XEP-0167: Jingle RTP Sessions

Scott Ludwig
mailto:scottlu@google.com
xmpp:scottlu@google.com

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

Sean Egan
mailto:seanegan@google.com
xmpp:seanegan@google.com

Robert McQueen
mailto:robert.mcqueen@collabora.co.uk
xmpp:robert.mcqueen@collabora.co.uk

Diana Cionoiu
mailto:diana@null.ro
xmpp:l-fy@jabber.null.ro

2020-04-22
Version 1.2.0

Status	Type	Short Name
Draft	Standards Track	jingle-rtp

This specification defines a Jingle application type for negotiating one or more sessions that use the Real-time Transport Protocol (RTP) to exchange media such as voice or video. The application type includes a straightforward mapping to Session Description Protocol (SDP) for interworking with SIP media endpoints.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

Jingle (XEP-0166) [1] can be used to initiate and negotiate a wide range of peer-to-peer sessions. One session type of interest is media such as voice or video. This document specifies an application format for negotiating Jingle media sessions, where the media is exchanged over the Realtime Transport Protocol (RTP; see RFC 3550 [2]).

2 Requirements

The Jingle application format defined herein is designed to meet the following requirements:

1. Enable negotiation of parameters necessary for media sessions using the Realtime Transport Protocol (RTP).
2. Map these parameters to Session Description Protocol (SDP; see RFC 4566 [3]) to enable interoperability.
3. Define informational messages related to typical RTP uses such as audio chat and video chat (e.g., ringing, on hold, on mute).

3 Jingle Conformance

In accordance with Section 10 of XEP-0166, this document specifies the following information related to the Jingle RTP application type:

1. The application format negotiation process is defined in the Negotiating a Jingle RTP Session section of this document.
2. The semantics of the <description/> element are defined in the Application Format section of this document.
3. A mapping of Jingle semantics to the Session Description Protocol is provided in the Mapping to Session Description Protocol section of this document.
4. A Jingle RTP session SHOULD use a datagram transport method (e.g. Jingle Raw UDP Transport Method (XEP-0177) [4] or the "ice-udp" method specified in Jingle ICE-UDP

[1] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[2] RFC 3550: RTP: A Transport Protocol for Real-Time Applications <http://tools.ietf.org/html/rfc3550>.
[3] RFC 4566: SDP: Session Description Protocol <http://tools.ietf.org/html/rfc4566>.
[4] XEP-0177: Jingle Raw UDP Transport Method <https://xmpp.org/extensions/xep-0177.html>.

Transport Method (XEP-0176) [5]), but MAY use a streaming transport if the end-to-end link has minimal latency and the media negotiated is not unduly heavy (e.g., it might be possible to use a streaming transport for audio, but not for video).

5. Jingle RTP supports two components: one for RTP itself and one for the Real Time Control Protocol (RTCP). The component numbered "1" MUST be associated with RTP and the component numbered "2" MUST be associated with RTCP. Even if an implementation does not support RTCP, it MUST accept Jingle content types that include component "2" by mirroring the second component in its replies (however, it would simply ignore the RTCP-related data during the RTP session).

6. Content is to be sent and received as follows:

   - For datagram transports, outbound content shall be encoded into RTP packets and each packet shall be sent individually over the transport. Each inbound packet received over the transport is an RTP packet.

   - For streaming transports, outbound content shall be encoded into RTP packets, framed in accordance with RFC 4571 [6], and sent in succession over the transport. Incoming data received over the transport shall be processed as a stream of RTP packets, where each RTP packet boundary marks the location of the next packet.

4 Application Format

A Jingle RTP session is described by a content type that contains one application format and one transport method. Each <content/> element defines a single RTP session. A Jingle negotiation MAY result in the establishment of multiple RTP sessions (e.g., one for audio and one for video). An application SHOULD consider all of the RTP sessions that are established via the same Jingle negotiation to be synchronized for purposes of streaming, playback, recording, etc.
RTP as defined in RFC 3550 is used in the context of various "profiles" that are defined by other specifications. Jingle RTP treats RTP profiles as follows:

1. By default the RTP profile in Jingle RTP MUST be considered "RTP/AVP" as defined in RFC 3551 [7].

[5] XEP-0176: Jingle ICE-UDP Transport Method <https://xmpp.org/extensions/xep-0176.html>.
[6] RFC 4571: Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport <http://tools.ietf.org/html/rfc4571>.
[7] RFC 3551: RTP Profile for Audio and Video Conferences with Minimal Control <http://tools.ietf.org/html/rfc3551>.

2. If the session initiation request contains an <encryption/> element to specify use of SRTP as described under Negotiation of SRTP, then the RTP profile MUST instead be considered "RTP/SAVP" as defined in RFC 3711 [8].

3. Future versions of this specification might define how to use other RTP profiles, such as "RTP/AVPF" and "RTP/SAVPF" as defined in RFC 4585 [9] and RFC 5124 [10] respectively.

The application format consists of one or more encodings contained within a wrapper <description/> element qualified by the 'urn:xmpp:jingle:apps:rtp:1' namespace (see Namespace Versioning regarding the possibility of incrementing the version number). In the language of RFC 4566 each encoding is a payload-type; therefore, each <payload-type/> element specifies an encoding that can be used for the RTP stream, as illustrated in the following example.

```
<description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
      <payload-type id='96' name='speex' clockrate='16000'/>
      <payload-type id='97' name='speex' clockrate='8000'/>
      <payload-type id='18' name='G729'/>
      <payload-type id='103' name='L16' clockrate='16000' channels='2'
         />
      <payload-type id='98' name='x-ISAC' clockrate='8000'/>
      <payload-type id='102' name='iLBC'/>
      <payload-type id='4' name='G723'/>
      <payload-type id='0' name='PCMU' clockrate='16000'/>
      <payload-type id='8' name='PCMA'/>
      <payload-type id='13' name='CN'/>
      <rtcp-mux/>
    </description>
```

The <description/> element is intended to be a child of a Jingle <content/> element as specified in XEP-0166.
The <description/> element MUST possess a 'media' attribute that specifies the media type, such as "audio" or "video", where the media type SHOULD be as registered at IANA MIME Media Types Registry [11].
The <description/> element MAY possess a 'ssrc' attribute that specifies the 32-bit synchronization source for this media stream, as defined in RFC 3550.
After inclusion of one or more <payload-type/> child elements, the <description/> element MAY also contain a <bandwidth/> element that specifies the allowable or preferred bandwidth for use by this application type. The 'type' attribute of the <bandwidth/> element SHOULD be a value for the SDP "bwtype" parameter as listed in the IANA Session Description Protocol

[8] RFC 3711: The Secure Real-time Transport Protocol (SRTP) <http://tools.ietf.org/html/rfc3711>.
[9] RFC 4585: Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF) <http://tools.ietf.org/html/rfc4585>.
[10] RFC 5124: Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF) <http://tools.ietf.org/html/rfc5124>.
[11] IANA registry of MIME media types <http://www.iana.org/assignments/media-types>.

Parameters Registry [12]. For RTP sessions, often the <bandwidth/> element will specify the "session bandwidth" as described in Section 6.2 of RFC 3550, measured in kilobits per second as described in Section 5.2 of RFC 4566.
Also, the <description/> element MAY contain a <rtcp-mux/> element that specifies the ability to multiplex RTP Data and Control Packets on a single port as described in RFC 5761 [13].
The encodings SHOULD be provided in order of preference by placing the most-preferred payload type as the first <payload-type/> child of the <description/> element and the least-preferred payload type as the last child.
The attributes of the <payload-type/> element are as follows:

Attribute	Description	Datatype	Inclusion
channels	The number of channels; if omitted, it MUST be assumed to contain one channel	unsignedByte (defaults to 1)	RECOMMENDED
clockrate	The sampling frequency in Hertz	unsignedInt	RECOMMENDED
id	The payload identifier	unsignedByte	REQUIRED
maxptime	Maximum packet time as specified in RFC 4566	unsignedInt	OPTIONAL
name	The appropriate subtype of the MIME type	string	RECOMMENDED for static payload types, REQUIRED for dynamic payload types
ptime	Packet time as specified in RFC 4566	unsignedInt	OPTIONAL

In Jingle RTP, the encodings are used in the context of RTP. The most common encodings for the Audio/Video Profile (AVP) of RTP are listed in RFC 3551 (these "static" types are reserved from payload ID 0 through payload ID 95), although other encodings are allowed (these "dynamic" types use payload IDs 96 to 127) in accordance with the dynamic assignment rules described in Section 3 of RFC 3551. The payload IDs are represented in the 'id' attribute.
Each <payload-type/> element MAY contain one or more child elements that specify particular parameters related to the payload. For example, as described in RFC 5574 [14], the "cng", "mode", and "vbr" parameters can be specified in relation to usage of the Speex [15] codec. Where such parameters are encoded via the "fmtp" SDP attribute, they shall be represented

[12] IANA registry of parameters related to the Session Description Protocol <http://www.iana.org/assignments/sdp-parameters>.
[13] RFC 5761: Multiplexing RTP Data and Control Packets on a Single Port <http://tools.ietf.org/html/rfc5761>.
[14] RFC 5574: RTP Payload Format for the Speex Codec <http://tools.ietf.org/html/rfc5574>.
[15] See <http://www.speex.org/>.

