforming to this specification:

- o The 'foundation' attribute is defined as a number in Jingle (unsigned byte) whereas ICE [RFC5245] defines it as a string, which can contain letters, digits and the '+' and '/' symbols. Gateway applications MUST therefore convert ICE originating foundations into integer numbers and they MUST guarantee that such

a conversion preserves foundation uniqueness. The exact mechanism for the conversion is undefined.

- Jingle defines a 'generation' attribute which is used to determine if an ICE restart is required. This attribute has no counterpart in SIP because ICE restarts are initiated by detecting a change in the ICE 'ufrag' and 'pwd' (see Section 9.1.1.1 of [RFC5245]). Gateways MUST therefore increase the generation number when they detect such a change.

- The 'id' attribute defined by Jingle has no SIP counterpart; thus applications are free to choose means to generate unique identifiers across the different candidates of an ICE generation.

- The 'network' attribute defined by Jingle has no counterpart in SIP and SHOULD be ignored.

6. Transport Fallback

Most Jingle endpoints will first attempt to use ICE as specified for Jingle in [XEP-0176] (since that is most likely to result in NAT traversal) and only if that does not succeed will they fall back to raw UDP [XEP-0177]. This fallback approach is described in the Jingle ICE specification [XEP-0176].

However, that approach depends on the use of XMPP service discovery [XEP-0030]. Because SIP does not have a method for determining endpoint capabilities, SIP endpoints use what can be termed "single-exchange fallback": they first try one method and if that fails they then send a re-INVITE with the second method.

One way to map single-exchange fallback to Jingle is for the Jingle endpoint to attempt ICE first and send a transport-replace if the SIP answer indicates no support for ICE, then send a SIP re-INVITE with the addresses in the transport-accept. Unfortunately, this approach will result a fairly substantial post-answer delay before media can flow.

Because such delays usually result in an unacceptable user experience, the trend for many calling applications is to first send only a candidate that is known beforehand to be highly likely to result in NAT traversal, which is almost always a candidate at a media relay (i.e., an ICE candidate of type "relay"). Such applications will then offer and perhaps switch to a host candidate, peer reflexive candidate, or server reflexive candidate only after media is flowing via the relayed candidate. This approach obviates the need for transport fallback from ICE to raw UDP during call setup, and instead works around the problem by using trickle ICE (for

those endpoints that support it) or re-INVITEs with updated transport candidates after call setup has been completed.

7. Call Hold

The Offer/Answer model [RFC3264] stipulates that streams are placed on hold by setting their direction to "sendonly". A session is placed on hold by doing this for all the streams it contains. The same semantics are also supported by Jingle through the "senders" element and its "initiator" and "responder" values (the Jingle specification also defines a value of "none", which maps to an a= value of "inactive", and a default value of "both", which maps to an a= value of "sendrecv").

The following example shows how the responder would put the call on hold (i.e., temporarily stop listening to media sent by the initiator) using a Jingle content-modify action and a modified value for the 'senders' attribute (here a value of "responder" is used to indicate that the responder might continue to send media, such as hold music).

Example 2: Call hold via 'senders' attribute

```
| <iq from='juliet@capulet.lit/balcony'
|     id='hz73n2l9'
|     to='romeo@montague.lit/orchard'
|     type='set'>
|   <jingle xmlns='urn:xmpp:jingle:1'
|           action='content-modify'
|           initiator='romeo@montague.lit/orchard'
|           sid='a73sjjvkla37jfea'>
|     <content creator='initiator'
|              media='audio'
|              name='this-is-the-audio-content'
|              senders='responder'/>
|   </jingle>
| </iq>
```

In addition to these semantics, however, the Jingle RTP Sessions specification [XEP-0167] also defines a more concise way for achieving the same end, which consists in sending a "hold" command within a "session-info" action, as shown in the following example.

