

XEP-0521: Jingle Audio/Video Conferences

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr
https://www.goffi.org

2026-10-13
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	av-conferences

This specification defines a way to hold multiparty conferences with an SFU via Jingle.

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 Requirements 1

3 Glossary 2

4 Overview 2

5 Joining a Conference 2

6 Receiving Streams 3

7 Adding a Stream 3

8 Origin of Forwarded Streams 3

9 Examples 4
- 9.1 Juliet Joins a Conference 4
- 9.2 Romeo Joins the Conference 6
- 9.3 Romeo Leaves the Conference 7

10 Configuration 8

11 Business Rules 8

12 Discovering Support 8

13 Security Considerations 9

14 IANA Considerations 10

15 XMPP Registrar Considerations 10

16 XML Schema 10

17 Acknowledgements 10

1 Introduction

Audio/Video calls are possible with a single destination via Jingle (XEP-0166) [1] and Jingle RTP Sessions (XEP-0167) [2], and associated XEPs. It may be desirable to have calls with multiple people at the same time. Three main strategies exist to achieve that:

- A mesh network where each peer contacts each other peer: this is straightforward to implement and doesn't require any extra service, but it is heavy on network bandwidth and doesn't scale.
- An MCU (Multipoint Conferencing Unit) which uses a central service that generally mixes the contents of participants: this is lighter on network bandwidth but heavy on computing resources and hard to scale.
- An SFU (Selective Forwarding Unit) which uses a service that redistributes streams of participants as efficiently as possible: this is lighter on computing resources, acceptable in terms of network bandwidth, and can scale.

The mesh strategy is covered by Multiparty Jingle (XEP-0272) [3]. There have been attempts to specify MCU and SFU strategies, notably with COnferences with LIghtweight BRIdging (COLIBRI) (XEP-0340) [4], but it is unmaintained and unused.
This specification proposes a simple way to implement an SFU strategy with a service called in a similar way to one-to-one calls.

2 Requirements

The design goals of this XEP are:

- Be as simple as possible to implement for clients that already implement one-to-one A/V calls.
- Be compatible with simulcast.
- Use a simple way to identify the originating XMPP entity of a stream.
- Use a simple way to configure the A/V conference room.

[1] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[2] XEP-0167: Jingle RTP Sessions <https://xmpp.org/extensions/xep-0167.html>.
[3] XEP-0272: Multiparty Jingle <https://xmpp.org/extensions/xep-0272.html>.
[4] XEP-0340: COnferences with LIghtweight BRIdging (COLIBRI) <https://xmpp.org/extensions/xep-0340.html>.

3 Glossary

- **A/V**: Audio/visual.
- **SFU**: Selective Forwarding Unit, a service that redistributes streams of participants as efficiently as possible.
- **SFU service**: An XMPP entity that services A/V conferences with an SFU.
- **Room**: A JID of an SFU service associated with a specific multiparty call.
- **Joining entity**: An XMPP entity that participates in an A/V conference room.
- **Stream**: A media (e.g. audio or video) exchanged in a conference, carried by a single content.
- **Simulcast**: Serving multiple versions of the same media content at different quality levels simultaneously, allowing the receiving end to select the most suitable version.

4 Overview

A/V conferences work by using a single Jingle session per participant. To join a conference, a client calls the JID of a conference room and sends its own streams. The conference service then delivers the streams of the other participants by adding and removing the corresponding contents of the session.
An <origin> element is used to associate a forwarded stream with the originating entity (see Origin of Forwarded Streams).
Ad-Hoc Commands (XEP-0050) [5] is used with a well-known node to configure the room.

5 Joining a Conference

To join a conference, a user calls the conference room JID using Jingle RTP Sessions (XEP-0167) [6] as for a one-to-one call. The calling entity sends its audio/video streams (typically from its microphone and its webcam) in an unidirectional way. It MUST have its content 'senders' attribute set to "initiator" as explained in Jingle (XEP-0166) [7] § 7.3.
A joining entity MAY send the same audio or video stream with different quality for simulcast purposes. It is up to the SFU service to then select which one to forward.
A participant leaves the conference by terminating the session.

[5] XEP-0050: Ad-Hoc Commands <https://xmpp.org/extensions/xep-0050.html>.
[6] XEP-0167: Jingle RTP Sessions <https://xmpp.org/extensions/xep-0167.html>.
[7] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.