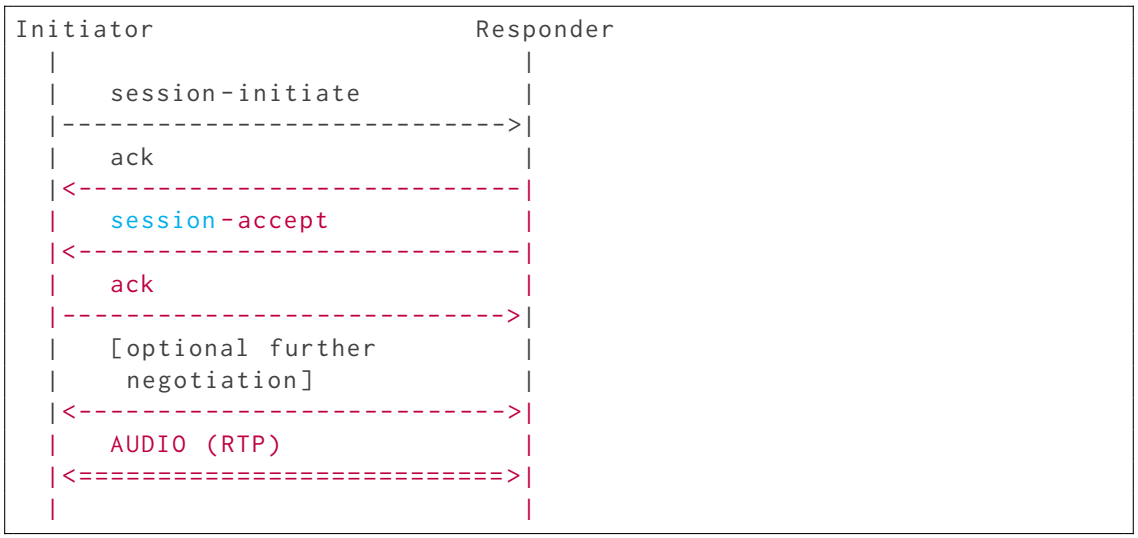
in Jingle via the following format:

```
<parameter name='foo' value='bar'/>
```

The order of parameter elements MUST be ignored.
Parameter names MUST be treated as case-sensitive.
Note: Parameter names are effectively guaranteed to be unique, since the Internet Assigned Numbers Authority (IANA) [16] maintains a registry of SDP parameters (see <http://www.iana.org/assignments/sdp-parameters>).

5 Negotiating a Jingle RTP Session

In general, the process for negotiating a Jingle RTP session is as follows:



When the initiator sends a session-initiate message to the responder, the <description/> element includes all of the payload types that the initiator can send and/or receive for Jingle RTP, each one encapsulated in a separate <payload-type/> element (the rules specified in RFC 3264 [17] SHOULD be followed regarding inclusion of payload types).

Listing 1: Initiation

```
<iq from='romeo@montague.lit/orchard'
```

[16] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[17] RFC 3264: An Offer/Answer Model with the Session Description Protocol (SDP) <http://tools.ietf.org/html/rfc3264>.

```
    id='ih28sx61'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='96' name='speex' clockrate='16000'/>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
        <payload-type id='0' name='PCMU'/>
        <payload-type id='103' name='L16' clockrate='16000' channels='
            2'/>
        <payload-type id='98' name='x-ISAC' clockrate='8000'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='asd88fgpdd777uzjYhagZg'
                 ufrag='8hhy'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='el0747fg11'
                   ip='10.0.1.1'
                   network='1'
                   port='8998'
                   priority='2130706431'
                   protocol='udp'
                   type='host'/>
        <candidate component='1'
                   foundation='2'
                   generation='0'
                   id='y3s2b30v3r'
                   ip='192.0.2.3'
                   network='1'
                   port='45664'
                   priority='1694498815'
                   protocol='udp'
                   rel-addr='10.0.1.1'
                   rel-port='8998'
                   type='srflx'/>
      </transport>
    </content>
  </jingle>
</iq>
```

Upon receiving the session-initiate stanza, the responder determines whether it can proceed with the negotiation. The general Jingle error cases are specified in XEP-0166 and illustrated

in the Scenarios section of this document.
If there is no immediate error, the responder acknowledges the session initiation request.

Listing 2: Responder acknowledges session-initiate

```
<iq from='juliet@capulet.lit/balcony'
    id='ih28sx61'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

Depending on user preferences or client configuration, a user agent controlled by a human user might need to wait for the user to affirm a desire to proceed with the session before continuing. When the user agent has received such affirmation (or if the user agent can automatically proceed for any reason, e.g. because no human intervention is expected or because a human user has configured the user agent to automatically accept sessions with a given entity), it returns a Jingle session-accept message. The session-accept message SHOULD include a subset of the payload types sent by the initiator, i.e., a list of the offered payload types that the responder can send and/or receive. The list that the responder sends SHOULD retain the ID numbers specified by the initiator. The order of the <payload-type/> elements indicates the responder's preferences, with the most-preferred type first.
In the following example, we imagine that the responder supports Speex at a clockrate of 8000 but not 16000, G729, and PCMA but not PMCU. Therefore the responder returns only two payload types (since PCMA was not offered).

Listing 3: Responder definitively accepts the session

```
<iq from='juliet@capulet.lit/balcony'
    id='i91fs6d5'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          initiator='romeo@montague.lit/orchard'
          responder='juliet@capulet.lit/balcony'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='YH75Fviy6338Vbrhrlp8Yh'
                 ufrag='9uB6'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='or2ii2syr1'
```

```
                    ip='192.0.2.1'
                    network='0'
                    port='3478'
                    priority='2130706431'
                    protocol='udp'
                    type='host'/>
      </transport>
    </content>
  </jingle>
</iq>
```

If the responder supports none of the payload-types offered by the initiator, the responder SHOULD terminate the session and include a Jingle reason of <failed-application/>.
If the responder accepts the session, the initiator acknowledges the session-accept message:

Listing 4: Initiator acknowledges session acceptance

```
<iq from='romeo@montague.lit/orchard'
    id='i91fs6d5'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

The initiator and responder would then attempt to establish connectivity for the data channel, Once they do, they would exchange media using any of the codecs that meet the following criteria:

- If the value of the 'senders' attribute is "initiator" then the initiator MAY use any codec that it can send and the responder can receive.
- If the value of the 'senders' attribute is "responder" then the responder MAY use any codec that it can send and the initiator can receive.
- If the value of the 'senders' attribute is "both" then the parties MAY use any codec that both parties can send and receive.

