XEP-0340: COnferences with LIghtweight BRIdging (COLIBRI)

Emil Ivov
mailto:emcho@jitsi.org
xmpp:emcho@sip-communicator.org

Lyubomir Marinov
mailto:lubo@jitsi.org
xmpp:lubo@sip-communicator.org

Philipp Hancke
mailto:fippo@andyet.com
xmpp:fippo@goodadvice.pages.de

2017-09-11
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	colibri

This specification defines an XMPP extension that allows real-time communications clients to discover and interact with conference bridges that provide conference mixing or relaying capabilities.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

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

Contents

1 Introduction

Delivering Conference Information to Jingle Participants (Coin) (XEP-0298) [1] defines a way for XMPP agents to establish and participate in tightly coupled conference calls. Such conference calls would typically involve a number of regular participants that establish direct one-to-one sessions with a single entity, often referred to as a focus agent. Focus agents are generally responsible for making sure that media sent from one call participant would be distributed to all others so that everyone would effectively hear or see everyone else. In other words they often act as media mixers.

Depending on the mixing technology used by media mixers, they may require significant bandwidth, processing resources or both. It is hence common for mixers to be hosted on dedicated servers that can provide such resources. They are then made reachable as rendez-vous points and conference call participants are required to call in, in order to join a conference call. This requires a certain amount of pre-call configuration to be completed by the service maintainers in order to create conference rooms and grant proper access to the expected participants. The authorization credentials are then often relayed to the participants in preparation of the call by other means, such as IM or mail.

In certain situations, such pre-call preparations are inconvenient and it is important for users to be able to establish ad-hoc conference calls. One way to achieve this is for user agents themselves to act as focus agents and media mixers. Everyone else just calls the user at the focus agent, who then can decide whether to accept or reject the calls as they arrive. This works particularly well for audio only calls as the amount of bandwidth and processing resources that they require is generally within reach for end-user devices.

The situation is quite different for video calls. Media decoding and especially encoding require considerably more resources with video than they do with audio. Today, encoding a single video flow with an acceptable quality is often the maximum that can be expected from an end-user device. The advantages that come with Moore's law will likely be insufficient to improve this, given the massive shift toward mobile devices and the ever-increasing user expectations toward video quality.

Therefore, this specification (COLIBRI) aims to provide a means for user agents to interact with conference mixers. Such interaction allows user agents to allocate mixing channels, indicate what conferences they should be attached to, what integers the various payload types map to, etc. Using COLIBRI would hence allow any user agent to organize conference calls and act as a signalling focus by outsourcing the actual media mixing to a dedicated mixer.

2 Terminology

Focus or Focus Agent The terms apply to XMPP agents that, in terms of signalling, stand at the center of a tightly-coupled conference call. In other words, all conference participants establish a Jingle (XEP-0166) XEP-0166: Jingle

[1] XEP-0298: Delivering Conference Information to Jingle Participants (Coin) <https://xmpp.org/extensions/xep-0298.html>.

<https://xmpp.org/extensions/xep-0166.html>. session with and only with that agent. Focus Agents are not necessarily performing media mixing themselves. In fact, the very purpose of this specification is to provide them with a means of handling this elsewhere.

Mixer or Bridge Throughout this document the term is used to depict an entity that is responsible for mixing and delivering to conference participants all media exchanged in a conference call. Mixers or bridges can provide their service by either performing Content Mixing, or RTP translation or both.

Content Mixing The term refers to a kind of media processing where the content of multiple input RTP streams is ”mixed” into a single output stream. In conference calls audio mixing implementations generally simply add and adjust all source audio streams to produce their single output stream. Video content mixing, on the other hand, is often implemented by creating composite images containing individual frames from the input streams. Another common implementation consists in producing an output that is identical to one of the input streams, often the one belonging to a currently active speaker.