Example 3: Call hold via session-info action

```
| <iq from='juliet@capulet.lit/balcony'
|     id='xv39z423'
|     to='romeo@montague.lit/orchard'
|     type='set'>
|   <jingle xmlns='urn:xmpp:jingle:1'
|           action='session-info'
|           initiator='romeo@montague.lit/orchard'
|           sid='a73sjjvkla37jfea'>
|     <hold xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
|   </jingle>
| </iq>
```

Gateways that receive either of the foregoing hold notifications from their Jingle side MUST generate a new offer on their SIP side, placing all streams in a "sendonly" state.

When relaying offers from SIP to XMPP, gateways are not required to translate "sendonly" attributes into a "hold" command as this would not always be possible (e.g., when not all streams have the same direction). Additionally, such conversions might introduce complications in case further offers placing a session on hold also contain other session modifications.

It is possible that, after one entity has put the other on hold, the second entity might put the first entity on hold. In this case, the effective direction would then be "inactive" in SDP and "none" in Jingle.

8. Early Media

[RFC3959] and [RFC3960] describe a number of scenarios relying on "early media". While similar attempts have also been made for XMPP, support for early media is not currently widely supported in Jingle implementations. Therefore, gateways SHOULD NOT forward SDP answers from SIP to Jingle until a final response has been received, except in cases where the gateway is in a position to confirm specific support for early media by the endpoint (one approach to such support can be found in [XEP-0269] but it has not yet been standardized).

Gateways MUST however store early media SDP answers when they are sent inside a reliable provisional response. In such cases, a subsequent final response can follow without an actual answer and the one from the provisional response will need to be forwarded to the Jingle endpoint.

9. Detecting Endless Loops

[RFC3261] defines a "Max-Forwards" header that allows intermediate entities such as SIP proxies to detect and prevent loops from occurring. The specifics of XMPP make such a prevention mechanism unnecessary for XMPP-only environments. With the introduction of SIP-to-XMPP gatewaying, however, it would be possible for loops to occur where messages are being repeatedly forwarded from XMPP to SIP to XMPP to SIP. This can happen not only between two endpoints, but also with the addition of a third endpoint into the mix (e.g., because one of the two original endpoints forwards a call to a third endpoint, thus converting a "spiral" into a loop).

To compensate for the lack of a "Max-Forwards" header in SIP, gateways MUST therefore keep track of all SIP transactions and Jingle sessions that they are currently serving and they MUST block re-entrant messages. Although the specifics of such tracking are a matter of implementation, the broad requirements is to consistently translate SIP dialog IDs into Jingle session ID, and vice versa, or generate an internal identifier for each session (e.g., by concatenating or hashing the combination of the SIP dialog ID and the Jingle session ID).

10. SDP Format-Specific Parameters

The SDP specification [RFC4566] defines "a=fmtp" attributes for the transmission of format-specific parameters as a single transparent string. Such strings can be used to convey either a single value or a sequence of parameters, separated by semi-colons, commas, or whatever delimiters are chosen by a particular payload type specification.

The Jingle RTP application format [XEP-0167], on the other hand, defines a <parameter/> element as follows:

```
<parameter name="paramName" value="paramValue"/>
```

A sequence of parameters is thus transmitted as an array of distinct name/value pairs, at least in the context of the Jingle RTP extension.

These differences make it difficult to devise a generic mechanism that accurately translates format parameters from Jingle RTP to SDP without the specifics of the payload being known to the gateway. In general this is not a major problem because most of the media type definitions supported in existing Jingle implementations follow the semicolon-separated parameter model (e.g., typical audio and video codecs). Possible exceptions include:

o The RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals (i.e., the "audio/telephone-event" payload type). As noted in Section 2.5.1.1 of [RFC4733], in SDP the "events" parameter is assumed to indicate support for DTMF events 0-15 even if the parameter is not included; a gateway SHOULD explicitly indicate this support in a Jingle parameter with name='events' and value='0-15'.

o The RTP Payload for Redundant Audio Data (i.e., the "audio/red" payload type). Although this payload type is defined in [RFC2198], the SDP representation is specified in Section 4.1.21 of [RFC3555] (note that this representation was not updated by [RFC4856]). In particular, the "pt" parameter can be mapped to a=fmtp lines as described in the payload type registration.