6 Mapping to Session Description Protocol

The SDP media type for Jingle RTP is "audio" (see Section 8.2.1 of RFC 4566) for audio media, "video" (see Section 8.2.1 of RFC 4566) for video media, etc. The media type is reflected in the Jingle 'media' attribute.
The Jingle <bandwidth/> element SHALL be mapped to an SDP b= line; in particular, the value of the 'type' attribute SHALL be mapped to the SDP <bwtype> parameter and the XML character data of the Jingle <bandwidth/> element SHALL be mapped to the SDP <bandwidth> parameter.
If the payload type is static (payload-type IDs 0 through 95 inclusive), it MUST be mapped to

an m= line as defined in RFC 4566. The generic format for this line is as follows:

```
m=<media> <port> <transport> <fmt list>
```

The SDP <media> parameter is ”audio” or ”video” or some other media type as specified by the Jingle ’media’ attribute, the <port> parameter is the preferred port for such communications (which might be determined dynamically), the <transport> parameter corresponds to the RTP profile as described under Application Format, and the <fmt list> parameter is the payload-type ID.
For example, consider the following static payload-type:

```
<description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
  <payload-type id="13" name="CN"/>
</description>
```

That Jingle-formatted information would be mapped to SDP as follows:

```
m=audio 9999 RTP/AVP 13
```

If the payload type is dynamic (payload-type IDs 96 through 127 inclusive), it SHOULD be mapped to an SDP media field plus an SDP attribute field named ”rtpmap”.
For example, consider a payload of 16-bit linear-encoded stereo audio sampled at 16KHz associated with dynamic payload-type 96:

```
<description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
  <payload-type id='96' name='speex' clockrate='16000'/>
</description>
```

That Jingle-formatted information would be mapped to SDP as follows:

```
m=audio 9999 RTP/AVP 96
a=rtpmap:96 speex/16000
```

As noted, if additional parameters are to be specified, they shall be represented as attributes of the <parameter/> child of the <payload-type/> element, as in the following example.

```
<description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
  <payload-type id='96' name='speex' clockrate='16000' ptime='40'>
    <parameter name='vbr' value='on'/>
    <parameter name='cng' value='on'/>
  </payload-type>
</description>
```

That Jingle-formatted information would be mapped to SDP as follows:

```
m=audio 9999 RTP/AVP 96
a=rtpmap:96 speex/16000
a=ptime:40
a=fmtp:96 vbr=on;cng=on
```

The formatting is similar for video parameters, as shown in the following example.

```
<description xmlns='urn:xmpp:jingle:apps:rtp:1' media='video'>
  <payload-type id='98' name='theora' clockrate='90000'>
    <parameter name='height' value='600'/>
    <parameter name='width' value='800'/>
    <parameter name='delivery-method' value='inline'/>
    <parameter name='configuration' value='somebase16string'/>
    <parameter name='sampling' value='YCbCr-4:2:2'/>
  </payload-type>
</description>
```

That Jingle-formatted information would be mapped to SDP as follows:

```
m=video 49170 RTP/AVP 98
a=rtpmap:98 theora/90000
a=fmtp:98 sampling=YCbCr-4:2:2; width=800; height=600;
delivery-method=inline; configuration=somebase16string;
```

7 Negotiation of SRTP

RFC 3711 [18] defines the Secure Real-time Transport Protocol, and RFC 4568 [19] defines the SDP "crypto" attribute for signalling and negotiating the use of SRTP in the context of offer-answer protocols such as SIP. To enable the use of SRTP and gatewaying to non-XMPP technologies that make use of the "crypto" SDP attribute, we define a corresponding <crypto/> element qualified by the 'urn:xmpp:jingle:apps:rtp:1' namespace.
If the initiator wishes to use SRTP, the session-initiate stanza shall include an <encryption/> element, which MUST contain at least one <crypto/> element and MAY include multiple instances of the <crypto/> element. The <encryption/> element MUST be a child of the <description/> element. If the initiator requires the session to be encrypted, the <encryption/> element MUST include a 'required' attribute whose logical value is TRUE and whose lexical value is "true" or "1" [20], where this attribute defaults to a logical value of FALSE (i.e., a lexical value of "false" or "0").

[18] RFC 3711: The Secure Real-time Transport Protocol (SRTP) <http://tools.ietf.org/html/rfc3711>.
[19] RFC 4568: Session Description Protocol (SDP) Security Descriptions for Media Streams <http://tools.ietf.org/html/rfc4568>.
[20] In accordance with Section 3.2.2.1 of XML Schema Part 2: Datatypes, the allowable lexical representations for the xs:boolean datatype are the strings "0" and "false" for the concept 'false' and the strings "1" and "true" for the concept 'true'; implementations MUST support both styles of lexical representation.

The <crypto/> element is defined as empty (i.e., not containing any child elements); the XML attributes of the <crypto/> element are as follows:

- crypto-suite -- this maps to the SDP "crypto-suite" parameter and has the same semantics (i.e., it is an identifier that describes the encryption and authentication algorithms).
- key-params -- this maps to the SDP "key-params" parameter and has the same semantics (i.e., it provides one or more sets of keying material for the crypto-suite in question).
- session-params -- this maps to the SDP "session-params" parameter and has the same semantics (i.e., it provides transport-specific parameters for SRTP negotiation).
- tag -- this maps to the SDP "tag" parameter and has the same semantics (i.e., it is a decimal number used as an identifier for a particular crypto element).

An example follows.

```
<encryption xmlns='urn:xmpp:jingle:apps:rtp:1'
            required='1'>
  <crypto
      crypto-suite='AES_CM_128_HMAC_SHA1_80'
      key-params='inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz
          |2^20|1:32'
      session-params='KDR=1 UNENCRYPTED_SRTCP'
      tag='1'/>
</encryption>
```

The mapping of that data to SDP is as follows.

```
a=crypto:1 AES_CM_128_HMAC_SHA1_80
    inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:32
    KDR=1 UNENCRYPTED_SRTCP
```

When the responder receives a session-initiate message containing an <encryption/> element, the responder MUST do one of the following:

1. Attempt to proceed with an encrypted session by including the acceptable credentials (i.e., the relevant <crypto/> element) in its session-accept message.
2. Attempt to proceed with an unencrypted session by not including any <crypto/> element in its session-accept message (it is up to the initiator to reject this attempt if desired).
3. Reject the initiator's offer by sending a session-terminate message with a Jingle reason of <security-error/> (typically with an RTP-specific condition of <invalid-crypto/>).

Which of these the responder does is a matter of personal security policies or client configuration.

Listing 5: Responder terminates session because of invalid crypto

```
<iq from='juliet@capulet.lit/balcony'
    id='nv71c396'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason>
      <security-error/>
      <invalid-crypto xmlns='urn:xmpp:jingle:apps:rtp:errors:1'/>
    </reason>
  </jingle>
</iq>
```

If the responder requires encryption but the initiator did not include an <encryption/> element in its offer, the responder MUST reject the offer by sending a session-terminate message with a Jingle reason of <security-error/> and an RTP-specific condition of <crypto-required/>.

Listing 6: Responder terminates session because crypto is required

```
<iq from='juliet@capulet.lit/balcony'
    id='nv71c396'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason>
      <security-error/>
      <crypto-required xmlns='urn:xmpp:jingle:apps:rtp:errors:1'/>
    </reason>
  </jingle>
</iq>
```

If the initiator requires encryption but the responder does not include an <encryption/> element in its session acceptance, the initiator MUST terminate the session with a Jingle reason of <security-error/> and an RTP-specific condition of <crypto-required/>.

Listing 7: Initiator terminates session because crypto is required

```
<iq from='romeo@montague.lit/orchard'
```

```
    id='ik3hs615'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason>
      <security-error/>
      <crypto-required xmlns='urn:xmpp:jingle:apps:rtp:errors:1'/>
    </reason>
  </jingle>
</iq>
```

8 Informational Messages

Informational messages can be sent by either party within the context of Jingle to communicate the status of a Jingle RTP session, device, or principal. The informational message MUST be an IQ-set containing a <jingle/> element of type "session-info", where the informational message is a payload element qualified by the 'urn:xmpp:jingle:apps:rtp:info:1' namespace. The following payload elements are defined. [21]
Note: Because an informational message is sent in an IQ-set, the receiving party MUST return either an IQ-result or an IQ-error (normally an IQ-result simply to acknowledge receipt).

8.1 Active

The <active/> payload indicates that the principal or device is again actively participating in the session after having been on mute or having put the other party on hold. The <active/> element applies to all aspects of the session, and thus does not possess a 'name' attribute.