RTP Translation RFC 3550 defines a translator as ”an intermediate system that forwards RTP packets with their synchronization source identifier intact.” This specification respects that definition but it also uses ”RTP Translation” in opposition with ”Content Mixing”. Conference bridges that perform RTP translation simply redirect each incoming RTP packet to all participants, often excluding the one where it came from. Contrary to content mixing, rtp translation generally requires less processing resources since it does not involve media manipulation. Bandwidth requirements on the other hand, could be significantly higher with RTP translation than with content mixing.

3 Requirements

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

1. Provide a means for conference focus agents to interact with conference mixers in order to configure payload type mappings and allocate ports or other resources that they could then advertise in Jingle sessions so that all media would traverse the bridge.
2. Impose no COLIBRI specific requirements on non-focus participants so that any Jingle supporting client would be able to participate.
3. [TODO] Anything else?

4 How It Works

This section provides a friendly introduction to COLIBRI.
In essence, the goal of COLIBRI is to provide focus agents with a way of using remote mixers as

if as they were available locally. The most important part of that is the possibility to allocate ports on the mixer interfaces and then use these ports when establishing Jingle sessions with the various participants.
Every participant in the conference call is assigned one port for RTP data and one for RTCP. An RTP/RTCP port couple is called a channel. Each participant would use one channel per media type. That is, a client participating with audio only would get one channel, while another one that joins with both audio and video would get two: one for audio and one for video.
Channels are used for streams from the bridge to participants and from participants to the bridge. Typically a channel would contain one stream from a participant to a bridge, for example their webcam or desktop, and one or more streams in the opposite direction (e.g. webcam or desktop streams from other participants to the one using this channel). This is not a requirement though and a channel can certainly be used for transportation of multiple streams in both directions in cases where one bridge is connected to another.
Typically channels would be created by the entity controlling a conference call. This could either be a conferencing server or a smart client capable of handling conferences. We would refer to both of these as the "focus". In either case, the important part is that the focus terminates all signalling. It is a signalling endpoint and it is responsible for all aspects of call signalling including offer/answer.
In other words, when setting up a conference, a focus would first allocate the necessary channels, then directly initiate sessions (invite) other participants into the call. Only, when sending the invitations to these participants, the focus would use the transport information (addresses and ports) that it would have received from the COLIBRI bridge, rather than its own.

5 Use cases

5.1 Creating a conference

The most important thing about setting up a conference is the creation of channels for every participant. Conference setup is not the only chance an organiser gets to declare all participants but typically when a conference call is setup it is because there are at least some number of known participants and there would be no point in delaying channel creation for them.
The following example shows how Romeo creates an audio/video conference at a bridge, requesting that three participant channels be created.

Listing 1: Creating a conference

```
SEND: <iq from='romeo@montague.lit/orchard'
          id='zid615d9'
          to='garden.montague.lit'
          type='set'>
        <conference xmlns='http://jitsi.org/protocol/colibri'>
          <content name='audio'>
            <channel initiator='true'>
```

```
            [optional payload and transport description]
          </channel>
          <channel initiator='true'/>
          ...
          <channel initiator='true'/>
        </content>
        <content name='video'>
          <channel initiator='true'/>
          ...
          <channel initiator='true'/>
        </content>
      </conference>
    </iq>
```

Notice how the 'initiator' channel above is set to true. The setting determines ICE and DTLS/SRTP behaviour for the bridge. In this specific case, 'initiator' being set to 'true' Romeo is requesting that the bridge behave as the initiator of the session which means that it would try to be the controlling ICE agent and also assume the 'dtls-actpass' role for DTLS/SRTP negotiation. A value of 'false' would have meant that the bridge would behave as the controlled ICE agent and assume the 'dtls-active' role.
When sending its result back, the bridge confirms creation of the requested channels and it also delivers transport information that would be necessary for participants to transport media to the bridge. These would most often include ICE candidates, ufrag and pwd parameters, and DTLS fingerprints.
Note that ICE is not mandatory for use and COLIBRI bridges can just as well perform Hosted NAT Traversal using latching and a RAW-UDP transport.