For implementations that wish to provide a general-purpose translation method, this specification makes the following recommendations:

1. Gateways that are aware of the formats in use SHOULD parse all format parameters and generate <parameter/> arrays and "a=fmtp" values accordingly.

2. When translating Jingle RTP to SIP, gateways that have no explicit support for the formats that are being negotiated SHOULD convert the list of <parameter/> elements into a single string, containing a sequence of "name=value" pairs, separated by a semi-colon and a space (i.e. "; ").

3. When translating SIP to Jingle RTP, gateways that have no explicit support for the formats that are being negotiated SHOULD tokenize the "a=fmtp" format string using one delimiter from the following list: ";", "; ", ",", ", ". The resulting tokens SHOULD then be parsed as "name=value" pairs. If this process does actually yield any such pairs, they SHOULD be used for generating the respective <parameter/> elements. If some of the tokens cannot be parsed into a "name=value" pair because they do not conform to the convention suggested in [RFC4855], or in case the format string could not be tokenized with the above delimiters, the remaining strings SHOULD be used as a value for the "value" attribute of the <parameter/> element and the corresponding "name" attribute SHOULD be left empty.

Here is a relatively simple example of the foregoing transformations, using the aforementioned example of the "audio/telephone-event" payload type (wherein the "events" parameter is implicitly named in the SDP):

SDP with format data (audio/telephone-event)

```
a=rtpmap:100 telephone-event/8000
a=fmtp:100 0-15,66,70
```

Jingle transformation (audio/telephone-event)

```
<parameter name="events" value="0-15,66,70"/>
```

A more complicated example would be handling of the "audio/red" payload type (wherein the "pt" parameter can be mapped to a=fmtp lines as described in [RFC3555]):

SDP with format data (audio/red)

```
m=audio 49170 RTP/AVP 99 0 103
a=rtpmap:99 RED/8000
a=fmtp:99 0/103
a=rtpmap:103 G729D/8000
a=fmtp:103 annexb=yes
```

Jingle transformation (audio/red)

```
<parameter name="pt" value="0,103"/>
<parameter name="annexb" value="yes"/>
```

11. Dialog Forking

The core SIP specification [RFC3261] defines semantics for dialog forking. Such semantics have not been defined for Jingle and need to be hidden from XMPP endpoints.

To achieve this, a SIP-to-XMPP gateway MUST NOT forward more than one provisional response on the Jingle side. Typically they would do so only for the first provisional response they receive and ignore the rest. This provisional response SHOULD be forwarded as if it originated from a "user@host" address (i.e., a "bare JID") corresponding to the AOR URI found in the "From" header of the SIP provisional response. The gateway MUST NOT attempt to translate GRUUs into full JIDs because it cannot know at this stage which of the dialogs established by these provisional responses will be used for the actual session.

Likewise, a gateway conforming to this specification MUST NOT forward more than a single final response received through SIP to the Jingle side. The gateway SHOULD terminate the SIP sessions whose received final response was not forwarded to the Jingle side.