Listing 8: Responder sends active message

```
<iq from='juliet@capulet.lit/balcony'
    id='yh3gr714'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <active xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
  </jingle>
```

[21] A <trying/> element (equivalent to the SIP 100 Trying response code) is not necessary, since each session-level message is acknowledged via XMPP IQ semantics.

```
</iq>
```

8.2 Hold

The <hold/> payload indicates that the principal is temporarily not listening for media from the other party. It is RECOMMENDED for the parties to handle informational <hold/> messages as follows (where the holdee is the party that receives the hold message and the holder is the party that sends the hold message):

- The holdee SHOULD stop sending media.
- The holdee MUST keep accepting media (this ensures that the holder can immediately start sending media again when switching back from hold to active, or can send hold music or other media).
- The holder MAY continue to send media (e.g. hold music).
- The holder MAY silently drop all media that it receives from the holdee.

Listing 9: Responder sends hold message

```
<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>
```

When the holder wishes to end the hold state, it sends an informational payload of <unhold/> or <active/>.

Listing 10: Responder ends the hold state

```
<iq from='juliet@capulet.lit/balcony'
    id='br81gd63'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <unhold xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
```

```
  </jingle>
</iq>
```

8.3 Mute

The <mute/> payload indicates that the principal is temporarily not sending media to the other party but continuing to accept media from the other party. The <mute/> element MAY possess a 'name' attribute whose value specifies a particular session to be muted (e.g., muting the audio aspect but not the video aspect of a voice+video chat). If no 'name' attribute is included, the recipient MUST assume that all sessions are to be muted.

Listing 11: Responder sends mute message

```
<iq from='juliet@capulet.lit/balcony'
    id='hg4891f5'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <mute xmlns='urn:xmpp:jingle:apps:rtp:info:1'
          creator='responder'
          name='voice'/>
  </jingle>
</iq>
```

To end the mute state, the party sends an informational payload of <unmute/> or <active/>.

Listing 12: Responder ends the mute state

```
<iq from='juliet@capulet.lit/balcony'
    id='ms91g47c'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <unmute xmlns='urn:xmpp:jingle:apps:rtp:info:1'
            creator='responder'
            name='voice'/>
  </jingle>
</iq>
```

8.4 Ringing

The <ringing/> payload indicates that the device is ringing but the principal has not yet interacted with it to answer (this maps to the SIP 180 response code).

Listing 13: Responder sends ringing message

```
<iq from='juliet@capulet.lit/balcony'
    id='tgr515bt'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <ringing xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
  </jingle>
</iq>
```

9 Exchanging Application Parameters

Before or during an RTP session, either party can share suggested application parameters with the other party by sending a Jingle stanza with an action of "description-info". The stanza shall contain only a <description/> element, which specifies suggested parameters for a given application type (e.g., a change to the height and width for display of a video stream). An example follows.

Listing 14: Entity sends application parameters

```
<iq from='romeo@montague.lit/orchard'
    id='pq6x5v37'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='description-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='webcam'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='video'>
        <payload-type id='98' name='theora' clockrate='90000'>
          <parameter name='height' value='768'/>
          <parameter name='width' value='1024'/>
        </payload-type>
      </description>
    </content>
  </jingle>
</iq>
```

The description-info message SHOULD include only the modified codecs, not the complete set of codecs (if those codecs have not changed). Their order is NOT meaningful. Furthermore, the data provided is purely advisory; the session SHOULD NOT fail if the receiving party cannot adjust its parameters accordingly.

10 Determining Support

To advertise its support for Jingle RTP Sessions and specific media types for RTP, when replying to Service Discovery (XEP-0030) [22] information requests an entity MUST return the following features:

- URNs for any version of this protocol that the entity supports -- e.g., "urn:xmpp:jingle:apps:rtp:1" for this version and "urn:xmpp:jingle:apps:rtp:0" for the previous version (see Namespace Versioning regarding the possibility of incrementing the version number)
- URNs for all of the media types that the entity supports -- e.g., "urn:xmpp:jingle:apps:rtp:audio" for RTP audio and "urn:xmpp:jingle:apps:rtp:video" for RTP video [23]

An example follows.

Listing 15: Service discovery information request

```
<iq from='romeo@montague.lit/orchard'
    id='bh3vd715'
    to='juliet@capulet.lit/balcony'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 16: Service discovery information response

```
<iq from='juliet@capulet.lit/balcony'
    id='bh3vd715'
    to='romeo@montague.lit/orchard'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:jingle:1'/>
    <feature var='urn:xmpp:jingle:apps:rtp:0'/>
    <feature var='urn:xmpp:jingle:apps:rtp:1'/>
    <feature var='urn:xmpp:jingle:apps:rtp:audio'/>
```

[22] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

[23] Support for the "audio" or "video" media type does not necessarily mean that the application supports all subtypes associated with those media types.

```
    <feature var='urn:xmpp:jingle:apps:rtp:video'/>
  </query>
</iq>
```

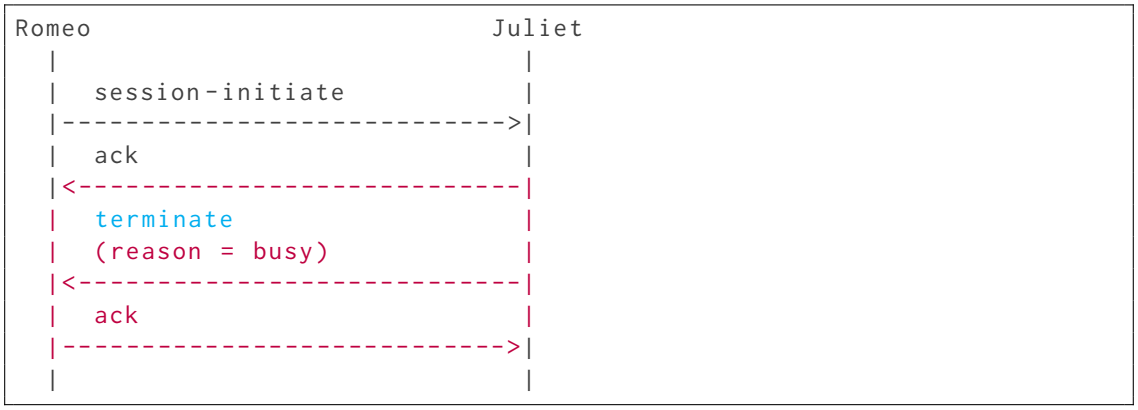
In order for an application to determine whether an entity supports this protocol, where possible it SHOULD use the dynamic, presence-based profile of service discovery defined in Entity Capabilities (XEP-0115) [24]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

11 Scenarios

The following sections show a number of Jingle RTP scenarios, roughly in order of increasing complexity.

11.1 Responder is Busy

In this scenario, Romeo initiates a voice chat with Juliet but she is otherwise engaged.
The session flow is as follows:



The protocol flow is as follows.

Listing 17: Initiator sends session-initiate

```
<iq from='romeo@montague.lit/orchard'
    id='rg6s5134'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
```

[24] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

```
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='96' name='speex' clockrate='16000'/>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
        <payload-type id='103' name='L16' clockrate='16000' channels='
            2'/>
        <payload-type id='98' name='x-ISAC' clockrate='8000'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='asd88fgpdd777uzjYhagZg'
                 ufrag='8hhy'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='el0747fg11'
                   ip='10.0.1.1'
                   network='1'
                   port='8998'
                   priority='2130706431'
                   protocol='udp'
                   type='host'/>
        <candidate component='1'
                   foundation='2'
                   generation='0'
                   id='y3s2b30v3r'
                   ip='192.0.2.3'
                   network='1'
                   port='45664'
                   priority='1694498815'
                   protocol='udp'
                   rel-addr='10.0.1.1'
                   rel-port='8998'
                   type='srflx'/>
      </transport>
    </content>
  </jingle>
</iq>
```

Listing 18: Responder acknowledges session-initiate

```
<iq from='juliet@capulet.lit/balcony'
    id='rg6s5134'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

However, the responder immediately terminates the session.