Listing 2: Result

```
RECV: <iq type='result' to='romeo@montague.lit/orchard'
          from='garden.montague.lit'
          id='zid615d9'>
        <conference xmlns='http://jitsi.org/protocol/colibri'
          id='cafb6f2c8197818e'>
          <content name='audio'>
            <channel rtp-level-relay-type='mixer' direction='recvonly'
                initiator='true' id='c6a142b7cf728fd0' expire='60'>
              <source xmlns='urn:xmpp:jingle:apps:rtp:ssma:0' ssrc='
                  3716204482'/>
              <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                  pwd='5d0mj' ufrag='amiq'>
                <fingerprint xmlns='urn:xmpp:jingle:apps:dtls:0' hash=
                    'sha-1'>
                  99:...:F6
                </fingerprint>
                <candidate type='host' protocol='udp' id='ca' ip='
                    10.0.1.1'
```

```
                component='1' port='5144' foundation='3'
                    generation='0'
                priority='2113932031' network='0'/>
            <candidate type='host' protocol='udp' id='ga' ip='
                10.0.1.1'
                component='2' port='5145' foundation='3'
                    generation='0'
                priority='2113932030' network='0'/>
          </transport>
        </channel>
        ...
      </content>
      <content name='video'>
        <channel rtp-level-relay-type='mixer' direction='recvonly'
            initiator='true' id='c9726594ccb4ede7' expire='60'>
          ...
        </channel>
        ...
      </content>
    </conference>
  </iq>
```

The above "result" also contains the following elements of interest for every channel:
- an ID that is necessary for any further modification from that the focus wants to set on a channel. [FIXME: clients should be able to specify these id-s so as not to rely on ordering to identify channels and get complexes thinking they are SDP parsers]
- an rtp-level-relay-type attribute with possible values of 'mixer' and 'translator' indicating how the bridge is going to deliver data on a specific channel [FIXME: this would definitely need to be specifiable from the client].
- mixer channels would also include ssrc-s for that channels in question. This is particularly necessary when SSRC-s need to be announced to participants (because people never learned how RTP works and are afraid from anything that wasn't explicitly announced with an Offer-/Answer exchange). Generally such announcements would be possible by simply propagating SSRCs that other participants announce. In a mixed flow however the SSRC would belong to the mixer (or COLIBRI bridge) so it would need to be known in advance. attribute
- the initiator value is echoed
- expire describes how many seconds the bridge will keep the channel open without media activity

5.2 Updating payload information for a channel

Channel updates can happen for various reasons. The following examples illustrate two of them:
- specifying payload types. While payload types in RTP are sometimes static (e.g. for older codecs such as G.711), this is not always the case for more recent types, which need to be assigned dynamically during session establishment. The tricky part here is that dynamic

means dynamic so every participant in a conference call may end up expecting different payload types. As a result, a COLIBRI bridge SHOULD know about everyone's expectations, which is why channels are updated with payload types. Note that if a bridge does see unknown payload types it MUST still relay them to other participants as they might have used some other mechanism to make sure they know what they mean.
- DTLS/SRTP fingerprints.