12. Sample Call Flow

The section illustrates the protocol flow of a basic voice chat session in which an XMPP user (juliet@example.com) is the initiator and a SIP user (romeo@example.net) is the responder. The signaling is communicated through a gateway. To simplify the example, the Jingle transport method negotiated is "raw UDP" as specified in [XEP-0177].

```
XMPP        XMPP          SIP         SIP
User       Server        Server       User
 |        + X2S GW          |            |
 |            |             |            |
 | (F1) XMPP  |             |            |
 | session-   |             |            |
 | initiate   |             |            |
 |...........>|             |            |
 | (F2) XMPP  |             |            |
 | IQ-result  |             |            |
 |<...........|             |            |
 |            | (F3) SIP    |            |
 |            | INVITE      |            |
 |            |***********>|            |
 |            |             | (F4) SIP   |
 |            |             | INVITE     |
 |            |             |**********>|
 |            |             | (F5) SIP   |
 |            |             | 180        |
 |            |             | ringing    |
 |            |             |<**********|
 |            | (F6) SIP    |            |
 |            | 180 ringing|            |
 |            |<***********|            |
 | (F7) XMPP  |             |            |
 | session-   |             |            |
 | info       |             |            |
 | (ringing)  |             |            |
 |<...........|             |            |
 | (F8) XMPP  |             |            |
 | IQ-result  |             |            |
 |...........>|             |            |
 |            |             | (F9) SIP   |
 |            |             | 200 OK     |
 |            |             |<**********|
 |            | (F10) SIP   |            |
 |            | 200 OK      |            |
 |            |<***********|            |
 | (F11) XMPP |             |            |
```

```
   | session-   |            |            |
   | accept     |            |            |
   |<...........|            |            |
   | (F12) XMPP |            |            |
   | IQ-result  |            |            |
   |...........>|            |            |
   |            | (F13) SIP  |            |
   |            | ACK        |            |
   |            |***********>|            |
   |            |            | (F14) SIP  |
   |            |            | ACK        |
   |            |            |**********>|
   |            |            |            |
   |<=======MEDIA SESSION OVER RTP======>|
   |            |            |            |
   |            |            | (F15) SIP  |
   |            |            |  BYE       |
   |            |            |<**********|
   |            | (F16) SIP  |            |
   |            | BYE        |            |
   |            |<***********|            |
   | (F17) XMPP |            |            |
   | session-   |            |            |
   | terminate  |            |            |
   |<...........|            |            |
   | (F18) XMPP |            |            |
   | IQ-result  |            |            |
   |...........>|            |            |
```

The packet flow is as follows.

First the XMPP user sends a Jingle session-initiation request to the SIP user.

Example 4: Jingle session-initiate (F1)

```
|   <iq from='juliet@example.com/t3hr0zny'
|       id='hu2s61f4'
|       from='romeo@example.net/v3rsch1kk3l1jk'
|       type='set'>
|     <jingle xmlns='urn:xmpp:jingle:1'
|             action='session-initiate'
|             initiator='juliet@example.com/t3hr0zny'
|             sid='a73sjjvkla37jfea'>
|       <content creator='initiator'
|                media='audio'
|                name='this-is-the-audio-content'>
|         <description xmlns='urn:xmpp:jingle:app:rtp:1'>
|           <payload-type id='96' name='speex' clockrate='16000'/>
|           <payload-type id='97' name='speex' clockrate='8000'/>
|           <payload-type id='18' name='G729'/>
|         </description>
|         <transport xmlns='urn:xmpp:jingle:transport:raw-udp'>
|           <candidate component='1' generation='0' id='u3gscv289p'
|                      ip='192.0.2.101' port='49172'/>
|         </transport>
|       </content>
|     </jingle>
|   </iq>
```

The gateway returns an XMPP IQ-result to the initiator on behalf of the responder.

Example 5: XMPP IQ-result from gateway (F2)

```
|   <iq from='juliet@example.com/t3hr0zny'
|       id='hu2s61f4'
|       to='romeo@example.net/v3rsch1kk3l1jk'
|       type='result'/>
```

The gateway transforms the Jingle session-initiate action into a SIP INVITE.

