XEP-0166: Jingle

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

Joe Beda
mailto:jbeda@google.com
xmpp:jbeda@google.com

Peter Saint-Andre
mailto:stpeter@stpeter.im
xmpp:stpeter@jabber.org
https://stpeter.im/

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

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

Joe Hildebrand
mailto:jhildebr@cisco.com
xmpp:hildjj@jabber.org

2018-09-19
Version 1.1.2

Status	Type	Short Name
Draft	Standards Track	jingle

This specification defines an XMPP protocol extension for initiating and managing peer-to-peer media sessions between two XMPP entities in a way that is interoperable with existing Internet standards. The protocol provides a pluggable model that enables the core session management semantics (compatible with SIP) to be used for a wide variety of application types (e.g., voice chat, video chat, file transfer) and with a wide variety of transport methods (e.g., TCP, UDP, ICE, application-specific transports).

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

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 **1**

2 How It Works **2**

3 Requirements **7**

4 Terminology **8**
4.1 Glossary . . . 8
4.2 Conventions . . . 9

5 Concepts and Approach **9**
5.1 Overall Session Management . . . 10

6 Session Flow **12**
6.1 Resource Determination . . . 12
6.2 Initiation . . . 12
6.3 Responder Response . . . 14
6.3.1 Acknowledgement . . . 14
6.3.2 Errors . . . 14
6.4 Negotiation . . . 16
6.5 Acceptance . . . 17
6.6 Modifying an Active Session . . . 18
6.7 Termination . . . 18
6.8 Informational Messages . . . 23

7 Formal Definition **24**
7.1 Jingle Element . . . 24
7.2 Action Attribute . . . 27
7.2.1 content-accept . . . 27
7.2.2 content-add . . . 27
7.2.3 content-modify . . . 28
7.2.4 content-reject . . . 28
7.2.5 content-remove . . . 28
7.2.6 description-info . . . 28
7.2.7 security-info . . . 28
7.2.8 session-accept . . . 29
7.2.9 session-info . . . 29
7.2.10 session-initiate . . . 29
7.2.11 session-terminate . . . 29
7.2.12 transport-accept . . . 29
7.2.13 transport-info . . . 29
7.2.14 transport-reject . . . 30
7.2.15 transport-replace . . . 30

7.2.16 Tie Breaking Related to Jingle Actions 30
7.3 Content Element 31
7.4 Reason Element 33

8 Transport Types **34**
8.1 Datagram 34
8.2 Streaming 35

9 Security Preconditions **35**

10 Error Handling **36**

11 Determining Support **37**

12 Conformance by Using Protocols **38**
12.1 Application Formats 38
12.2 Transport Methods 38
12.3 Security Preconditions 39

13 Security Considerations **39**
13.1 Transport Security 39
13.2 Denial of Service 39
13.3 Communication Through Gateways 39
13.4 Information Exposure 40
13.5 Redirection 40

14 IANA Considerations **40**

15 XMPP Registrar Considerations **40**
15.1 Protocol Namespaces 40
15.2 Namespace Versioning 41
15.3 Jingle Application Formats Registry 41
15.4 Jingle Transport Methods Registry 41

16 XML Schemas **42**
16.1 Jingle 42
16.2 Jingle Errors 45

17 History **45**

18 Acknowledgements **46**

1 Introduction

The purpose of Jingle is to enable one-to-one, peer-to-peer media sessions between XMPP entities, where the negotiation occurs over the XMPP signalling channel and the media is exchanged over a data channel that is usually a dedicated non-XMPP transport. Jingle is designed in a modular way:

- Developers can easily plug in support for a wide variety of application types, such as voice and video chat (see Jingle RTP Sessions (XEP-0167) [1]), file transfer (see Jingle File Transfer (XEP-0234) [2]), application sharing, collaborative editing, whiteboarding, and secure transmission of end-to-end XML streams (see Jingle XML Streams (XEP-0247) [3]).

- The transport methods are also pluggable, so that Jingle implementations can use any appropriate datagram transport such as User Datagram Protocol (UDP; RFC 768 [4]) as negotiated via Jingle Raw UDP Transport Method (XEP-0177) [5] or Jingle ICE-UDP Transport Method (XEP-0176) [6], or any appropriate streaming transport such as Transmission Control Protocol (TCP; RFC 793 [7]), SOCKS5 Bytestreams (XEP-0065) [8] as negotiated via Jingle SOCKS5 Bytestreams Transport Method (XEP-0260) [9], and In-Band Bytestreams (XEP-0047) [10] as negotiated via Jingle In-Band Bytestreams Transport Method (XEP-0261) [11].

- This modular approach also extends to the security preconditions that need to be met before application data can be exchanged over a given transport, such as negotiation of Transport Layer Security (TLS; RFC 5246 [12]) for streaming transports and negotiation of Datagram Transport Layer Security (DTLS; RFC 4347 [13]) for datagram transports.

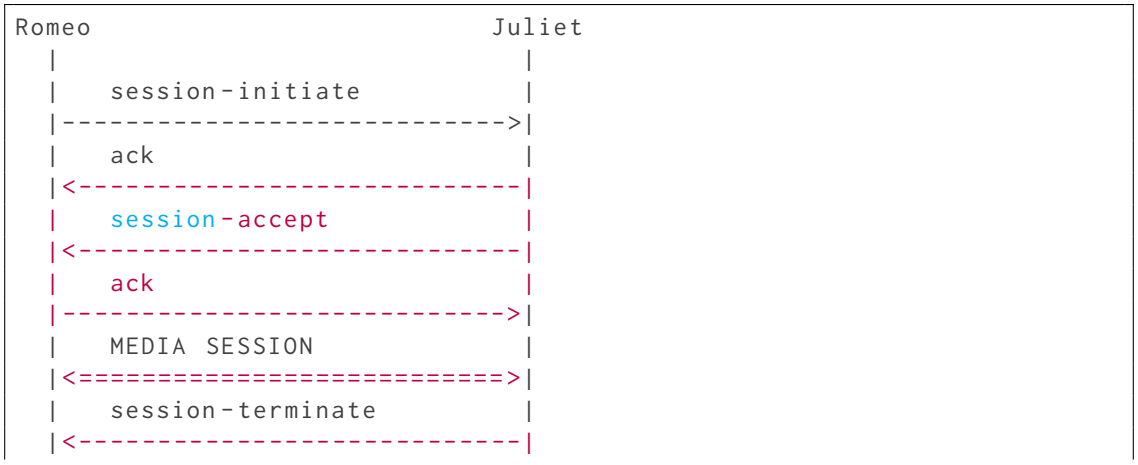
It is expected that most application types, transport methods, and security preconditions will be documented in specifications produced by the XMPP Standards Foundation (XSF) [14] or the

[1] XEP-0167: Jingle RTP Sessions <https://xmpp.org/extensions/xep-0167.html>.
[2] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.
[3] XEP-0247: Jingle XML Streams <https://xmpp.org/extensions/xep-0247.html>.
[4] RFC 768: User Datagram Protocol <http://tools.ietf.org/html/rfc0768>.
[5] XEP-0177: Jingle Raw UDP Transport Method <https://xmpp.org/extensions/xep-0177.html>.
[6] XEP-0176: Jingle ICE-UDP Transport Method <https://xmpp.org/extensions/xep-0176.html>.
[7] RFC 793: Transmission Control Protocol <http://tools.ietf.org/html/rfc0793>.
[8] XEP-0065: SOCKS5 Bytestreams <https://xmpp.org/extensions/xep-0065.html>.
[9] XEP-0260: Jingle SOCKS5 Bytestreams Transport Method <https://xmpp.org/extensions/xep-0260.html>.
[10] XEP-0047: In-Band Bytestreams <https://xmpp.org/extensions/xep-0047.html>.
[11] XEP-0261: Jingle In-Band Bytestreams Transport Method <https://xmpp.org/extensions/xep-0261.html>.
[12] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.
[13] RFC 4347: Datagram Transport Layer Security <http://tools.ietf.org/html/rfc4347>.
[14] The XMPP Standards Foundation (XSF) is an independent, non-profit membership organization that develops open extensions to the IETF's Extensible Messaging and Presence Protocol (XMPP). For further information, see <https://xmpp.org/about/xmpp-standards-foundation>.