Listing 19: Responder terminates the session

```
<iq from='juliet@capulet.lit/balcony'
    id='ch3vs61d'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason>
      <busy/>
    </reason>
  </jingle>
</iq>
```

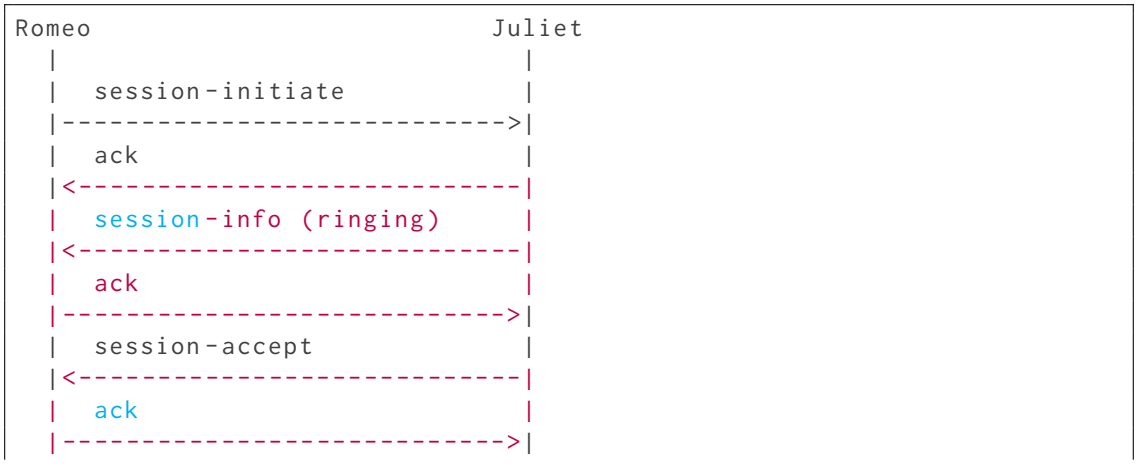
Note: It might be wondered why the responder does not accept the session and then terminate. That order would be acceptable, too, but here we assume that the responder's client has immediate information about the responder's free/busy status (e.g., because the responder is on the phone) and therefore returns an automated busy signal without requiring user interaction.

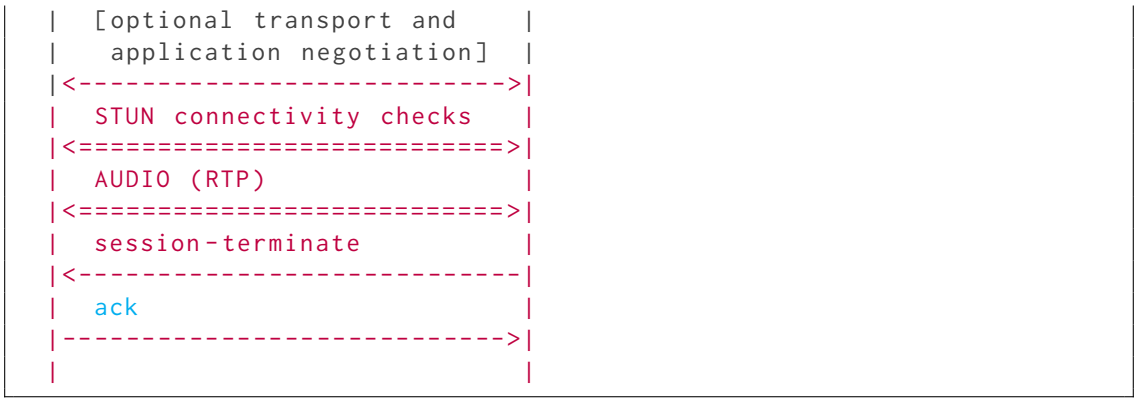
Listing 20: Initiator acknowledges termination

```
<iq from='romeo@montague.lit/orchard'
    id='ch3vs61d'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

11.2 Jingle Audio via RTP, Negotiated with ICE-UDP

In this scenario, Romeo initiates a voice chat with Juliet using a transport method of ICE-UDP. The parties also exchange informational messages.
The session flow is as follows:





The protocol flow is as follows.

Listing 21: Initiator sends session-initiate

```
<iq from='romeo@montague.lit/orchard'
    id='ds9864v6'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='96' name='speex' clockrate='16000'/>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
        <payload-type id='103' name='L16' clockrate='16000' channels='
           2'/>
        <payload-type id='98' name='x-ISAC' clockrate='8000'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='asd88fgpdd777uzjYhagZg'
                 ufrag='8hhy'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='el0747fg11'
                   ip='10.0.1.1'
                   network='1'
                   port='8998'
                   priority='2130706431'
                   protocol='udp'
                   type='host'/>
        <candidate component='1'
                   foundation='2'
```

```
                    generation='0'
                    id='y3s2b30v3r'
                    ip='192.0.2.3'
                    network='1'
                    port='45664'
                    priority='1694498815'
                    protocol='udp'
                    rel-addr='10.0.1.1'
                    rel-port='8998'
                    type='srflx'/>
      </transport>
    </content>
  </jingle>
</iq>
```

Listing 22: Responder acknowledges session-initiate

```
<iq from='juliet@capulet.lit/balcony'
    id='ds9864v6'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

Listing 23: Responder sends ringing message

```
<iq from='juliet@capulet.lit/balcony'
    id='ed81vd64'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <ringing xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
  </jingle>
</iq>
```

Listing 24: Initiator acknowledges ringing message

```
<iq from='romeo@montague.lit/orchard'
    id='ed81vd64'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

As soon as possible, the responder's client sends a session-accept message to the initiator.

Listing 25: Responder sends session-accept

```
<iq from='juliet@capulet.lit/balcony'
    id='lj3bf87g'
```

```
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          initiator='romeo@montague.lit/orchard'
          responder='juliet@capulet.lit/balcony'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='YH75Fviy6338Vbrhrlp8Yh'
                 ufrag='9uB6'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='or2ii2syr1'
                   ip='192.0.2.1'
                   network='0'
                   port='3478'
                   priority='2130706431'
                   protocol='udp'
                   type='host'/>
      </transport>
    </content>
  </jingle>
</iq>
```

The initiator acknowledges the session-accept message.

Listing 26: Initiator acknowledges session-accept

```
<iq from='romeo@montague.lit/orchard'
    id='lj3bf87g'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

Once connectivity is established (which might necessitate the exchange of additional candidates via transport-info messages), the parties begin to exchange media. In this case they would use RTP to 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 <payload-type/> children).
The parties can continue the session as long as desired.
Eventually, one of the parties terminates the session.

Listing 27: Responder terminates the session

```
<iq from='juliet@capulet.lit/balcony'
    id='wps8b597'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason>
      <success/>
      <text>Sorry, gotta go!</text>
    </reason>
  </jingle>
</iq>
```

The other party then acknowledges termination of the session:

Listing 28: Initiator acknowledges termination

```
<iq from='romeo@montague.lit/orchard'
    id='wps8b597'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

11.3 Jingle Audio via SRTP, Negotiated with ICE-UDP

In this scenario, Romeo initiates a secure voice chat with Juliet using a transport method of ICE-UDP. The parties also exchange informational messages.
The session flow is as follows:

```
Romeo                          Juliet
  |                              |
  |  session-initiate            |
  |  (with keying material)      |
  |----------------------------->|
  |  ack                         |
  |<-----------------------------|
  |  session-info (ringing)      |
  |<-----------------------------|
  |  ack                         |
  |----------------------------->|
  |  session-accept              |
  |<-----------------------------|
  |  ack                         |
  |----------------------------->|
  |  [optional transport and     |
  |   application negotiation]   |
```

```
  |<--------------------------->|
  |  STUN connectivity checks   |
  |<===========================>|
  |  AUDIO (RTP)                |
  |<===========================>|
  |  session-terminate          |
  |<----------------------------|
  |  ack                        |
  |---------------------------->|
  |                             |
```

The protocol flow is as follows.

Listing 29: Initiator sends session-initiate