Example 6: SIP transformation of Jingle session-initiate (F3)

```
|   INVITE sip:romeo@example.net SIP/2.0
|   Via: SIP/2.0/TCP client.example.com:5060;branch=z9hG4bK74bf9
|   Max-Forwards: 70
|   From: Juliet Capulet <sip:juliet@example.com>;tag=t3hr0zny
|   To: Romeo Montague <sip:romeo@example.net>
|   Call-ID: 3848276298220188511@example.com
|   CSeq: 1 INVITE
|   Contact: <sip:juliet@client.example.com;transport=tcp>
|   Content-Type: application/sdp
|   Content-Length: 184

|   v=0
|   o=alice 2890844526 2890844526 IN IP4 client.example.com
|   s=-
|   c=IN IP4 192.0.2.101
|   t=0 0
|   m=audio 49172 RTP/AVP 18 96 97
|   a=rtpmap:96 sppex/16000
|   a=rtpmap:97 speex/8000
|   a=rtpmap:18 G729
```

The responder returns a SIP 180 Ringing message.

Example 7: SIP 180 Ringing message (F5)

```
|   SIP/2.0 180 Ringing
|   Via: SIP/2.0/TCP client.example.com:5060;branch=z9hG4bK74bf9;\
|        received=192.0.2.101
|   From: Juliet Capulet <sip:juliet@example.com>;tag=t3hr0zny
|   To: Romeo Montague <sip:romeo@example.net>;tag=v3rsch1kk3l1jk
|   Call-ID: 3848276298220188511@example.com
|   CSeq: 1 INVITE
|   Contact: <sip:romeo@client.example.net;transport=tcp>
|   Content-Length: 0
```

The gateway transforms the ringing message into XMPP syntax.

Example 8: XMPP transformation of SIP 180 Ringing message (F7)

```
|   <iq from='romeo@montague.net/v3rsch1kk3l1jk'
|       id='ol3ba71g'
|       to='juliet@example.com/t3hr0zny'
|       type='set'>
|     <jingle xmlns='urn:xmpp:jingle:1'
|             action='session-info'
|             initiator='juliet@example.com/t3hr0zny'
|             sid='a73sjjvkla37jfea'>
|       <ringing xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
|     </jingle>
|   </iq>
```

The initiator returns an IQ-result acknowledging receipt of the ringing message, which is used only by the gateway and not transformed into SIP syntax.

Example 9: XMPP entity acknowledges ringing message (F8)

```
|  <iq from='juliet@example.com/t3hr0zny'
|      id='ol3ba71g'
|      to='romeo@example.net/v3rsch1kk3l1jk'
|      type='result'/>
```

The responder sends a SIP 200 OK to the initiator in order to accept the session initiation request.

Example 10: SIP user accepts session request (F9)

```
|   SIP/2.0 200 OK
|   Via: SIP/2.0/TCP client.example.com:5060;branch=z9hG4bK74bf9;\
|        received=192.0.2.101
|   From: Juliet Capulet <sip:juliet@example.com>;tag=t3hr0zny
|   To: Romeo Montague <sip:romeo@example.net>;tag=v3rsch1kk3l1jk
|   Call-ID: 3848276298220188511@example.com
|   CSeq: 1 INVITE
|   Contact: <sip:romeo@client.example.net;transport=tcp>
|   Content-Type: application/sdp
|   Content-Length: 147
|
|   v=0
|   o=romeo 2890844527 2890844527 IN IP4 client.example.net
|   s=-
|   c=IN IP4 192.0.2.201
|   t=0 0
|   m=audio 3456 RTP/AVP 97
|   a=rtpmap:97 speex/8000
```

The gateway transforms the 200 OK into a Jingle session-accept action.