Internet Engineering Task Force (IETF) [15]; however, developers can also define proprietary methods for custom functionality.
Although Jingle provides a general framework for session management, the original target application for Jingle was simple voice and video chat. We stress the word "simple". The purpose of Jingle was not to build a full-fledged telephony application that supports call waiting, call forwarding, call transfer, hold music, IVR systems, find-me-follow-me functionality, conference calls, and the like. These features are of interest to some user populations, but adding support for them to the core Jingle layer would introduce unnecessary complexity into a technology that is designed for simple but generalized session negotiation.
Furthermore, Jingle is not intended to supplant or replace existing Internet technologies based on the Session Initiation Protocol (SIP; RFC 3261 [16]). Because dual-stack XMPP+SIP clients are difficult to build, Jingle was designed as a pure XMPP signalling protocol. However, Jingle is at the same time designed to interwork with SIP so that the millions of deployed XMPP clients can be added onto existing Voice over Internet Protocol (VoIP) networks, rather than limiting XMPP users to a separate and distinct network.

2 How It Works

This section provides a friendly introduction to Jingle.
In essence, Jingle enables two XMPP entities (e.g., romeo@montague.lit and juliet@capulet.lit) to set up, manage, and tear down a multimedia session. The negotiation takes place over XMPP, and the media transfer typically takes place outside of XMPP. A simplified session flow would be as follows: [17]



[15] The Internet Engineering Task Force is the principal body engaged in the development of new Internet standard specifications, best known for its work on standards such as HTTP and SMTP. For further information, see <http://www.ietf.org/>.
[16] RFC 3261: Session Initiation Protocol (SIP) <http://tools.ietf.org/html/rfc3261>.
[17] Naturally, more complex scenarios are possible; such scenarios are described in other specifications, such as Jingle RTP Sessions for voice and video chat.

```
   |    ack                          |
   |-------------------------------->|
   |                                 |
```

To illustrate the basic flow, we show a truncated example with a ”stub” application format and transport method (skipping non-essential steps to enforce the most essential concepts and ignoring security preconditions for now).

Listing 1: Initiator sends session-initiate (stub)

```
<iq from='romeo@montague.lit/orchard'
    id='zid615d9'
    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='this-is-a-stub'>
      <description xmlns='urn:xmpp:jingle:apps:stub:0'/>
      <transport xmlns='urn:xmpp:jingle:transports:stub:0'/>
    </content>
  </jingle>
</iq>
```

In this example, the initiator (romeo@montague.lit/orchard) sends a session initiation offer to the responder (juliet@capulet.lit/balcony), where the session is defined as the exchange of ”stub” media over a ”stub” transport.
After the responding client acknowledges receipt of the session-initiate message (not shown here), it prompts the responding user (if any) to choose whether she wants to proceed with the session (however, it does not need to prompt the user if for example she has configured her client to automatically accept session requests from this particular initiator). If she wants to proceed she selects the appropriate interface element and her client sends a session-accept message to the initiator.

Listing 2: Responder definitively accepts the session

```
<iq from='juliet@capulet.lit/balcony'
    id='rc61n59s'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          responder='juliet@capulet.lit/balcony'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='this-is-a-stub'>
      <description xmlns='urn:xmpp:jingle:apps:stub:0'/>
      <transport xmlns='urn:xmpp:jingle:transports:stub:0'/>
```

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

The initiating client acknowledges receipt of the session-accept message (not shown here) and the parties can exchange ”stub” media data over the ”stub” transport.
Eventually, one of the parties (here the responder) will terminate the session.

Listing 3: Responder terminates the session

```
<iq from='juliet@capulet.lit/balcony'
    id='le71fa63'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <success/>
    </reason>
  </jingle>
</iq>
```

The initiating client acknowledges receipt of the session-terminate message (not shown here) and the session is ended.
We now ”fill in the blanks” for the <description/> and <transport/> elements with a more complex example: a voice chat session, where the application type is a Jingle RTP session (with several different codec possibilities) and the transport method is ICE-UDP.

Listing 4: Initiator sends session-initiate

```
<iq from='romeo@montague.lit/orchard'
    id='ph37a419'
    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 message, the responder determines whether it can proceed with the negotiation. If there is no error, the responder acknowledges the session initiation request.

Listing 5: Responder acknowledges session-initiate

```
<iq from='juliet@capulet.lit/balcony'
    id='ph37a419'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

When the responding user affirms that she would like to proceed with the session, the responding client sends a session-accept message to the initiator (including in this example the subset of offered codecs that the responding client supports and one or more transport candidates generated by the responder).

Listing 6: Responder definitively accepts the session

```
<iq from='juliet@capulet.lit/balcony'
    id='yd71f495'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          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>
```

And the initiating client acknowledges session acceptance:

Listing 7: Initiator acknowledges session acceptance

```
<iq from='romeo@montague.lit/orchard'
    id='yd71f495'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

Once the parties finish the transport negotiation, they would then exchange media using any of the acceptable codecs.
Eventually, one of the parties (here the responder) will terminate the session.

Listing 8: Responder terminates the session

```
<iq from='juliet@capulet.lit/balcony'
    id='vua614d9'
    to='romeo@montague.lit/orchard'
    type='set'>
```

```
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <success/>
      <text>Sorry, gotta go!</text>
    </reason>
  </jingle>
</iq>
```

The other party then acknowledges termination of the session:

Listing 9: Initiator acknowledges termination

```
<iq from='romeo@montague.lit/orchard'
    id='vua614d9'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

3 Requirements

The protocol defined herein is designed to meet the following requirements:

1. Make it possible to manage a wide variety of peer-to-peer sessions (including but not limited to voice and video) within XMPP.
2. When a peer-to-peer connection cannot be negotiated, make it possible to fall back to relayed communications.
3. Clearly separate the signalling channel (XMPP) from the data channel.
4. Clearly separate the application format (e.g., RTP audio) from the transport method (e.g., UDP).
5. Make it possible to add, modify, and remove both application types and transport methods in an existing session.
6. Make it relatively easy to implement support for the protocol in standard Jabber/XMPP clients.
7. Where communication with non-XMPP entities is needed, push as much complexity as possible onto server-side gateways between the XMPP network and the non-XMPP network.

This document defines the signalling protocol only. Additional documents specify the following:

- Various application formats (audio, video, etc.) and, where possible, mapping of those types to the Session Description Protocol (SDP; see RFC 4566 [18]); examples include Jingle RTP Sessions and Jingle File Transfer.

- Various transport methods; examples include Jingle ICE-UDP Transport Method, Jingle Raw UDP Transport Method, Jingle In-Band Bytestreams Transport Method, and Jingle SOCKS5 Bytestreams Transport Method.

- Various methods of securing the transport before using it to send application data; the only method defined so far is Transport Layer Security as described in Jingle XTLS [19].

- Procedures for mapping the Jingle signalling protocol to existing signalling standards such as the IETF's Session Initiation Protocol (SIP) and the ITU's H.323 protocol (see H.323 [20]); see for example draft-ietf-stox-media [21].

4 Terminology

4.1 Glossary

Application Format The data format of the content type being established, which formally declares one purpose of the session (e.g., "audio" or "video"). This is the 'what' of the session (i.e., the bits to be transferred), such as the acceptable codecs when establishing a voice conversation. In Jingle XML syntax the application format is the namespace of the <description/> element.

Component A numbered stream of data that needs to be transmitted between the endpoints for a given content type in the context of a given session. It is up to the transport to negotiate the details of each component. Depending on the content type, multiple components might be needed (e.g., one to transmit an RTP stream and another to transmit RTCP timing information).

Content Type A pair formed by the combination of one application format and one transport method.

Session One or more content types negotiated between two entities. It is delimited in time by a session-initiate action and a session-terminate action. During the lifetime of a session,

[18] RFC 4566: SDP: Session Description Protocol <http://tools.ietf.org/html/rfc4566>.
[19] Extensible Messaging and Presence Protocol (XMPP) End-to-End Encryption Using Transport Layer Security ("XTLS") <http://tools.ietf.org/html/draft-meyer-xmpp-e2e-encryption>.
[20] ITU Recommendation H.323: Packet-based Multimedia Communications Systems (September 1999).
[21] Interworking between the Session Initiation Protocol (SIP) and the Extensible Messaging and Presence Protocol (XMPP): Media Sessions <http://tools.ietf.org/html/draft-ietf-stox-media> (work in progress).

content types can be added or removed. A session consists of at least one content type at a time.

Transport Method The method for establishing data stream(s) between entities. Possible transports might include ICE-UDP, ICE-TCP, Raw UDP, In-Band Bytestreams, SOCKS5 Bytestreams, etc. This is the 'how' of the session. In Jingle XML syntax this is the namespace of the <transport/> element. The transport method defines how to transfer bits from one host to another. Each transport method MUST specify whether it is "datagram" or "streaming" as described in the Transport Types section of this document.