```
<iq from='romeo@montague.lit/orchard'
    id='vy3g641x'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='96' name='speex' clockrate='16000'/>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
        <payload-type id='103' name='L16' clockrate='16000' channels='
            2'/>
        <payload-type id='98' name='x-ISAC' clockrate='8000'/>
        <encryption required='1'>
          <crypto
              crypto-suite='AES_CM_128_HMAC_SHA1_80'
              key-params='
                  inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz
                  |2^20|1:32'
              session-params='KDR=1_UNENCRYPTED_SRTCP'
              tag='1'/>
        </encryption>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='asd88fgpdd777uzjYhagZg'
                 ufrag='8hhy'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='el0747fg11'
                   ip='10.0.1.1'
```

```
                    network='1'
                    port='8998'
                    priority='2130706431'
                    protocol='udp'
                    type='host'/>
        <candidate  component='1'
                    foundation='2'
                    generation='0'
                    id='y3s2b30v3r'
                    ip='192.0.2.3'
                    network='1'
                    port='45664'
                    priority='1694498815'
                    protocol='udp'
                    rel-addr='10.0.1.1'
                    rel-port='8998'
                    type='srflx'/>
      </transport>
    </content>
  </jingle>
</iq>
```

To signal that the initiator wishes to use SRTP, the initiator's client includes keying material via the <encryption/> element (with one set of keying material per <crypto/> element). Here the initiator also signals that encryption is mandatory via the 'required' attribute.
The responder immediately acknowledges the session initiation request.

Listing 30: Responder acknowledges session-initiate

```
<iq from='juliet@capulet.lit/balcony'
    id='vy3g641x'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

If the keying material is acceptable, the responder's continues with the negotiation. If the keying material is not acceptable, the responder's client terminates the session as described under Negotiation of SRTP.

Listing 31: Responder sends ringing message

```
<iq from='juliet@capulet.lit/balcony'
    id='z2d85b6l'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
```

```
    <ringing xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
  </jingle>
</iq>
```

Listing 32: Initiator acknowledges ringing message

```
<iq from='romeo@montague.lit/orchard'
    id='z2d85b6l'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

As soon as possible, the responder's client sends a session-accept message to the initiator. In this case, the session-accept message includes a <crypto/> element to indicate that the responder finds the offered keying material acceptable.

Listing 33: Responder sends session-accept

```
<iq from='juliet@capulet.lit/balcony'
    id='ywf364b1'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          initiator='romeo@montague.lit/orchard'
          responder='juliet@capulet.lit/balcony'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
        <encryption>
          <crypto
              crypto-suite='AES_CM_128_HMAC_SHA1_80'
              key-params='
                  inline:PS1uQCVeeCFCanVmcjkpPywjNWhcYD0mXXtxaVBR
                  |2^20|1:32'
              session-params='KDR=1;UNENCRYPTED_SRTCP'
              tag='1'/>
        </encryption>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='YH75Fviy6338Vbrhrlp8Yh'
                 ufrag='9uB6'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='or2ii2syr1'
                   ip='192.0.2.1'
                   network='0'
```

```
                    port='3478'
                    priority='2130706431'
                    protocol='udp'
                    type='host'/>
      </transport>
    </content>
  </jingle>
</iq>
```

The initiator acknowledges the session-accept action.

Listing 34: Initiator acknowledges session-accept

```
<iq from='romeo@montague.lit/orchard'
    id='ywf364b1'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

Once connectivity is established (which might necessitate the exchange of additional candidates via transport-info messages), the parties begin to exchange media. In this case they would use RTP to 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 <payload-type/> children).
The parties can continue the session as long as desired.
Eventually, one of the parties terminates the session.

Listing 35: Responder terminates the session

```
<iq from='juliet@capulet.lit/balcony'
    id='ws71bf94'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason>
      <success/>
      <text>Sorry, gotta go!</text>
    </reason>
  </jingle>
</iq>
```

The other party then acknowledges termination of the session:

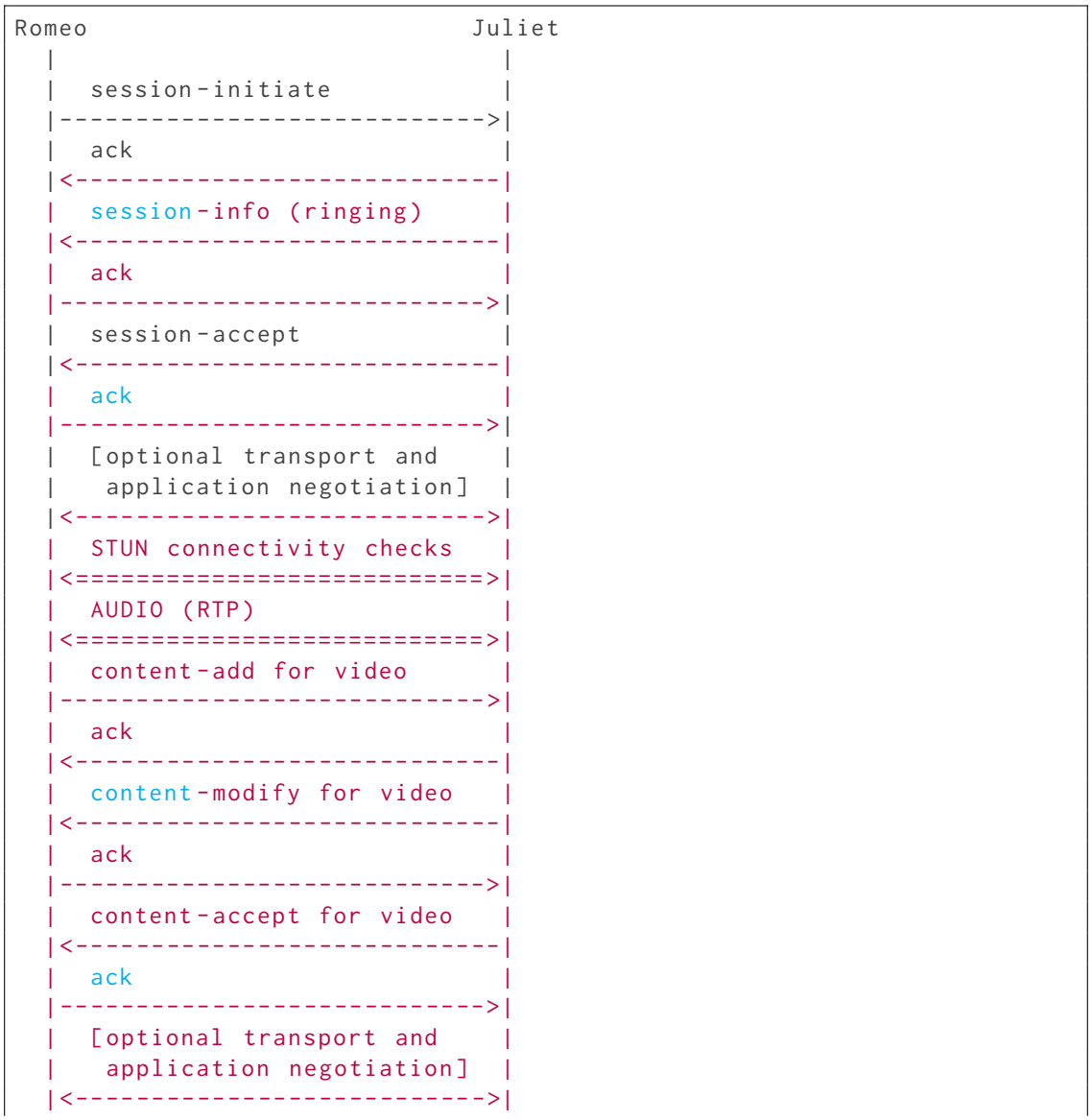
Listing 36: Initiator acknowledges termination

```
<iq from='romeo@montague.lit/orchard'
```

```
    id='ws71bf94'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

11.4 Jingle Audio and Video via RTP, Negotiated with ICE-UDP

In this scenario, Romeo initiates an audio chat with Juliet using a transport method of ICE-UDP. After the chat is active, Romeo adds video. The parties also exchange various informational messages
The session flow is as follows (some of these messages are sent in parallel):



```
|  STUN connectivity checks   |
|<=========================>|
|  AUDIO + VIDEO (RTP)        |
|<=========================>|
|  content-modify for video   |
|<--------------------------|
|  ack                        |
|-------------------------->|
|  session-terminate          |
|<--------------------------|
|  ack                        |
|-------------------------->|
|                             |
```

The protocol flow is as follows.

Listing 37: Initiation