Example 11: XMPP transformation of SIP 200 OK (F11)

```
|   <iq from='romeo@example.net/v3rsch1kk3l1jk'
|       id='pd1bf839'
|       to='juliet@example.com/t3hr0zny'
|       type='set'>
|     <jingle xmlns='urn:xmpp:jingle:1'
|             action='session-accept'
|             initiator='juliet@example.com/t3hr0zny'
|             responder='romeo@example.net/v3rsch1kk3l1jk'
|             sid='a73sjjvkla37jfea'>
|       <content creator='initiator'
|                media='audio'
|                name='this-is-the-audio-content'>
|         <description xmlns='urn:xmpp:jingle:app:rtp:1'>
|           <payload-type id='97' name='speex' clockrate='8000'/>
|         </description>
|         <transport xmlns='urn:xmpp:jingle:transport:raw-udp'>
|           <candidate id='9eg13am7' ip='192.0.2.101'
|                      port='49172' generation='0'/>
|         </transport>
|       </content>
|     </jingle>
|   </iq>
```

If the payload types and transport candidate can be successfully used by both parties, then the initiator acknowledges the session-accept action.

Example 12: XMPP user acknowledges session-accept (F12)

```
|   <iq from='romeo@example.net/v3rsch1kk3l1jk'
|       id='pd1bf839'
|       to='juliet@example.com/t3hr0zny'
|       type='result'/>
```

The parties now begin to exchange media. In this case they would exchange audio using the Speex codec at a clockrate of 8000 since that is the highest-priority codec for the responder (as determined by the XML order of the <payloadtype/> children).

The parties can continue the session as long as desired.

Eventually, one of the parties (in this case the responder) terminates the session.

Example 13: SIP user ends session (F15)

```
|   BYE sip:juliet@client.example.com SIP/2.0
|   Via: SIP/2.0/TCP client.example.net:5060;branch=z9hG4bKnashds7
|   Max-Forwards: 70
|   From: Romeo Montague <sip:romeo@example.net>;tag=8321234356
|   To: Juliet Capulet <sip:juliet@example.com>;tag=9fxced76sl
|   Call-ID: 3848276298220188511@example.com
|   CSeq: 4 BYE
|   Content-Length: 0
```

The gateway transforms the SIP BYE into XMPP syntax.

Example 14: XMPP transformation of SIP BYE (F17)

```
| <iq from='romeo@example.net/v3rsch1kk3l1jk'
|     id='rv301b47'
|     to='juliet@example.com/t3hr0zny'
|     type='set'>
|   <jingle xmlns='urn:xmpp:jingle:1'
|           action='session-terminate'
|           initiator='juliet@example.com/t3hr0zny'
|           sid='a73sjjvkla37jfea'/>
|     <reason>
|       <success/>
|     </reason>
|   </jingle>
| </iq>
```

The initiator returns an IQ-result acknowledging receipt of the session termination, which is used only by the gateway and not transformed into SIP syntax.

Example 15: XMPP user acknowledges end of session (F18)

```
  <iq from='romeo@example.net/v3rsch1kk3l1jk'
    id='rv301b47'
    to='juliet@example.com/t3hr0zny'
    type='result'/>
```

13. IANA Considerations

This document requests no actions by the IANA.

14. Security Considerations

Detailed security considerations for session management are given for SIP in [RFC3261] and for XMPP in [XEP-0166] (see also [RFC6120]). The security considerations provided in [RFC7247] also apply.

The addition of gateways to the security model of media signaling introduces some new risks. In particular, end-to-end security properties (especially confidentiality and integrity) between media endpoints that interface through a gateway can be provided only if common formats are supported. Specification of those common formats is out of scope for this document and, unfortunately, no generalized method for end-to-end encryption of signaling messages has yet been defined, even outside of recognized standards development organizations (e.g., [RFC3862] and [RFC3923] are not widely implemented and Off-the-Record Messaging [OTR] can handle only human-readable chat messages, not structured signaling payloads).

15. References

15.1. 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.

[RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

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

[RFC4733] Schulzrinne, H. and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals", RFC 4733, December 2006.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

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

[RFC7247] Saint-Andre, P., Houri, A., and J. Hildebrand, "Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Architecture, Addresses, and Error Handling", RFC 7247, May 2014.