Listing 3: Focus updates payload information for a channel

```
SEND: <iq to='garden.montague.lit' from='romeo@montague.lit/orchard'
          type='set' id='74s'>
        <conference xmlns='http://jitsi.org/protocol/colibri'
            id='cafb6f2c8197818e'>
          <content name='audio'>
            <channel id='c6a142b7cf728fd0' initiator='true'>
              <payload-type id='111' name='opus' clockrate='48000'
                  channels='2'/>
              <payload-type id='0' name='PCMU' clockrate='8000'
                  channels='1'/>
              <payload-type id='8' name='PCMA' clockrate='8000'
                  channels='1'/>
              <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                   ufrag='WP+qAQZGnDhhM+87' pwd='0
                       Uxdzy9gTryxAkmAx2LD1TYR'>
                <fingerprint hash='sha-256' xmlns='
                    urn:xmpp:jingle:apps:dtls:0'
                     setup='active'>
                  08:...:C7
                </fingerprint>
              </transport>
            </channel>
          </content>
          <content name='video'>
            <channel id='c9726594ccb4ede7' initiator='true'>
              <payload-type id='100' name='VP8' clockrate='90000'
                  channels='1'/>
              <payload-type id='116' name='red' clockrate='90000'
                  channels='1'/>
              <payload-type id='117' name='ulpfec' clockrate='90000'
                  channels='1'/>
              <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                   ufrag='hpx55NAN46sNYwF+' pwd='hkg/
                       YRpjXZx4qDG3KbzB3qr1'>
                <fingerprint hash='sha-256' xmlns='
                    urn:xmpp:jingle:apps:dtls:0'
                     setup='active'>
                  08:...:C7
                </fingerprint>
              </transport>
```

```
            </channel>
          </content>
        </conference>
      </iq>
```

Note that while the result in this case is essentially an acknowledgement, it still carries a full representation of the bridge.

Listing 4: Result

```
RECV: <iq to='romeo@montague.lit/orchard' from='garden.montague.lit'
        type='result' id='74s'>
        <conference xmlns='http://jitsi.org/protocol/colibri'
            id='cafb6f2c8197818e'>
          <content name='audio'>
            <channel rtp-level-relay-type='mixer' direction='recvonly'
                initiator='true' id='c6a142b7cf728fd0' expire='60'>
              <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                  pwd='5d0mj' ufrag='amiq'>
                <fingerprint xmlns='urn:xmpp:jingle:apps:dtls:0' hash=
                    'sha-1'>
                  99:...:F6
                </fingerprint>
                <candidate type='host' protocol='udp' id='ca' ip='
                    10.0.1.1' component='1'
                    port='5144' foundation='3' generation='0' priority
                        ='2113932031'
                    network='0'/>
                <candidate type='host' protocol='udp' id='ga' ip='
                    10.0.1.1' component='2'
                    port='5145' foundation='3' generation='0' priority
                        ='2113932030'
                    network='0'/>
              </transport>
            </channel>
            ...
          </content>
          <content name='video'>
            <channel rtp-level-relay-type='translator' initiator='true
                '
                id='c9726594ccb4ede7' expire='60'>
              ...
            </channel>
            ...
          </content>
        </conference>
      </iq>
```

5.3 Updating transport information for a channel

Listing 5: Result

```
SEND: <iq to='garden.montague.lit' from='romeo@montague.lit/orchard'
          type='set' id='1581'>
        <conference xmlns='http://jitsi.org/protocol/colibri' id='
            cafb6f2c8197818e'>
          <content name='audio'>
            <channel id='c6a142b7cf728fd0' initiator='true'>
              <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'>
                <candidate foundation='2241210590' component='1'
                    protocol='udp'
                    priority='2113937151' ip='192.168.2.101' port='
                        61141' type='host'
                    generation='0' network='1' id='1234'/>
              </transport>
            </channel>
          </content>
        </conference>
      </iq>
```

Listing 6: Result

```
RECV: <iq type='result' to='romeo@montague.lit/orchard' from='garden.
    montague.lit' id='1581'>
        <conference xmlns='http://jitsi.org/protocol/colibri' id='
            cafb6f2c8197818e'>
          <content name='audio'>
            <channel rtp-level-relay-type='translator' initiator='true
                '
                id='c6a142b7cf728fd0' expire='60'>
              <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                  pwd='5d0mja27fgl83r9tsl1b9gkk4f' ufrag='amiqp'>
                <fingerprint xmlns='urn:xmpp:jingle:apps:dtls:0' hash=
                    'sha-1'>
                  99:..:F6
                </fingerprint>
                <candidate type='host' protocol='udp' id='1' ip='
                    10.0.1.1' component='1'
                    port='5144' foundation='3' generation='0' priority
                        ='2113932031' network='0'/>
                <candidate type='host' protocol='udp' id='2' ip='
                    10.0.1.1' component='2'
                    port='5145' foundation='3' generation='0' priority
                        ='2113932030' network='0'/>
              </transport>
            </channel>
          </content>
        </conference>
```

```
      </iq>
```

Essentially that information is the transport description from the bridge.