4.2 Conventions

In diagrams, the following conventions are used:

- Single-dashed lines (---) represent Jingle stanzas that are sent via the XMPP signalling channel.
- Double-dashed lines (===) represent media packets that are sent via the data channel, which typically is not an XMPP channel (although the Jingle In-Band Bytestreams Transport Method is an exception) but instead is a direct or mediated channel between the endpoints.

5 Concepts and Approach

Jingle consists of three parts, each with its own syntax and semantics:

1. Overall session management
2. Application types (the "what")
3. Transport methods (the "how")

This document defines the semantics and syntax for overall session management. It also provides pluggable "slots" for application formats and transport methods, which are specified in separate documents.
At the most basic level, the process for initial negotiation of a Jingle session is as follows:

1. One user (the "initiator") sends to another user (the "responder") a session-initiate message containing at least one content definition, each of which defines one application type, one transport method, and optionally one security precondition.
2. If the responder wishes to proceed, it sends a session-accept message to the initiator, optionally including one or more transport candidates (depending on the transport method specified in the session-initiate message).

3. The parties attempt to establish connectivity over the offered transport method as defined in the relevant specification, which might involve the exchange of transport-info messages for additional transport candidates; if connectivity cannot be established then the parties might attempt to fall back to another transport method using the transport-replace and transport-accept messages.

4. Optionally, the parties attempt to establish security for the transport method before using it to exchange application data.

5. Optionally, either party can add or remove content definitions, or change the direction of the media flow, using the content-add, content-remove, and content-modify messages.

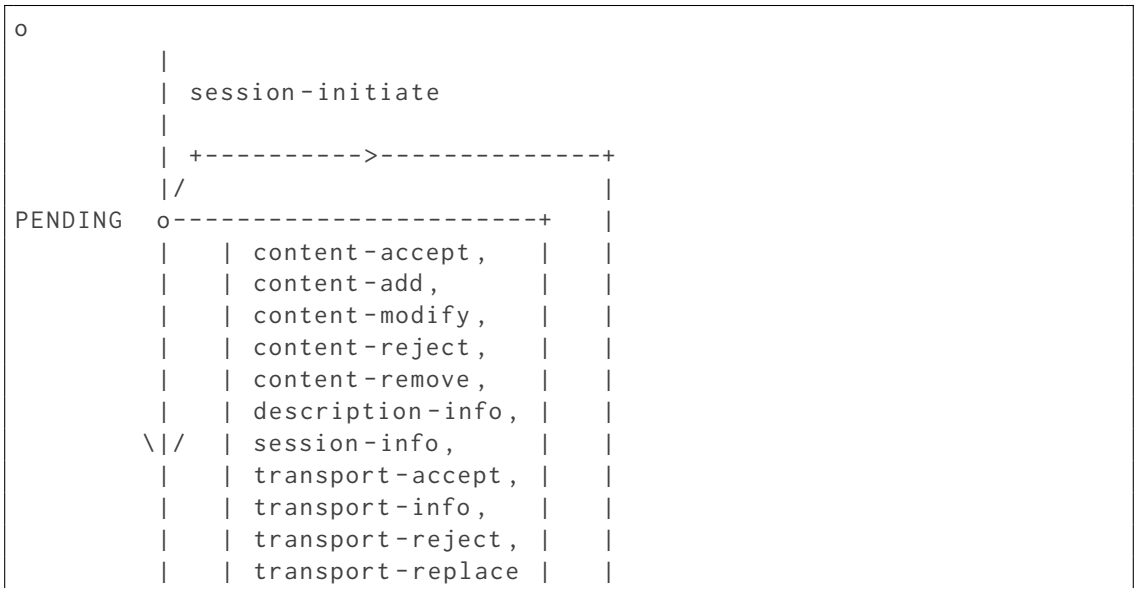
6. Optionally, either party can send session-info messages (e.g., to inform the other party that its device is ringing).

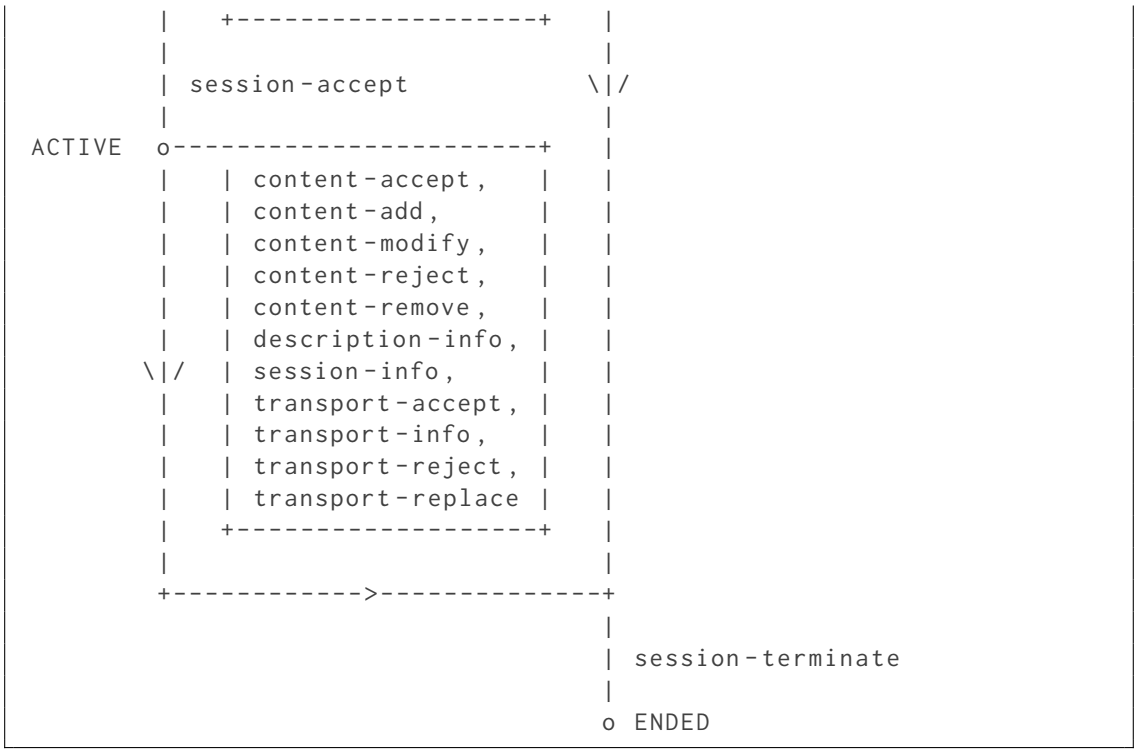
7. As soon as the initiator and responder determine that data can flow over the negotiated transport (potentially only after a security precondition has been met), they start sending application data over the transport.

Even after application data is being exchanged, the parties can adjust the session definition by sending additional Jingle messages, such as content-modify, content-remove, content-add, description-info, security-info, session-info, and transport-replace.

5.1 Overall Session Management

The state machine for overall session management (i.e., the state per Session ID) is as follows:





As shown, there are three overall session states:

1. PENDING
2. ACTIVE
3. ENDED

Note: While it is allowed to send all actions while in the PENDING state, typically the responder will send a session-accept message as quickly as possible in order to expedite the transport negotiation; see the Security Considerations section of this document regarding information exposure when the responder sends transport candidates to the initiator.
The actions related to management of the overall Jingle session are as follows (detailed definitions are provided in the Action Attribute section of this document).

content-accept Accept a content-add action received from another party.

content-add Add one or more new content definitions to the session.

content-modify Change the directionality of media sending.

content-reject Reject a content-add action received from another party.

content-remove Remove one or more content definitions from the session.

description-info Exchange information about parameters for an application type.

security-info Exchange information about security preconditions.

session-accept Definitively accept a session negotiation.

session-info Send session-level information, such as a ping or a ringing message.

session-initiate Request negotiation of a new Jingle session.

session-terminate End an existing session.

transport-accept Accept a transport-replace action received from another party.

transport-info Exchange transport candidates.

transport-reject Reject a transport-replace action received from another party.

transport-replace Redefine a transport method or replace it with a different method.

6 Session Flow

This section defines the high-level flow of a Jingle session. More detailed descriptions are provided in the specifications for Jingle application formats and transport methods.

6.1 Resource Determination

In order to initiate a Jingle session, the initiator needs to determine which of the responder's XMPP resources is best for the desired application format. Methods for doing so are out of scope for this specification. However, see the Determining Support section of this document for relevant information.

6.2 Initiation

Once the initiator has discovered which of the responder's XMPP resources is ideal for the desired application format, it sends a session initiation request to the responder. This request is an IQ-set containing a <jingle/> element qualified by the 'urn:xmpp:jingle:1' namespace (see Namespace Versioning regarding the possibility of incrementing the version number), where the value of the 'action' attribute is "session-initiate" and where the <jingle/> element contains one or more <content/> elements. Each <content/> element defines a content type to be transferred during the session, and each <content/> element in turn contains one <description/> child element that specifies a desired application format and one <transport/> child element that specifies a potential transport method, as well as (optionally) one <security/> element that specifies a security precondition that needs to be met before the parties can exchange application data over the negotiated transport.