6 Receiving Streams

The SFU service selects and delivers the streams of other participants by sending a **content-add** action on the participant's session, including one content per stream (e.g. one audio and one video content). The added contents MUST have their 'senders' attribute set to "responder". The action MUST include an <origin> element identifying the entity the streams originate from (see Origin of Forwarded Streams).
The contents of the streams of the same origin MUST share a single transport, declared with a BUNDLE group, as defined in Jingle Grouping Framework (XEP-0338) [8]. The service MAY use one BUNDLE group for the streams of an origin, or a single BUNDLE group for all the streams of the session.
The service MAY send **content-add** actions while the session is still in the PENDING state (as allowed by Jingle (XEP-0166) [9]), so that it can start delivering the streams of already joined participants right away.
The joining entity SHOULD **content-accept** each added content. If it does not want to receive the corresponding stream, it MAY **content-reject** it.
When a stream has to stop being delivered to a participant (e.g. because its origin left the conference, or because the service changed its selection), the service MUST remove the corresponding contents with a **content-remove** action.

7 Adding a Stream

A participant MAY add additional streams of its own to the session, e.g. to share its screen, by sending a **content-add** action with the corresponding contents (one content per stream, their 'senders' attribute set to "initiator"), as for its initial streams (see Joining a Conference). The participant MAY use a new BUNDLE group for the added streams, or add them to an existing BUNDLE group (see Receiving Streams). The service MAY then deliver the added streams to the other participants, with the participant as origin.
An entity MAY have several concurrent streams in a conference (e.g. streams from its camera and from its screen). Each stream is carried by a single content. How the streams of an entity are rendered (e.g. one tile per source, or combined) is up to the implementation.

8 Origin of Forwarded Streams

Each **content-add** action carrying streams forwarded by the SFU service MUST include an <origin> element, qualified by the 'urn:xmpp:jingle:av-conferences:0' namespace, as a child of the <jingle> element (i.e. a sibling of the <content> elements), as shown below. Its 'entity' attribute identifies the entity the streams originate from, as a URI (typically an "xmpp:" URL, in the form "xmpp:user@domain" or "xmpp:user@domain/resource").

[8] XEP-0338: Jingle Grouping Framework <https://xmpp.org/extensions/xep-0338.html>.
[9] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.

A **content-add** action carries the streams of a single origin, so a single <origin> element identifies the origin of all the contents of the action.

Listing 1: Content-add action of forwarded streams with their origin

```
<jingle action="content-add" ...>
  <group xmlns="urn:xmpp:jingle:apps:grouping:0" semantics="BUNDLE">
    <content name="0"/>
    <content name="1"/>
  </group>
  <content creator="responder" name="0" senders="responder">
    <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="audio">
      <!-{}- audio stream description -{}->
    </description>
    <transport xmlns="urn:xmpp:jingle:transports:ice-udp:1"
      ufrag="ufrag_2"
      pwd="ice_pwd_2">
      <!-{}- transport child elements -{}->
    </transport>
  </content>
  <content creator="responder" name="1" senders="responder">
    <!-{}- video content -{}->
  </content>
  <origin xmlns="urn:xmpp:jingle:av-conferences:0"
    entity="xmpp:romeo@montague.lit"/>
</jingle>
```

A stream MAY originate from a non-XMPP entity (e.g. using the SFU's own user interface or with a gateway to another protocol). In that case, the 'entity' attribute MAY use another URI scheme.