5.4 Adding a new channel

ICE candidates are another reason why a focus might want to update a channel. Earlier examples indicated how conference setup could be completed without providing any transport information whatsoever. Whenever that is the case, such information would need to be provided through channel modification.

Listing 7: Adding a new channel

```
SEND: <iq to='garden.montague.lit' from='romeo@montague.lit/orchard'
    type='get' id='247'>
        <conference xmlns='http://jitsi.org/protocol/colibri' id='
            cafb6f2c8197818e'>
          <content creator='initiator' name='audio'>
            <channel initiator='true'/>
          </content>
          <content creator='initiator' name='video'>
            <channel initiator='true'/>
          </content>
        </conference>
      </iq>
```

Listing 8: Result

```
RECV: <iq type='result' to='romeo@montague.lit/orchard' from='garden.
    montague.lit' id='247'>
        <conference xmlns='http://jitsi.org/protocol/colibri' id='49
            a91b4f6694bc6a'>
          <content name='audio'>
            <channel rtp-level-relay-type='mixer' direction='recvonly'
                initiator='true'
                id='e97d7f794fbed74b' expire='60'>
              <source xmlns='urn:xmpp:jingle:apps:rtp:ssma:0' ssrc='
                  2579640556'/>
              <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
                  pwd='75e88spurhqbih628ord5a3l9b' ufrag='6c7a6'>
                <fingerprint xmlns='urn:xmpp:jingle:apps:dtls:0' hash=
                    'sha-1'>
                  A9:...:2F
                </fingerprint>
                <candidate type='host' protocol='udp' id='1' ip='
                    10.0.1.1' component='1'
                    port='5168' foundation='3' generation='0' priority
                        ='2113932031'
```

```
                network='0'/>
            <candidate type='host' protocol='udp' id='2' ip='
                10.0.1.1' component='2'
                port='5169' foundation='3' generation='0' priority
                    ='2113932030'
                network='0'/>
          </transport>
        </channel>
      </content>
      <content name='video'>
        <channel rtp-level-relay-type='translator' initiator='true
            '
            id='b4557f274216f99a' expire='60'>
          <transport xmlns='urn:xmpp:jingle:transports:ice-udp:1'
              pwd='1oriuuhjfq6884ln9d3g1rjq13' ufrag='3gh3o'>
            <fingerprint xmlns='urn:xmpp:jingle:apps:dtls:0' hash=
                'sha-1'>
              D7:...:C2
            </fingerprint>
            <candidate type='host' protocol='udp' id='1' ip='
                10.0.1.1' component='1'
                port='5170' foundation='3' generation='0' priority
                    ='2113932031'
                network='0'/>
            <candidate type='host' protocol='udp' id='2' ip='
                10.0.1.1' component='2'
                port='5171' foundation='3' generation='0' priority
                    ='2113932030'
                network='0'/>
          </transport>
        </channel>
      </content>
    </conference>
  </iq>
```

6 Determining Support

If an entity supports COLIBRI, it SHOULD advertise that fact by returning a feature of "http://jitsi.org/protocol/colibri" in response to a Service Discovery (XEP-0030) [2] information request.

Listing 9: Service Discovery Information Request

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

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

```
    to='belfry.shakespeare.lit'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 10: Service Discovery Information Response

```
<iq from='belfry.shakespeare.lit'
    id='ku6e51v3'
    to='kingclaudius@shakespeare.lit/castle'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='http://jitsi.org/protocol/colibri'/>
  </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) [3]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

7 Security Considerations

PENDING

8 Open Issues

PENDING

9 XML Schemas

PENDING

10 Acknowledgements

Jitsi's participation in this specification is funded by the NLnet Foundation.

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