Listing 10: Initiator sends session-initiate

```
<iq from='romeo@montague.lit/orchard'
    id='xs51r0k4'
    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>
```

Application types ought not to be mixed beyond necessity within a single session. Therefore the session initiation request (along with subsequent additions) will include only content-types that can be grouped together into a coherent session within a given Jingle application. For example, two parties might start an audio call but then add a video aspect to that call. If one of the parties decides to send a file to the other party as a result of discussion over the audio/video session or a text chat conversation, conceptually that is probably a separate session (unless file exchange or screen sharing or some other application type is an integral part of a broader collaboration experience and needs to be calibrated with the audio/video session).
Note: The syntax and semantics of the <description/>, <transport/>, and <security/> elements are out of scope for this document, since they are defined in related specifications. The syntax and semantics of the <jingle/> and <content/> elements are specified in this document under Formal Definition.

6.3 Responder Response

6.3.1 Acknowledgement

Unless one of the following errors occurs, the responder MUST acknowledge receipt of the initiation request.

Listing 11: Responder acknowledges session-initiate

```
<iq from='juliet@capulet.lit/balcony'
    id='xs51r0k4'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

However, after acknowledging the session initiation request, the responder might subsequently determine that it cannot proceed with negotiation of the session (e.g., because it does not support any of the offered application formats or transport methods, because a human user is busy or unable to accept the session, because a human user wishes to formally decline the session, etc.). In these cases, the responder SHOULD immediately acknowledge the session initiation request but then terminate the session with an appropriate reason as described in the Termination section of this document.

6.3.2 Errors

There are several reasons why the responder might immediately return an error instead of acknowledging receipt of the initiation request:

- The initiator is unknown to the responder and the responder does not communicate with unknown entities.

- The responder does not support Jingle.
- The responder wishes to redirect the request to another address.
- The responder does not have sufficient resources to participate in a session.
- The initiation request was malformed.

If the initiator is unknown to the responder (e.g., via presence subscription as defined in RFC 3921 [22]) and the responder has a policy of not communicating via Jingle with unknown entities, it MUST return a <service-unavailable/> error.

Listing 12: Initiator is unknown to responder

```
<iq from='juliet@capulet.lit/balcony'
    id='xs51r0k4'
    to='romeo@montague.lit/orchard'
    type='error'>
  <error type='cancel'>
    <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the responder does not support Jingle, it MUST return a <service-unavailable/> error.

Listing 13: Responder does not support Jingle

```
<iq from='juliet@capulet.lit/balcony'
    id='xs51r0k4'
    to='romeo@montague.lit/orchard'
    type='error'>
  <error type='cancel'>
    <service-unavailable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the responder wishes to redirect the request to another address, it MUST return a <redirect/> error.

Listing 14: Responder redirection

```
<iq from='juliet@capulet.lit/balcony'
    id='xs51r0k4'
    to='romeo@montague.lit/orchard'
    type='error'>
  <error type='modify'>
```

[22] RFC 3921: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc3921>.

```
    <redirect xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
      xmpp:voicemail@capulet.lit
    </redirect>
  </error>
</iq>
```

If the responder does not have sufficient resources to participate in a session, it MUST return a <resource-constraint/> error.

Listing 15: Responder has insufficent resources

```
<iq from='juliet@capulet.lit/balcony'
    id='xs51r0k4'
    to='romeo@montague.lit/orchard'
    type='error'>
  <error type='wait'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the initiation request was malformed, the responder MUST return a <bad-request/> error.

Listing 16: Initiation request malformed

```
<iq from='juliet@capulet.lit/balcony'
    id='xs51r0k4'
    to='romeo@montague.lit/orchard'
    type='error'>
  <error type='cancel'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

6.4 Negotiation

Although in general it is preferable for the responder to send a session-accept message as soon as possible, some forms of negotiation might be necessary before the parties can agree on an acceptable set of application formats and transport methods. There are many potential parameter combinations, as defined in the relevant specifications for various application formats and transport methods.
The allowable negotiations (e.g., content-level and transport-level negotiations) include:

- Exchanging particular transport candidates via the transport-info action.
- Modifying the communication direction for a content type via the content-modify action.

- Changing the definition of a content type via the transport-replace action (typically to fall back to a more suitable transport).
- Adding a content type via the content-add action.
- Removing a content type via the content-remove action.

These forms of negotiation can also occur after the session has been accepted.

6.5 Acceptance

As soon as possible after receiving the session-initiate message, the responder informs the initiator that she wishes to proceed with the session by sending a session-accept message.

Listing 17: Responder accepts the session

```
<iq from='juliet@capulet.lit/balcony'
    id='jd82f517'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          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>
```

Note: After receiving and acknowledging the "session-initiate" action received from the initiator, the responding client SHOULD present an interface element that enables a human user to explicitly agree to proceeding with the session (e.g., an "Accept Incoming Call?"

pop-up window including ”Yes” and ”No” buttons). However, the responding client SHOULD NOT return a ”session-accept” action to the initiator until the responder has explicitly agreed to proceed with the session (unless the initiator is on a list of entities whose sessions are automatically accepted).
The initiator then acknowledges the responder's definitive acceptance.

Listing 18: Initiator acknowledges session acceptance

```
<iq from='romeo@montague.lit/orchard'
    id='jd82f517'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

The session is now in the ACTIVE state. However, this does not necessarily mean that the parties can exchange application data yet, because further negotiation might be necessary (e.g., to fall back from the offered transport method to a suitable alternative).

6.6 Modifying an Active Session

Once a session is in the ACTIVE state, it might be modified via a content-add, content-modify, content-remove, or transport-info message. Examples of such modifications are shown in the specifications for various application formats and transport methods.

6.7 Termination

In order to gracefully end the session (which can be done at any point after acknowledging receipt of the initiation request, including immediately thereafter in order to decline the request), either the responder or the initiator MUST send a session-terminate message to the other party.
The party that terminates the session SHOULD include a <reason/> element that specifies why the session is being terminated. Examples follow.
Probably the primary reason for terminating a session is that the session has ended successfully (e.g., because a file has been sent or a voice call has completed); in this case, the recommended condition is <success/>.

Listing 19: Terminating the session (success)

```
<iq from='juliet@capulet.lit/balcony'
    id='bv81gs75'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
```

```
    <reason>
      <success/>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party is busy; in this case, the recommended condition is <busy/>.

Listing 20: Terminating the session (busy)

```
<iq from='juliet@capulet.lit/balcony'
    id='hr81fs63'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <busy/>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party wishes to formally decline the session; in this case, the recommended condition is <decline/>.

Listing 21: Terminating the session (session formally declined)

```
<iq from='juliet@capulet.lit/balcony'
    id='ky47g295'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <decline/>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party already has an existing session with the other party and wishes to use that session rather than initiate a new session; in this case, the recommended condition is <alternative-session/> and the terminating party SHOULD include the session ID of the alternative session in the <sid/> element.

Listing 22: Terminating the session (existing session)

```
<iq from='juliet@capulet.lit/balcony'
    id='ay3r2b86'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <alternative-session>
        <sid>b84tkkwlmb48kgfb</sid>
      </alternative-session>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party does not support any of the offered transport methods; in this case, the recommended condition is <unsupported-transports/>.

Listing 23: Terminating the session (no offered transport method supported)

```
<iq from='juliet@capulet.lit/balcony'
    id='h82bs51g'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <unsupported-transports/>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party has determined that transport setup has failed in an unrecoverable fashion (e.g., all transport methods have been exhausted even after fallback and the last method attempted has failed); in this case, the recommended condition is <failed-transport/>.

Listing 24: Terminating the session (transport negotiation failed)

```
<iq from='juliet@capulet.lit/balcony'
    id='pe81ga88'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
```

```
          sid='a73sjjvkla37jfea'>
    <reason>
      <failed-transport/>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party does not support any of the offered application types; in this case, the recommended condition is <unsupported-applications/>.