9 Examples

9.1 Juliet Joins a Conference

Listing 2: Juliet initiates a Jingle session with the conference room

```
<iq id="av_iq_1" type="set" from="juliet@capulet.lit/balcony" to="
   ball@conferences.shakespeare.lit">
     <jingle xmlns="urn:xmpp:jingle:1"
       sid="av_jingle_session_1"
       action="session-initiate"
       initiator="juliet@capulet.lit/balcony">
     <group xmlns="urn:xmpp:jingle:apps:grouping:0" semantics="BUNDLE
        ">
       <content name="video0"/>
       <content name="audio1"/>
```

```
      </group>
      <content creator="initiator" name="video0" senders="initiator">
        <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="video">
          <!-{}- video stream description -{}->
        </description>
        <transport xmlns="urn:xmpp:jingle:transports:ice-udp:1"
          ufrag="ufrag_1"
          pwd="ice_pwd_1">
        <fingerprint xmlns="urn:xmpp:jingle:apps:dtls:0" hash="sha-256
           " setup="actpass">
          <!-{}- some fingerprint -{}->
        </fingerprint>
      </transport>
    </content>
    <content creator="initiator" name="audio1" senders="initiator">
      <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="audio">
        <!-{}- audio stream description -{}->
      </description>
      <transport xmlns="urn:xmpp:jingle:transports:ice-udp:1"
        ufrag="ufrag_1"
        pwd="ice_pwd_1">
        <!-{}- transport child elements -{}->
      </transport>
  </content>
</jingle>
</iq>

<iq to="juliet@capulet.lit/balcony" from="ball@conferences.shakespeare
    .lit" id="av_iq_1" type="result"/>
```

Listing 3: Conference room accepts the session

```
<iq id="room_iq_1" type="set" from="ball@conferences.shakespeare.lit"
    to="juliet@capulet.lit/balcony">
  <jingle xmlns="urn:xmpp:jingle:1"
    sid="av_jingle_session_1"
    action="session-accept"
    responder="ball@conferences.shakespeare.lit">
    <group xmlns="urn:xmpp:jingle:apps:grouping:0" semantics="BUNDLE">
      <content name="video0"/>
      <content name="audio1"/>
    </group>
    <content creator="initiator" name="video0">
      <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="video">
        <!-{}- video stream description -{}->
      </description>
      <!-{}- transport element -{}->
    </content>
    <content creator="initiator" name="audio1">
      <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="audio">
```

```
        <!-{}- audio stream description -{}->
      </description>
      <!-{}- transport element -{}->
    </content>
  </jingle>
</iq>
```

9.2 Romeo Joins the Conference

Listing 4: Conference room adds Romeo's streams to Juliet's session

```
<iq id="room_iq_2" type="set" from="ball@conferences.shakespeare.lit"
    to="juliet@capulet.lit/balcony">
  <jingle xmlns="urn:xmpp:jingle:1"
    sid="av_jingle_session_1"
    action="content-add">
    <group xmlns="urn:xmpp:jingle:apps:grouping:0" semantics="BUNDLE">
      <content name="0"/>
      <content name="1"/>
    </group>
    <content creator="responder" name="0" senders="responder">
      <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="audio">
        <!-{}- audio stream description -{}->
      </description>
      <transport xmlns="urn:xmpp:jingle:transports:ice-udp:1"
        ufrag="ufrag_2"
        pwd="ice_pwd_2">
        <fingerprint xmlns="urn:xmpp:jingle:apps:dtls:0" hash="sha-256
            " setup="actpass">
          <!-{}- some fingerprint -{}->
        </fingerprint>
      </transport>
    </content>
    <content creator="responder" name="1" senders="responder">
      <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="video">
        <!-{}- video stream description -{}->
      </description>
      <transport xmlns="urn:xmpp:jingle:transports:ice-udp:1"
        ufrag="ufrag_2"
        pwd="ice_pwd_2">
        <fingerprint xmlns="urn:xmpp:jingle:apps:dtls:0" hash="sha-256
            " setup="actpass">
          <!-{}- some fingerprint -{}->
        </fingerprint>
      </transport>
    </content>
    <origin xmlns="urn:xmpp:jingle:av-conferences:0" entity="
        xmpp:romeo@montague.lit"/>
```

```
  </jingle>
</iq>

<iq to="ball@conferences.shakespeare.lit" from="juliet@capulet.lit/
    balcony" id="room_iq_2" type="result"/>
```

Listing 5: Juliet accepts the added contents

```
<iq id="av_iq_2" type="set" from="juliet@capulet.lit/balcony" to="
    ball@conferences.shakespeare.lit">
  <jingle xmlns="urn:xmpp:jingle:1"
    sid="av_jingle_session_1"
    action="content-accept">
    <content creator="responder" name="0">
      <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="audio">
        <!-{}- audio stream description -{}->
      </description>
      <!-{}- transport element -{}->
    </content>
    <content creator="responder" name="1">
      <description xmlns="urn:xmpp:jingle:apps:rtp:1" media="video">
        <!-{}- video stream description -{}->
      </description>
      <!-{}- transport element -{}->
    </content>
  </jingle>
</iq>

<iq to="juliet@capulet.lit/balcony" from="ball@conferences.shakespeare
    .lit" id="av_iq_2" type="result"/>
```