[XEP-0030] Hildebrand, J., Eatmon, R., and P. Saint-Andre, "Service Discovery", XSF XEP 0030, June 2008.

[XEP-0166] Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and J. Hildebrand, "Jingle", XSF XEP 0166, June 2007.

[XEP-0167] Ludwig, S., Saint-Andre, P., Egan, S., and R. McQueen, "Jingle RTP Sessions", XSF XEP 0167, February 2009.

[XEP-0176] Beda, J., Ludwig, S., Saint-Andre, P., Hildebrand, J., and S. Egan, "Jingle ICE-UDP Transport Method", XSF XEP 0176, February 2009.

[XEP-0177] Beda, J., Saint-Andre, P., Ludwig, S., Hildebrand, J., and S. Egan, "Jingle Raw UDP Transport", XSF XEP 0177, February 2009.

15.2. Informative References

[I-D.ietf-mmusic-trickle-ice-sip] Ivov, E., Marocco, E., and C. Holmberg, "A Session Initiation Protocol (SIP) usage for Trickle ICE", draft-ietf-mmusic-trickle-ice-sip-02 (work in progress), July 2015.

[I-D.ietf-mmusic-trickle-ice]
Ivov, E., Rescorla, E., and J. Uberti, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", draft-ietf-mmusic-trickle-ice-02 (work in progress), January 2015.

[OTR] Ian Goldberg, , "Off-the-Record Messaging", <https://otr.cypherpunks.ca/>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3555] Casner, S. and P. Hoschka, "MIME Type Registration of RTP Payload Formats", RFC 3555, July 2003.

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

[RFC3923] Saint-Andre, P., "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)", RFC 3923, October 2004.

[RFC3959] Camarillo, G., "The Early Session Disposition Type for the Session Initiation Protocol (SIP)", RFC 3959, December 2004.

[RFC3960] Camarillo, G. and H. Schulzrinne, "Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP)", RFC 3960, December 2004.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC4856] Casner, S., "Media Type Registration of Payload Formats in the RTP Profile for Audio and Video Conferences", RFC 4856, DOI 10.17487/RFC4856, February 2007, <http://www.rfc-editor.org/info/rfc4856>.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S., and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC5768] Rosenberg, J., "Indicating Support for Interactive Connectivity Establishment (ICE) in the Session Initiation Protocol (SIP)", RFC 5768, April 2010.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

[RFC7081] Ivov, E., Saint-Andre, P., and E. Marocco, "CUSAX: Combined Use of the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP)", RFC 7081, November 2013.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7395] Stout, L., Moffitt, J., and E. Cestari, "An Extensible Messaging and Presence Protocol (XMPP) Subprotocol for WebSocket", RFC 7395, October 2014.

[XEP-0124] Paterson, I., Smith, D., Saint-Andre, P., Moffitt, J., and L. Stout, "Bidirectional-streams Over Synchronous HTTP (BOSH)", XSF XEP 0124, November 2013.

[XEP-0234]
Saint-Andre, P., "Jingle File Transfer", XSF XEP 0234, August 2014.

[XEP-0269]
Cionoiu, D. and P. Saint-Andre, "Jingle Early Media", XSF XEP 0269, May 2009.

Appendix A. Acknowledgements

Thanks to Dave Crocker, Philipp Hancke, Paul Kyzivat, and Jonathan Lennox for their feedback. Jonathan in particular provided helpful suggestions regarding the transport fallback section.

The authors gratefully acknowledge the assistance of Markus Isomaki and Yana Stamcheva as the working group chairs and Ben Campbell as the sponsoring Area Director.

Peter Saint-Andre wishes to acknowledge Cisco Systems, Inc., for employing him during his work on earlier draft versions of this document.

Authors' Addresses

Peter Saint-Andre
&yet

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

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

Email: saul@ag-projects.com

Emil Ivov
Jitsi
Strasbourg 67000
France

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