Listing 25: Terminating the session (no offered application type supported)

```
<iq from='juliet@capulet.lit/balcony'
    id='yd62vd67'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <unsupported-applications/>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party has determined that setup of the application type has failed in an unrecoverable fashion (e.g., the client cannot initialize audio processing for a voice call); in this case, the recommended condition is <failed-application/>.

Listing 26: Terminating the session (application setup failed)

```
<iq from='juliet@capulet.lit/balcony'
    id='kd82vs71'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <failed-application/>
    </reason>
  </jingle>
</iq>
```

Another reason for terminating the session is that the terminating party supports the offered application type but does not support the offered or negotiated parameters (e.g., in a voice

call none of the payload types); in this case, the recommended condition is <incompatible-parameters/>.

Listing 27: Terminating the session (incompatible parameters)

```
<iq from='juliet@capulet.lit/balcony'
    id='hb3m59s7'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <incompatible-parameters/>
    </reason>
  </jingle>
</iq>
```

Note: Other reasons for terminating the session might apply, and the foregoing list is not exhaustive.
Upon receiving session-terminate message, the other party MUST then acknowledge termination of the session:

Listing 28: Acknowledging termination

```
<iq from='romeo@montague.lit/orchard'
    id='h82bs51g'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

Note: As soon as an entity sends a session-terminate action, it MUST consider the session to be in the ENDED state (even before receiving acknowledgement from the other party). If the terminating entity receives additional Jingle-related IQ-sets from the other party after sending the session-terminate action, it MUST reply with an <unknown-session/> error.

Listing 29: Unknown-session error

```
<iq from='romeo@montague.lit/orchard'
    id='ur71vs62'
    to='juliet@capulet.lit/balcony'
    type='error'>
  <error type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <unknown-session xmlns='urn:xmpp:jingle:errors:1'/>
  </error>
</iq>
```

Not all Jingle sessions end gracefully. When the parties to a Jingle session also exchange XMPP presence information, receipt of <presence type='unavailable'/> from the other party SHOULD be considered a session-ending event that justifies proactively sending a session-terminate message to the seemingly unavailable party -- if, that is, no other communication has been received within 5 or 10 seconds from the seemingly unavailable party in the form of XMPP signalling traffic, connectivity checks, or continued media transfer.

6.8 Informational Messages

At any point after initiation of a Jingle session, either entity MAY send an informational message to the other party, for example to inform the other party that a device is ringing.

Listing 30: Responder sends ringing message

```
<iq from='juliet@capulet.lit/balcony'
    id='hq7rg186'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          sid='a73sjjvkla37jfea'>
    <ringing xmlns='urn:xmpp:jingle:apps:rtp:1:info'/>
  </jingle>
</iq>
```

An informational message MUST be an IQ-set containing a <jingle/> element whose 'action' attribute is set to a value of "session-info", "description-info", or "transport-info"; the <jingle/> element SHOULD further contain a payload child element (specific to the application format or transport method) that specifies the information being communicated. If the party that receives an informational message does not understand the payload, it MUST return a <feature-not-implemented/> error with a Jingle-specific error condition of <unsupported-info/>.

Listing 31: Responder returns unsupported-info error

```
<iq from='romeo@montague.lit/orchard'
    id='hq7rg186'
    to='juliet@capulet.lit/balcony'
    type='error'>
  <error type='modify'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-
        stanzas'/>
    <unsupported-info xmlns='urn:xmpp:jingle:errors:1'/>
  </error>
</iq>
```

However, the <jingle/> element associated with a session-info message MAY be empty. If either party receives an empty session-info message for an active session, it MUST send an empty IQ result; this usage functions as a ”ping” to determine session vitality via the XMPP signalling channel.

Listing 32: Responder sends session ping

```
<iq from='juliet@capulet.lit/balcony'
    id='ug37vb25'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          sid='a73sjjvkla37jfea'/>
</iq>
```

Listing 33: Initiator returns IQ-result

```
<iq from='romeo@montague.lit/orchard'
    id='ug37vb25'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

7 Formal Definition

7.1 Jingle Element

The <jingle/> element MAY be empty or contain one or more <content/> elements (for which see Content Element).
The attributes of the <jingle/> element are as follows.

Attribute	Definition	Inclusion
action	A Jingle action as described under Action Attribute.	REQUIRED

Attribute	Definition	Inclusion
initiator*	The full JID of the entity that has initiated the session flow. When the Jingle action is ”session-initiate”, the <jingle/> element SHOULD possess an ’initiator’ attribute that explicitly specifies the full JID of the initiating entity; for all other actions, the <jingle/> element SHOULD NOT possess an ’initiator’ attribute and the recipient of the message SHOULD ignore the value if provided. The value of the ’initiator’ attribute MAY be different from the ’from’ address on the IQ-set of the session-initiate message (e.g., to handle certain interactions involving call managers, soft switches, and media relays). This usage shall be defined in other specifications, for example, in Jingle Session Transfer (XEP-0251) XEP-0251: Jingle Session Transfer <https://xmpp.org/extensions/xep-0251.html>.. However, in all cases if the ’initiator’ and ’from’ values differ then the responder MUST NOT interact with the ’initiator’ JID unless it trusts the ’initiator’ JID or trusts that the ’from’ JID is allowed to authorize the ’initiator’ JID to act on the ’from’ JID’s behalf. In the absence of explicit rules for handling this case, the responder SHOULD simply ignore the ’initiator’ attribute and treat the ’from’ JID as the initiating entity. After sending acknowledgement of the session-initiate message, the responder MUST send all future commmunications about the Jingle session to the initiator (whether the initiator is considered the ’from’ JID or the ’initiator’ JID).	RECOMMENDED for session-initiate, NOT RECOMMENDED otherwise

Attribute	Definition	Inclusion
responder*	The full JID of the entity that has replied to the initiation, which can be different from the 'to' address on the IQ-set. When the Jingle action is "session-accept", the <jingle/> element SHOULD possess a 'responder' attribute that explicitly specifies the full JID of the responding entity; for all other actions, the <jingle/> element SHOULD NOT possess a 'responder' attribute and the recipient of the message SHOULD ignore the value if provided. The value of the 'responder' attribute MAY be different from the 'from' address on the IQ-set of the session-accept message, where the logic for handling any difference between the 'responder' JID and the 'from' JID follows the same logic as for session-initiate messages (see above). After sending acknowledgement of the session-accept message, the initiator MUST send all future commmunications about this Jingle session to the responder (whether the responder is considered the 'from' JID or the 'responder' JID).	RECOMMENDED for session-accept, NOT RECOMMENDED otherwise

Attribute	Definition	Inclusion
sid	A random session identifier generated by the initiator, which effectively maps to the local-part of a SIP "Call-ID" parameter; this SHOULD match the XML Nmtoken production See <http://www.w3.org/TR/2000/WD-xml-2e-20000814#NT-Nmtoken> so that XML character escaping is not needed for characters such as '&'. In some situations the Jingle session identifier might have security implications. See RFC 4086 RFC 4086: Randomness Requirements for Security <http://tools.ietf.org/html/rfc4086>. regarding requirements for randomness.	REQUIRED

7.2 Action Attribute

The value of the 'action' attribute MUST be one of the following. If an entity receives a value not defined here, it MUST ignore the attribute and MUST return a <bad-request/> error to the sender. There is no default value for the 'action' attribute.

7.2.1 content-accept

The **content-accept** action is used to accept a content-add action received from another party.

7.2.2 content-add

The **content-add** action is used to add one or more new content definitions to the session. The sender MUST specify only the added content definition(s), not the added content definition(s) plus the existing content definition(s). Therefore it is the responsibility of the recipient to maintain a local copy of the current content definition(s). If the recipient wishes to include the new content definition in the session, it MUST send a content-accept action to the other party; if not, it MUST send a content-reject action to the other party.

7.2.3 content-modify

The **content-modify** action is used to change the direction of an existing content definition through modification of the 'senders' attribute. If the recipient deems the directionality of a content-modify action to be unacceptable, it MAY reply with a contrary content-modify action, terminate the session, or simply refuse to send or accept application data in the new direction. In any case, the recipient MUST NOT send a content-accept action in response to the content-modify.

7.2.4 content-reject

The **content-reject** action is used to reject a content-add action received from another party. If the content-reject results in zero content definitions for the session, the entity that receives the content-reject SHOULD send a session-terminate action to the other party (since a session with no content definitions is void).

7.2.5 content-remove

The **content-remove** action is used to remove one or more content definitions from the session. The sender MUST specify only the removed content definition(s), not the removed content definition(s) plus the remaining content definition(s). Therefore it is the responsibility of the recipient to maintain a local copy of the current content definition(s). Upon receiving a content-remove from the other party, the recipient MUST NOT send a content-accept and MUST NOT continue to negotiate the transport method or send application data related to that content definition.
If the content-remove results in zero content definitions for the session, the entity that receives the content-remove SHOULD send a session-terminate action to the other party (since a session with no content definitions is void).

7.2.6 description-info

The **description-info** action is used to send informational hints about parameters related to the application type, such as the suggested height and width of a video display area or suggested configuration for an audio stream.

7.2.7 security-info

The **security-info** action is used to send information related to establishment or maintenance of security preconditions.

7.2.8 session-accept

The **session-accept** action is used to definitively accept a session negotiation (implicitly this action also serves as a content-accept). A session-accept action indicates a willingness to proceed with the session (which might necessitate further negotiation before media can be exchanged). The session-accept action indicates acceptance *only* of the content definition(s) whose disposition type is "session" (the default value of the <content/> element's 'disposition' attribute), not any content definition(s) whose disposition type is something other than "session" (e.g., "early-session" for early media).
In the session-accept stanza, the <jingle/> element MUST contain one or more <content/> elements, each of which MUST contain one <description/> element and one <transport/> element.

7.2.9 session-info

The **session-info** action is used to send session-level information, such as a session ping or (for Jingle RTP sessions) a ringing message.

7.2.10 session-initiate

The **session-initiate** action is used to request negotiation of a new Jingle session. When sending a session-initiate with one <content/> element, the value of the <content/> element's 'disposition' attribute MUST be "session" (if there are multiple <content/> elements then at least one MUST have a disposition of "session"); if this rule is violated, the responder MUST return a <bad-request/> error to the initiator.

7.2.11 session-terminate

The **session-terminate** action is used to end an existing session.

7.2.12 transport-accept

The **transport-accept** action is used to accept a transport-replace action received from another party.

7.2.13 transport-info

The **transport-info** action is used to exchange transport candidates; it is mainly used in Jingle ICE-UDP but might be used in other transport specifications.

7.2.14 transport-reject

The **transport-reject** action is used to reject a transport-replace action received from another party.

7.2.15 transport-replace

The **transport-replace** action is used to redefine a transport method, typically for fallback to a different method (e.g., changing from ICE-UDP to Raw UDP for a datagram transport, or changing from SOCKS5 Bytestreams (XEP-0065) [23] to In-Band Bytestreams (XEP-0047) [24] for a streaming transport). If the recipient wishes to use the new transport definition, it MUST send a transport-accept action to the other party; if not, it MUST send a transport-reject action to the other party.

7.2.16 Tie Breaking Related to Jingle Actions

It is possible that the same Jingle action can be sent at the same time by both parties. There are two possible scenarios:

No existing session If there is no existing session and both parties simultaneously send a Jingle session-initiate message with a content-type that is functionally equivalent (e.g., each message requests initiation of a voice call), the action with the lower of the two session IDs MUST overrule the other action, where by "lower" is meant the session ID that is sorted first using "i;octet" collation as specified in Section 9.3 of RFC 4790 RFC 4790: Internet Application Protocol Collation Registry <http://tools.ietf.org/html/rfc4790>. (in the unlikely event that the random session IDs are the same, the action sent by the lower of the JabberIDs MUST overrule the other action). The party that receives the session-initiate action with the lower of the two session IDs MUST acknowledge the action or return an error condition that would normally be returned when receiving a session-initiate message, and the party that receives the session-initiate action with the higher of the two session IDs MUST return a <conflict/> error to the other party, which SHOULD include a Jingle-specific condition of <tie-break/>.

Existing session In the context of an existing session, the action sent by the initiator MUST overrule the action sent by the responder; i.e., both parties MUST accept the action sent by the initiator and the initiator MUST return a <conflict/> error to the responder for the duplicate action, which SHOULD include a Jingle-specific condition of <tie-break/>.

In both scenarios, the error to be returned is <conflict/>, as shown in the following example.

[23] XEP-0065: SOCKS5 Bytestreams <https://xmpp.org/extensions/xep-0065.html>.
[24] XEP-0047: In-Band Bytestreams <https://xmpp.org/extensions/xep-0047.html>.

Listing 34: Initiator returns conflict error on tie-break