9.3 Romeo Leaves the Conference

Listing 6: Conference room removes Romeo's streams from Juliet's session

```
<iq id="room_iq_3" type="set" from="ball@conferences.shakespeare.lit"
    to="juliet@capulet.lit/balcony">
  <jingle xmlns="urn:xmpp:jingle:1"
    sid="av_jingle_session_1"
    action="content-remove">
    <content creator="responder" name="0"/>
    <content creator="responder" name="1"/>
  </jingle>
</iq>

<iq to="ball@conferences.shakespeare.lit" from="juliet@capulet.lit/
    balcony" id="room_iq_3" type="result"/>
```

10 Configuration

It may be desirable to have configuration options on the SFU service or on one of its rooms. For instance, a conference room may offer the option to record the session. While the definition of those options is beyond the scope of this specification, the configuration MUST be done via Ad-Hoc Commands (XEP-0050) [10] on the well-known node "urn:xmpp:av-conferences:config".

11 Business Rules

Only other peers' streams are sent to a joining entity. An SFU service MUST NOT send back a stream to the entity that initially sent it.
An SFU service MUST NOT initiate independent Jingle sessions to deliver streams to a joined entity: the streams are only delivered as contents of the entity's session, via **content-add** actions.
An SFU service MUST ensure that the names of the contents it adds do not collide with the existing contents of the session, as a BUNDLE group identifies its contents by name only.

12 Discovering Support

If a client or a service supports the protocol specified in this XEP, it MUST advertise it by including the 'urn:xmpp:jingle:av-conferences:0' discovery feature in response to a Service Discovery (XEP-0030) [11] information request.

Listing 7: Service Discovery information request

```
<iq type='get'
    from='juliet@example.org/balcony'
    to='romeo@example.org/orchard'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 8: Service Discovery information response

```
<iq type='result'
    from='romeo@example.org/orchard'
    to='juliet@example.org/balcony'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:xmpp:jingle:av-conferences:0'/>
```

[10] XEP-0050: Ad-Hoc Commands <https://xmpp.org/extensions/xep-0050.html>.
[11] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
    ...
  </query>
</iq>
```

If an entity is an A/V conferences service as specified by this XEP, it MUST have an identity with a category of "conference" and a value of "audio-video" in a response to a Service Discovery (XEP-0030) [12] information request.

Listing 9: Service Discovery information request

```
<iq type='get'
    from='juliet@example.org/balcony'
    to='ball@conference.example.org'
    id='disco2'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 10: Service Discovery information response

```
<iq type='result'
    from='ball@conference.example.org'
    to='juliet@example.org/balcony'
    id='disco2'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <identity category='conference' type='audio-video'/>
    ...
    <feature var='urn:xmpp:jingle:av-conferences:0'/>
    ...
  </query>
</iq>
```

13 Security Considerations

Streams are sent to the SFU service for redistribution. Although it is technically possible to have the stream encrypted end-to-end, this specification assumes that streams are decoded at the SFU level and are therefore not end-to-end encrypted, unlike one-to-one calls. Future specifications may outline a method for maintaining end-to-end encryption, but this is beyond the scope of this XEP; developers and end-users MUST assume that streams are decrypted at the SFU service level.

The general security considerations for one-to-one calls apply, with the added risk that more parties are participating in the call. However, the IP address of a device sending a stream is only known by the SFU service, not by other peers, unlike in one-to-one or Multiparty Jingle (XEP-0272) [13] calls.

[12] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[13] XEP-0272: Multiparty Jingle <https://xmpp.org/extensions/xep-0272.html>.

The <origin> element is not authenticated: it is provided by the SFU service, which is responsible for having established the identity of the entities that sent the streams (e.g. via the JID of their Jingle sessions). The service could however alter an <origin> or a stream, either intentionally or through a bug.

14 IANA Considerations

TODO

15 XMPP Registrar Considerations

TODO

16 XML Schema

TODO

17 Acknowledgements

Thanks to NLNet foundation/NGI Assure for funding the work on this specification.