```
<iq from='romeo@montague.lit/orchard'
    id='sf93gv76'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='96' name='speex' clockrate='16000'/>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
        <payload-type id='103' name='L16' clockrate='16000' channels='
           2'/>
        <payload-type id='98' name='x-ISAC' clockrate='8000'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                 pwd='asd88fgpdd777uzjYhagZg'
                 ufrag='8hhy'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='el0747fg11'
                   ip='10.0.1.1'
                   network='1'
                   port='8998'
                   priority='2130706431'
                   protocol='udp'
                   type='host'/>
        <candidate component='1'
```

```
                    foundation='2'
                    generation='0'
                    id='y3s2b30v3r'
                    ip='192.0.2.3'
                    network='1'
                    port='45664'
                    priority='1694498815'
                    protocol='udp'
                    rel-addr='10.0.1.1'
                    rel-port='8998'
                    type='srflx'/>
      </transport>
    </content>
  </jingle>
</iq>
```

Listing 38: Responder acknowledges session-initiate request

```
<iq from='juliet@capulet.lit/balcony'
    id='sf93gv76'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

Listing 39: Responder sends ringing message

```
<iq from='juliet@capulet.lit/balcony'
    id='nf91g647'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <ringing xmlns='urn:xmpp:jingle:apps:rtp:info:1'/>
  </jingle>
</iq>
```

Listing 40: Initiator acknowledges ringing message

```
<iq from='romeo@montague.lit/orchard'
    id='nf91g647'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

The responder sends a session-accept message to the initiator.

Listing 41: Responder sends session-accept

```
<iq from='juliet@capulet.lit/balcony'
```

```
    id='pr93b5ga'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          initiator='romeo@montague.lit/orchard'
          responder='juliet@capulet.lit/balcony'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='97' name='speex' clockrate='8000'/>
        <payload-type id='18' name='G729'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
        <candidate component='1'
                   foundation='1'
                   generation='0'
                   id='or2ii2syr1'
                   ip='192.0.2.1'
                   network='0'
                   port='3478'
                   priority='2130706431'
                   protocol='udp'
                   type='host'/>
      </transport>
    </content>
  </jingle>
</iq>
```

The initiator acknowledges the session-accept action.

Listing 42: Initiator acknowledges session-accept

```
<iq from='romeo@montague.lit/orchard'
    id='pr93b5ga'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

Once end-to-end connectivity is established (which might necessitate the exchange of additional candidates via transport-info messages), the parties begin to exchange media. In this case they would use RTP to 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 <payload-type/> children).
Romeo, being an amorous young man, requests to add video to the audio chat.

Listing 43: Adding video

```
<iq from='romeo@montague.lit/orchard'
```

```
    id='ij6s4198'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='content-add'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='webcam'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='video'>
        <payload-type id='98' name='theora' clockrate='90000'>
          <parameter name='height' value='600'/>
          <parameter name='width' value='800'/>
          <parameter name='delivery-method' value='inline'/>
          <parameter name='configuration' value='somebase16string'/>
          <parameter name='sampling' value='YCbCr-4:2:2'/>
        </payload-type>
        <payload-type id='28' name='nv' clockrate='90000'/>
        <payload-type id='25' name='CelB' clockrate='90000'/>
        <payload-type id='32' name='MPV' clockrate='90000'/>
        <bandwidth type='AS'>128</bandwidth>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:0'/>
    </content>
  </jingle>
</iq>
```

Juliet's client automatically acknowledges receipt and then asks her if she wants to add video.

Listing 44: Acknowledging request

```
<iq from='juliet@capulet.lit/balcony'
    id='ij6s4198'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

However, Juliet is having a bad hair day, so she sends a content-modify message to Romeo that sets the senders to "initiator" (she doesn't mind receiving video but she doesn't want to send video).

Listing 45: Responder sends content-modify

```
<iq from='juliet@capulet.lit/balcony'
    id='rh49l1k4'
    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' name='webcam' senders='initiator'/>
  </jingle>
</iq>
```

Listing 46: Initiator acknowledges content-modify

```
<iq from='romeo@montague.lit/orchard'
    id='rh49l1k4'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

Note: If the responder supports none of the payload-types offered by the initiator, the responder MUST reply to the content-add request with a content-reject; this message SHOULD include a Jingle reason of <failed-application/> and a list of the payload-types that the responder supports (along with an empty element for the same transport offered by the initiatior), as shown in the following example.

Listing 47: Alternate flow: responder sends content-reject

```
<iq from='juliet@montague.lit/balcony'
    id='vx2s91a6'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='content-reject'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='webcam'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='video'>
        <payload-type id='101' name='H263-1998' clockrate='90000'/>
        <payload-type id='102' name='H263-2000' clockrate='90000'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:0'/>
    </content>
    <reason>
      <failed-application/>
    </reason>
  </jingle>
</iq>
```

However, here we assume that Juliet's client supports at least one of the payload-types offered by Romeo's client, and that she Juliet accepts the offer of adding video to the session.

Listing 48: Responder accepts additional content-type

```
<iq from='juliet@capulet.lit/balcony'
    id='ih481v7s'
    to='romeo@montague.lit/orchard'
    type='set'>
```

```
  <jingle xmlns='urn:xmpp:jingle:1'
          action='content-accept'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='webcam'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='video'>
        <payload-type id='98' name='theora' clockrate='90000'>
          <parameter name='height' value='600'/>
          <parameter name='width' value='800'/>
          <parameter name='delivery-method' value='inline'/>
          <parameter name='configuration' value='somebase16string'/>
          <parameter name='sampling' value='YCbCr-4:2:2'/>
        </payload-type>
        <bandwidth type='AS'>128</bandwidth>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:ice-udp:0'/>
    </content>
  </jingle>
</iq>
```

Romeo then acknowledges the content-accept.

Listing 49: Initiator acknowledges content-accept

```
<iq from='romeo@montague.lit/orchard'
    id='ih481v7s'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

If necessary, the parties then negotiate transport methods and codec parameters for video, and Romeo starts to send video to Juliet, where the video is exchanged using the Theora codec with a height of 600 pixels, a width of 800 pixels, a bandwidth limit of 128,000 kilobits per second, etc.
After some number of minutes, Juliet returns and agrees to start sending video, too.

Listing 50: Responder sends content-modify message

```
<iq from='juliet@capulet.lit/balcony'
    id='di492bf8'
    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' name='webcam' senders='both'/>
  </jingle>
</iq>
```

Listing 51: Initiator acknowledges content-modify message

```
<iq from='romeo@montague.lit/orchard'
    id='di492bf8'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

Now the two parties would exchange audio and video in both directions.
The parties can continue the session as long as desired.
Other events might occur throughout the life of the session. For example, one of the parties might want to tweak the video parameters using a description-info action.

Listing 52: Initiator sends changes to application parameters

```
<iq from='romeo@montague.lit/orchard'
    id='xu3bg810'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='description-info'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='webcam'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='video'>
        <payload-type id='98' name='theora' clockrate='90000'>
          <parameter name='height' value='768'/>
          <parameter name='width' value='1024'/>
          <parameter name='delivery-method' value='inline'/>
          <parameter name='configuration' value='somebase16string'/>
          <parameter name='sampling' value='YCbCr-4:2:2'/>
        </payload-type>
        <bandwidth type='AS'>128</bandwidth>
      </description>
    </content>
  </jingle>
</iq>
```

Listing 53: Responder acknowledges description-info

```
<iq from='juliet@capulet.lit/balcony'
    id='xu3bg810'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

Eventually, one of the parties terminates the session.

Listing 54: Initiator sends session-terminate

```
<iq from='romeo@montague.lit/orchard'
```

```
    id='fl2v387j'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason>
      <success/>
      <text>I&apos;m outta here!</text>
    </reason>
  </jingle>
</iq>
```

Listing 55: Responder acknowledges session-terminate

```
<iq from='juliet@capulet.lit/balcony'
    id='fl2v387j'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

12 Implementation Notes

12.1 DTMF

XMPP applications that use Jingle RTP sessions for voice chat MUST support and prefer native RTP methods of communicating DTMF information, in particular the "audio/telephone-event" and "audio/tone" media types. It is NOT RECOMMENDED to use the protocol described in Jingle DTMF (XEP-0181) [25] for communicating DTMF information with RTP-aware endpoints.