```
<iq from='romeo@montague.lit/orchard'
    id='hb2f164w'
    to='juliet@capulet.lit/balcony'
    type='error'>
  <error type='cancel'>
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <tie-break xmlns='urn:xmpp:jingle:errors:1'/>
  </error>
</iq>
```

7.3 Content Element

The attributes of the <content/> element are as follows.

Attribute	Definition	Inclusion
creator	Which party originally generated the content type (used to prevent race conditions regarding modifications); the defined values are "initiator" and "responder" (where the default is "initiator"). The value of the 'creator' attribute for a given content type MUST always match the party that originally generated the content type, even for Jingle actions that are sent by the other party in relation to that content type (e.g., subsequent content-modify or transport-info messages). The combination of the 'creator' attribute and the 'name' attribute is unique among both parties to a Jingle session.	REQUIRED

Attribute	Definition	Inclusion
disposition	How the content definition is to be interpreted by the recipient. The meaning of this attribute matches the ”Content-Disposition” header as defined in RFC 2183 RFC 2183: Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field <http://tools.ietf.org/html/rfc2183>. and applied to SIP by RFC 3261 RFC 3261: Session Initiation Protocol (SIP) <http://tools.ietf.org/html/rfc3261>.. The value of this attribute SHOULD be one of the values registered in the IANA Mail Content Disposition Values and Parameters Registry IANA registry of Mail Content Disposition Values and Parameters <http://www.iana.org/assignments/mail-cont-disp>.. The default value of this attribute is ”session”.	OPTIONAL
name	A unique name or identifier for the content type according to the creator, which MAY have meaning to a human user in order to differentiate this content type from other content types (e.g., two content types containing video media could differentiate between ”roompan” and ”slides”). If there are two content types with the same value for the ’name’ attribute, they shall understood as alternative definitions for the same purpose (e.g., a legacy method and a standards-based method for establishing a voice call), typically to smooth the transition from an older technology to Jingle.	REQUIRED

Attribute	Definition	Inclusion
senders	Which parties in the session will be generating content (i.e., the direction in which a Jingle session is active); the allowable values are ”both”, ”initiator”, ”none”, and ”responder” (where the default is ”both”). Note that the defined values of the ’senders’ attribute in Jingle correspond to the SDP attributes of ”sendrecv”, ”sendonly”, ”inactive”, and ”recvonly” defined in RFC 4566 RFC 4566: SDP: Session Description Protocol <http://tools.ietf.org/html/rfc4566>. and used in the offer-answer model RFC 3264 RFC 3264: An Offer/Answer Model with the Session Description Protocol (SDP) <http://tools.ietf.org/html/rfc3264>..	OPTIONAL except when sending content-modify, in which case it is REQUIRED.

7.4 Reason Element

The structure of the <reason/> element is as follows.

- The <reason/> element MUST contain an element that provides machine-readable information about the condition that prompted the action.
- The <reason/> element MAY contain a <text/> element that provides human-readable information about the reason for the action.
- The <reason/> element MAY contain an element qualified by some other namespace that provides more detailed machine-readable information about the reason for the action.

A <reason/> element can be included with any Jingle action, and is not limited to session termination events.
The defined conditions are described in the following table.

Element	Description
<alternative-session/>	The party prefers to use an existing session with the peer rather than initiate a new session; the Jingle session ID of the alternative session SHOULD be provided as the XML character data of the <sid/> child.
<busy/>	The party is busy and cannot accept a session.
<cancel/>	The initiator wishes to formally cancel the session initiation request.
<connectivity-error/>	The action is related to connectivity problems.
<decline/>	The party wishes to formally decline the session.
<expired/>	The session length has exceeded a pre-defined time limit (e.g., a meeting hosted at a conference service).
<failed-application/>	The party has been unable to initialize processing related to the application type.
<failed-transport/>	The party has been unable to establish connectivity for the transport method.
<general-error/>	The action is related to a non-specific application error.
<gone/>	The entity is going offline or is no longer available.
<incompatible-parameters/>	The party supports the offered application type but does not support the offered or negotiated parameters.
<media-error/>	The action is related to media processing problems.
<security-error/>	The action is related to a violation of local security policies.
<success/>	The action is generated during the normal course of state management and does not reflect any error.
<timeout/>	A request has not been answered so the sender is timing out the request.
<unsupported-applications/>	The party supports none of the offered application types.
<unsupported-transports/>	The party supports none of the offered transport methods.

8 Transport Types

Jingle defines two types of transport methods.

8.1 Datagram

A datagram transport has one or more components with which to exchange packets with UDP-like behavior. Packets might be of arbitrary length, might be received out of order, and might not be received at all (i.e., the transport is lossy). Each component is assigned a string identifier and has a maximum packet length.
Applications compatible with datagram transports MUST specify how many components are necessary, what identifier to assign each component, and how each component will be used.

8.2 Streaming

A streaming transport has one or more components with which to exchange bidirectional bytestreams with TCP-like behavior. Bytes are received reliably and in order, and applications MUST NOT rely on a stream being chunked in any specific way. Each component is assigned a string identifier and has a maximum packet length.
Applications compatible with stream transports MUST specify how many components are necessary, what identifier to assign each component, and what data shall be exchanged over the transport.

9 Security Preconditions

The initiator MAY include a <security/> element in its offer to signal that it wishes to enforce some security precondition on the session. A stub example follows.

Listing 35: Initiator sends session-initiate with security precondition (stub)

```
<iq from='romeo@montague.lit/orchard'
    id='tiw51bv9'
    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='this-is-a-stub'>
      <description xmlns='urn:xmpp:jingle:apps:stub:0'/>
      <transport xmlns='urn:xmpp:jingle:transports:stub:0'/>
      <security xmlns='urn:xmpp:jingle:security:stub:0'/>
    </content>
  </jingle>
</iq>
```

Currently the only security precondition that is envisioned will enforce the use of end-to-end encryption for the transport before application data can be exchanged. This document does not define any security preconditions, just as it does not define any application types or transport methods. See Jingle XTLS [25] for an in-progress description of a security precondition using Transport Layer Security (TLS).
In order to exchange information about the establishment or maintenance of a security precondition, either party might send a Jingle security-info message. For example, when attempting to negotiate the use of TLS the initiator might send hints about his supported TLS methods (e.g., X.509 certificates and Secure Remote Password) in his session-initiate message and the responder might also send hints about her supported methods (e.g., X.509 and SRP) in

[25] Extensible Messaging and Presence Protocol (XMPP) End-to-End Encryption Using Transport Layer Security ("XTLS") <http://tools.ietf.org/html/draft-meyer-xmpp-e2e-encryption>.

her session-accept message; however, it is possible that the initiator might be able to verify the responder's certificate and therefore needs to inform the responder (via a security-info message) that he can in the end support only the X.509 method for this negotiation. An example follows.

Listing 36: Initiator sends security-info message

```
<iq from='romeo@montague.lit/orchard'
    id='zyw6m167'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='security-info'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='xmlstream'>
      <security xmlns='urn:xmpp:jingle:security:xtls:0'>
        <method name='x509'/>
      </security>
    </content>
  </jingle>
</iq>
```

If one of the parties attempts to send information over the unsecured XMPP signalling channel that the other party expects to receive over the encrypted data channel, the receiving party SHOULD return a <not-acceptable/> error to the sender, including a <security-required/> element qualified by the 'urn:xmpp:jingle:errors:1' namespace. An example follows.

Listing 37: Initiator sends security-required error

```
<iq from='romeo@montague.lit/orchard'
    id='bsi381f5'
    to='juliet@capulet.lit/balcony'
    type='error'>
  <error type='cancel'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <security-required xmlns='urn:xmpp:jingle:errors:1'/>
  </error>
</iq>
```

10 Error Handling

The Jingle-specific error conditions are as follows. These condition elements are qualified by the 'urn:xmpp:jingle:errors:1' namespace (see Namespace Versioning regarding the possibility of incrementing the version number).

Jingle Condition	XMPP Condition	Description
<out-of-order/>	<unexpected-request/>	The request cannot occur at this point in the state machine (e.g., session-initiate after session-accept).
<tie-break/>	<conflict/>	The request is rejected because it was sent while the initiator was awaiting a reply on a similar request.
<unknown-session/>	<item-not-found/>	The 'sid' attribute specifies a session that is unknown to the recipient (e.g., no longer live according to the recipient's state machine because the recipient previously terminated the session).
<unsupported-info/>	<feature-not-implemented/>	The recipient does not support the informational payload of a session-info action.

11 Determining Support

If an entity supports Jingle, it MUST advertise that fact by returning a feature of "urn:xmpp:jingle:1" (see Namespace Versioning regarding the possibility of incrementing the version number) in response to a Service Discovery (XEP-0030) [26] information request. The response MUST also include features for the application formats and transport methods supported by the responding entity, as described in the relevant specifications.

Listing 38: Service Discovery Information Request

```
<iq from='kingclaudius@shakespeare.lit/castle'
    id='ku6e51v3'
    to='laertes@shakespeare.lit/castle'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 39: Service Discovery Information Response

```
<iq from='laertes@shakespeare.lit/castle'
    id='ku6e51v3'
    to='kingclaudius@shakespeare.lit/castle'
```

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

```
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:jingle:1'/>
    <feature var='urn:xmpp:jingle:apps:rtp:1'/>
    <feature var='urn:xmpp:jingle:apps:rtp:audio'/>
    <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) [27]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

12 Conformance by Using Protocols

12.1 Application Formats

A document that specifies a Jingle application format (e.g., RTP sessions) MUST define:

1. How successful application format negotiation occurs.
2. A <description/> element and associated semantics for representing the application format.
3. If and how the application format can be mapped to the Session Description Protocol, including the appropriate SDP media type (see Section 8.2.1 of RFC 4566 [28]).
4. Whether the media data for the application format shall be sent over a streaming transport method or a datagram transport method (or, if both, which is preferred).
5. If the chosen transport handles "components", define how the components shall be identified and assigned.
6. Exactly how the media data is to be sent and received over a streaming or datagram transport.

12.2 Transport Methods

A document that specifies a Jingle transport method (e.g., raw UDP) MUST define:

1. How successful transport negotiation occurs.

[27] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[28] RFC 4566: SDP: Session Description Protocol <http://tools.ietf.org/html/rfc4566>.

2. A <transport/> element and associated semantics for representing the transport method.
3. Whether the transport is a streaming method or a datagram method.
4. If the transport supports multiple components.

12.3 Security Preconditions

A document that specifies a Jingle security precondition MUST define:

1. A <security/> element and associated semantics for representing the security precondition.
2. Whether the security precondition applies to streaming transport methods, datagram transport methods, or both.
3. When the precondition is met so that application data can be sent over the negotiated transport.

13 Security Considerations

13.1 Transport Security

It is strongly recommended to protect the transport method using an appropriate security precondition (e.g., Transport Layer Security). However, methods for doing so are out of scope for this specification.

13.2 Denial of Service

Jingle sessions can be resource-intensive. Therefore, it is possible to launch a denial-of-service attack against an entity by burdening it with too many Jingle sessions. Care MUST be taken to accept sessions only from known entities and only if the entity's device is able to process such sessions.

13.3 Communication Through Gateways

Jingle communications can be enabled through gateways to non-XMPP networks, whose security characteristics can be quite different from those of XMPP networks. (For example, on some SIP networks authentication is optional and "from" addresses can be easily forged.) Care MUST be taken in communicating through such gateways.

13.4 Information Exposure

Mere negotiation of a Jingle session can expose sensitive information about the parties (e.g., IP addresses, or even the full JID of the responder). Care MUST be taken in communicating such information, and end-to-end encryption SHOULD be used if the parties do not trust the intermediate servers or gateways.

13.5 Redirection

The 'initiator' and 'responder' attributes can be used to redirect a session from one JID to another JID (i.e., the 'initiator' or 'responder' attribute might not match the 'from' or 'to' attribute of the sender). An application SHOULD NOT accept the redirection unless the bare JIDs match (i.e., the session is being redirected from one authorized resource to another authorized resource associated with the same account).

14 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [29].

15 XMPP Registrar Considerations

15.1 Protocol Namespaces

This specification defines the following XML namespaces:

- urn:xmpp:jingle:1
- urn:xmpp:jingle:errors:1

The XMPP Registrar [30] includes the foregoing namespaces in its registry at <https://xmpp.org/registrar/namespaces.html>, as governed by XMPP Registrar Function (XEP-0053) [31].

[29] 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/>.

[30] 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/>.

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

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 Jingle Application Formats Registry

The XMPP Registrar maintains a registry of Jingle application formats at <https://xmpp.org/registrar/jingle-apps.html>. All application format registrations shall be defined in separate specifications (not in this document). Application types defined within the XEP series MUST be registered with the XMPP Registrar, resulting in protocol URNs of the form "urn:xmpp:jingle:app:name:X" (where "name" is the registered name of the application format and "X" is a non-negative integer).
In order to submit new values to this registry, the registrant shall define an XML fragment of the following form and either include it in the relevant XMPP Extension Protocol or send it to the email address <registrar@xmpp.org>:

```
<application>
  <name>The name of the application format.</name>
  <desc>A natural-language summary of the application format.</desc>
  <transport>
    Whether the media can be sent over a "streaming" transport,
    a "datagram" transport, or "both".
  </transport>
  <doc>The document in which the application format is specified.</doc
     >
</application>
```

15.4 Jingle Transport Methods Registry

The XMPP Registrar maintains a registry of Jingle transport methods at <https://xmpp.org/registrar/jingle-transports.html>. All transport method registrations shall be defined in separate specifications (not in this document). Transport methods defined within the XEP series MUST be registered with the XMPP Registrar, resulting in protocol URNs of the form "urn:xmpp:jingle:transport:name" (where "name" is the registered name of the transport method).
In order to submit new values to this registry, the registrant shall define an XML fragment of the following form and either include it in the relevant XMPP Extension Protocol or send it to the email address <registrar@xmpp.org>:

```
<transport>
```

```
  <name>The name of the transport method.</name>
  <desc>A natural-language summary of the transport method.</desc>
  <type>
    Whether the transport method can be "streaming", "datagram",
    or "both".
  </type>
  <doc>The document in which this transport method is specified.</doc>
</transport>
```

16 XML Schemas

16.1 Jingle

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

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

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0166: http://www.xmpp.org/extensions/xep-0166.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='jingle'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='content'
                    type='contentElementType'
                    minOccurs='0'
                    maxOccurs='unbounded'/>
        <xs:element name='reason'
                    type='reasonElementType'
                    minOccurs='0'
                    maxOccurs='1'/>
        <xs:any namespace='##other' minOccurs='0' maxOccurs='unbounded
           '/>
      </xs:sequence>
      <xs:attribute name='action' use='required'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='content-accept'/>
            <xs:enumeration value='content-add'/>
```

```
            <xs:enumeration value='content-modify'/>
            <xs:enumeration value='content-reject'/>
            <xs:enumeration value='content-remove'/>
            <xs:enumeration value='description-info'/>
            <xs:enumeration value='security-info'/>
            <xs:enumeration value='session-accept'/>
            <xs:enumeration value='session-info'/>
            <xs:enumeration value='session-initiate'/>
            <xs:enumeration value='session-terminate'/>
            <xs:enumeration value='transport-accept'/>
            <xs:enumeration value='transport-info'/>
            <xs:enumeration value='transport-reject'/>
            <xs:enumeration value='transport-replace'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name='initiator' type='xs:string' use='optional'/>
      <xs:attribute name='responder' type='xs:string' use='optional'/>
      <xs:attribute name='sid' type='xs:NMTOKEN' use='required'/>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='alternativeSessionElementType'>
    <xs:sequence>
      <xs:element name='sid'
                  minOccurs='0'
                  maxOccurs='1'
                  type='xs:NMTOKEN'/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='contentElementType'>
    <xs:sequence>
      <xs:any namespace='##other' minOccurs='0' maxOccurs='unbounded'/
          >
    </xs:sequence>
    <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='disposition'
                  use='optional'
                  type='xs:NCName'
                  default='session'/>
```

```
    <xs:attribute name='name'
                  use='required'
                  type='xs:string'/>
    <xs:attribute name='senders'
                  use='optional'
                  default='both'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='both'/>
          <xs:enumeration value='initiator'/>
          <xs:enumeration value='none'/>
          <xs:enumeration value='responder'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name='reasonElementType'>
    <xs:sequence>
      <xs:choice>
        <xs:element name='alternative-session'
                    type='alternativeSessionElementType'/>
        <xs:element name='busy' type='empty'/>
        <xs:element name='cancel' type='empty'/>
        <xs:element name='connectivity-error' type='empty'/>
        <xs:element name='decline' type='empty'/>
        <xs:element name='expired' type='empty'/>
        <xs:element name='failed-application' type='empty'/>
        <xs:element name='failed-transport' type='empty'/>
        <xs:element name='general-error' type='empty'/>
        <xs:element name='gone' type='empty'/>
        <xs:element name='incompatible-parameters' type='empty'/>
        <xs:element name='media-error' type='empty'/>
        <xs:element name='security-error' type='empty'/>
        <xs:element name='success' type='empty'/>
        <xs:element name='timeout' type='empty'/>
        <xs:element name='unsupported-applications' type='empty'/>
        <xs:element name='unsupported-transports' type='empty'/>
      </xs:choice>
      <xs:element name='text' type='xs:string' minOccurs='0' maxOccurs
         ='1'/>
      <xs:any namespace='##other' minOccurs='0' maxOccurs='1'/>
    </xs:sequence>
  </xs:complexType>

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

```
  </xs:simpleType>

</xs:schema>
```

16.2 Jingle Errors

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

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

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0166: http://www.xmpp.org/extensions/xep-0166.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='out-of-order' type='empty'/>
  <xs:element name='tie-break' type='empty'/>
  <xs:element name='unknown-session' type='empty'/>
  <xs:element name='unsupported-info' type='empty'/>

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

</xs:schema>
```

17 History

Until Jingle was developed, there existed no widely-adopted standard for initiating and managing peer-to-peer interactions between XMPP entities. Although several large service providers and Jabber client teams had written and implemented their own proprietary XMPP extensions for peer-to-peer signalling (usually only for voice), those technologies were not open and did not always take into account requirements to interoperate with SIP-based technologies. The only existing open protocol was A Transport for Initiating and Negotiating Sessions (XEP-0111) [32], which made it possible to initiate and manage peer-to-peer sessions,

[32] XEP-0111: A Transport for Initiating and Negotiating Sessions <https://xmpp.org/extensions/xep-0111.html>.

but which did not provide enough of the key signalling semantics to be easily implemented in Jabber/XMPP clients. [33]
The result was an unfortunate fragmentation within the XMPP community regarding signalling protocols. Essentially, there were two possible approaches to solving the problem:

1. Recommend that all client developers implement a dual-stack (XMPP + SIP) solution.
2. Define a full-featured protocol for XMPP signalling.

Implementation experience indicates that a dual-stack approach might not be feasible on all the computing platforms for which Jabber clients have been written, or even desirable on platforms where it is feasible. [34] Therefore, it seemed reasonable to define an XMPP signalling protocol that could provide the necessary session management semantics while also making it relatively straightforward to interoperate with existing Internet standards.
As a result of feedback received on A Transport for Initiating and Negotiating Sessions (XEP-0111) [35], the original authors of this document (Joe Hildebrand and Peter Saint-Andre) began to define such a signalling protocol, code-named Jingle. Upon communication with members of the Google Talk team, [36] it was discovered that the emerging Jingle approach was conceptually (and even syntactically) quite similar to the signalling protocol used in the Google Talk application. Therefore, in the interest of interoperability and adoption, we decided to harmonize the two approaches. The signalling protocol specified herein is, therefore, substantially equivalent to the original Google Talk protocol, with several adjustments based on feedback received from implementors as well as for publication by the XMPP Standards Foundation.

18 Acknowledgements

The authors would like to thank Rohan Mahy for his valuable input on early versions of the Jingle specifications. Thiago Camargo, Diana Cionoiu, Olivier Crête, Dafydd Harries, Antti Ijäs, Tim Julien, Lauri Kaila, Justin Karneges, Jussi Laako, Steffen Larsen, Marcus Lundblad, Dirk Meyer, Anthony Minessale, Akito Nozaki, Matt O'Gorman, Mike Ruprecht, Rob Taylor, Will Thompson, Matt Tucker, Justin Uberti, Saku Vainio, Unnikrishnan Vikrama Panicker, Brian West, Jeff Williams, and others have also provided helpful input. Thanks also to those who

[33] It is true that TINS made it relatively easy to implement an XMPP-to-SIP gateway; however, in line with the long-time Jabber philosophy of "simple clients, complex servers", it would be better to force complexity onto the server-side gateway and to keep the client as simple as possible.
[34] For example, one large ISP decided to switch to a pure XMPP approach after having implemented and deployed a dual-stack client for several years.
[35] XEP-0111: A Transport for Initiating and Negotiating Sessions <https://xmpp.org/extensions/xep-0111.html>.
[36] Google Talk is an instant messaging and voice/video chat service and client provided by Google; see <http://www.google.com/talk/>.

have commented on the Standards SIG [37] and Jingle [38] mailing lists.

[37] The Standards SIG is a standing Special Interest Group devoted to development of XMPP Extension Protocols. The discussion list of the Standards SIG is the primary venue for discussion of XMPP protocol extensions, as well as for announcements by the XMPP Extensions Editor and XMPP Registrar. To subscribe to the list or view the list archives, visit <https://mail.jabber.org/mailman/listinfo/standards/>.

[38] Before this specification was formally accepted by the XMPP Standards Foundation as an XMPP Extension Protocol, it was discussed on the semi-private <jingle@jabber.org> mailing list. This list has since been resurrected as a special-purpose venue for discussion of Jingle protocols and implementation; interested developers can subscribe and access the archives at at <http://mail.jabber.org/mailman/listinfo/jingle/>.