12.2 When to Listen for Audio

When the Jingle RTP content type is accepted via a session-accept action, both initiator and responder SHOULD start listening for audio as defined by the negotiated transport method and audio application format. For interoperability with telephony systems, after the responder acknowledges the session initiation request, the responder SHOULD send a "ringing" message and both parties SHOULD play any audio received. For more detailed suggestions in the context of early media, see Jingle Early Media (XEP-0269) [26].

[25] XEP-0181: Jingle DTMF <https://xmpp.org/extensions/xep-0181.html>.
[26] XEP-0269: Jingle Early Media <https://xmpp.org/extensions/xep-0269.html>.

13 Security Considerations

In order to secure the data stream, implementations SHOULD use encryption methods appropriate to the RTP data transport. It is RECOMMENDED to use SRTP as defined in the Negotiation of SRTP section of this document. The SRTP keying material SHOULD (1) be tied to a separate, secure connection such as provided by DTLS (RFC 4347 [27]) where the keys are established as described in DTLS-SRTP [28] and/or (2) protected by sending the Jingle signalling over a secure channel that protects the confidentiality and integrity of the SRTP-related signalling data.
While it is also possible to use native RTP methods, such as RFC 6189 [29] as described in Use of ZRTP in Jingle RTP Sessions (XEP-0262) [30], this specification does not actively encourage or discourage the use of such methods.

14 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [31].

15 XMPP Registrar Considerations

15.1 Protocol Namespaces

This specification defines the following XML namespaces:

- urn:xmpp:jingle:apps:rtp:1
- urn:xmpp:jingle:apps:rtp:errors:1
- urn:xmpp:jingle:apps:rtp:info:1

The XMPP Registrar [32] includes the foregoing namespaces in its registry at <https://xmpp.org/registrar/namespaces.html>, as governed by XMPP Registrar Function (XEP-

[27] RFC 4347: Datagram Transport Layer Security <http://tools.ietf.org/html/rfc4347>.
[28] Datagram Transport Layer Security (DTLS) Extension to Establish Keys for Secure Real-time Transport Protocol (SRTP) <http://tools.ietf.org/html/draft-ietf-avt-dtls-srtp>. Work in progress.
[29] RFC 6189: ZRTP: Media Path Key Agreement for Unicast Secure RTP <http://tools.ietf.org/html/rfc6189>.
[30] XEP-0262: Use of ZRTP in Jingle RTP Sessions <https://xmpp.org/extensions/xep-0262.html>.
[31] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.
[32] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.

0053) [33].

15.2 Namespace Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

15.3 Service Discovery Features

For each RTP media type that an entity supports, it MUST advertise support for the "urn:xmpp:jingle:apps:rtp:[media]" feature, where the string "[media]" is replaced by the appropriate media type such as "audio" or "video".
The initial registry submission is as follows.

```
<var>
  <name>urn:xmpp:jingle:apps:rtp:audio</name>
  <desc>Signals support for audio sessions via RTP</desc>
  <doc>XEP-0167</doc>
</var>
<var>
  <name>urn:xmpp:jingle:apps:rtp:video</name>
  <desc>Signals support for video sessions via RTP</desc>
  <doc>XEP-0167</doc>
</var>
```

15.4 Jingle Application Formats

The XMPP Registrar includes "rtp" in its registry of Jingle application formats at <https://xmpp.org/registrar/jingle-apps.html>. The registry submission is as follows:

```
<application>
  <name>rtp</name>
  <desc>
    Jingle sessions that support media exchange
    via the Real-time Transport Protocol.
  </desc>
  <transport>datagram</transport>
  <doc>XEP-0167</doc>
</application>
```

[33] XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.

16 XML Schemas

16.1 Application Format

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:jingle:apps:rtp:1'
    xmlns='urn:xmpp:jingle:apps:rtp:1'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0167: http://www.xmpp.org/extensions/xep-0167.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='description'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='payload-type'
                    type='payloadElementType'
                    minOccurs='0'
                    maxOccurs='unbounded'/>
        <xs:element name='rtcp-mux'
                    minOccurs='0'
                    maxOccurs='1'/>
        <xs:element name='encryption'
                    type='encryptionElementType'
                    minOccurs='0'
                    maxOccurs='1'/>
        <xs:element name='bandwidth'
                    type='bandwidthElementType'
                    minOccurs='0'
                    maxOccurs='1'/>
      </xs:sequence>
      <xs:attribute name='media'
                    type='xs:NCName'
                    use='required'/>
      <xs:attribute name='ssrc'
                    type='xs:string'
                    use='optional'/>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='bandwidthElementType'>
    <xs:simpleContent>
```

```
      <xs:extension base='xs:string'>
        <xs:attribute name='type'
                      type='xs:string'
                      use='required'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name='cryptoElementType'>
    <xs:simpleContent>
      <xs:extension base='empty'>
        <xs:attribute name='crypto-suite' type='xs:NCName' use='
            required'/>
        <xs:attribute name='key-params' type='xs:string' use='required
            '/>
        <xs:attribute name='session-params' type='xs:string' use='
            optional'/>
        <xs:attribute name='tag' type='xs:string' use='required'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name='encryptionElementType'>
    <xs:sequence>
      <xs:element name='crypto'
                  type='cryptoElementType'
                  minOccurs='0'
                  maxOccurs='unbounded'/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='payloadElementType'>
    <xs:sequence>
      <xs:element name='parameter'
                  type='parameterElementType'
                  minOccurs='0'
                  maxOccurs='unbounded'/>
    </xs:sequence>
    <xs:attribute name='channels'
                  type='xs:unsignedByte'
                  use='optional'
                  default='1'/>
    <xs:attribute name='clockrate' type='xs:unsignedInt' use='optional
        '/>
    <xs:attribute name='id' type='xs:unsignedByte' use='required'/>
    <xs:attribute name='maxptime' type='xs:unsignedInt' use='optional'
        />
    <xs:attribute name='name' type='xs:string' use='optional'/>
    <xs:attribute name='ptime' type='xs:unsignedInt' use='optional'/>
```

```
  </xs:complexType>

  <xs:complexType name='parameterElementType'>
    <xs:simpleContent>
      <xs:extension base='empty'>
        <xs:attribute name='name' type='xs:string' use='required'/>
        <xs:attribute name='value' type='xs:string' use='required'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

16.2 Errors

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:jingle:apps:rtp:errors:1'
    xmlns='urn:xmpp:jingle:apps:rtp:errors:1'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0167: http://www.xmpp.org/extensions/xep-0167.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='crypto-required' type='empty'/>

  <xs:element name='invalid-crypto' type='empty'/>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

16.3 Informational Messages

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:jingle:apps:rtp:info:1'
    xmlns='urn:xmpp:jingle:apps:rtp:info:1'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0167: http://www.xmpp.org/extensions/xep-0167.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='active' type='empty'/>

  <xs:element name='hold' type='empty'/>

  <xs:element name='mute' type='mutingElementType'/>

  <xs:element name='ringing' type='empty'/>

  <xs:element name='unhold' type='empty'/>

  <xs:element name='unmute' type='mutingElementType'/>

  <xs:complexType name='mutingElementType'>
    <xs:simpleContent>
      <xs:extension base='empty'>
        <xs:attribute name='creator' use='required'>
          <xs:simpleType>
            <xs:restriction base='xs:NCName'>
              <xs:enumeration value='initiator'/>
              <xs:enumeration value='responder'/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name='name'
                      type='xs:string'
                      use='required'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
```

```
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

17 Acknowledgements

Thanks to Milton Chen, Paul Chitescu, Olivier Crête, Tim Julien, Steffen Larsen, Jeff Muller, Mike Ruprecht, Sjoerd Simons, Will Thompson, Justin Uberti, Unnikrishnan Vikrama Panicker, Paul Witty, and Konstantin Kozlov for their feedback.