XEP-0298: Delivering Conference Information to Jingle Participants (Coin)

Emil Ivov
mailto:emcho@jitsi.org
xmpp:emcho@jit.si

Enrico Marocco
mailto:enrico.marocco@telecomitalia.it
xmpp:enrico@tilab.com

Saúl Ibarra Corretgé
mailto:saul@ag-projects.com
xmpp:saul@ag-projects.com

2015-07-02
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	coin

This specification defines an XMPP extension for tightly coupled conference calls. It allows users who participate in multiparty Jingle calls via a focus agent (mixer) to retrieve information and receive notifications about the state of the call and the other participants. This extension is also meant to provide a straightforward way of connecting SIP and XMPP clients to the same conference room.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

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

Contents

1 Introduction

Jingle (XEP-0166) [1] defines a way for XMPP agents to establish and control one-to-one media sessions. It is possible for either participant in such a session to also establish additional conversations and then serve as a media mixer. This could be viewed as a classic conference call scenario and is also often referred to as a tightly coupled conference.
Basic participation or hosting of tightly coupled conferences requires no specific protocol support. With the exception of the mixing agent, call members, however, all perceive the session as a regular one-to-one call. They have no way of obtaining additional information about how many and what other users are participating.
The Coin extension (short for Conference Information) allows media mixers to deliver to participants additional information about the status of the call, and that of its members.
A conference participant exchanges Coin IQs only with the agent they have established a session with. This means that it can also be used in cases where only a subset of the users on a call are using XMPP while others are connected via alternative mechanisms such as SIP conferencing as defined in RFC 4579 [2]

2 Terminology

Mixer Throughout this document the term is used to depict an entity that is responsible for mixing and delivering to conference participants both signalling and media. Other specifications refer to mixers and focus agents as two distinct entities but we find this separation to be unnecessary in the current specification and view both as a single logical entity. This entity may be a person hosting the conference and doing the mixing or a dedicated entity to which participants connect in order to establish a conference. For the purposes of this specification, both scenarios are equivalent.

3 Requirements

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

1. Provide a means for mixer agents in tightly coupled conferences to advertise call and member state information to the call participants.
2. Reuse the existing format and XML schema already defined in RFC 4575.
3. Impose no requirements on agents joining the call other than those necessary to establish a regular one-to-one call.

[1] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[2] RFC 4579: Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents <http://tools.ietf.org/html/rfc4579>.

4. Allow straightforward interoperability with other conferencing mechanisms such as RFC 4579 [3] or Multiparty Jingle (XEP-0272) [4]

4 How It Works

This section provides a friendly introduction to Coin.
In essence Coin allows clients that establish Jingle calls to determine whether their peer is acting as a mixer or to announce themselves as such. This way non-mixer participants would know when they are participating in a conference call and would be able to notify the user accordingly.
Once in a call, participants and mixers can use Coin to exchange RFC 4575 [5] conference information indicating what participants are currently on the call and what their status is.

5 Creating a conference call

When creating conference calls mixers SHOULD indicate the nature of the call as early as possible. This is necessary in order to allow other participating user agents to adapt their user interface in an appropriate way.

```
<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>
    <conference-info xmlns='urn:xmpp:coin:1' isfocus='true'/>
  </jingle>
</iq>
```

Similarly mixers being dialed by new participants SHOULD indicate the nature of the call by including the <conference-info/> element into the Jingle session-accept message.
Finally, when transforming an existing one-to-one session into a conference or vice-versa a mixer SHOULD send a Jingle session-info message with the appropriate <conference-info/>

[3] RFC 4579: Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents <http://tools.ietf.org/html/rfc4579>.
[4] XEP-0272: Multiparty Jingle <https://xmpp.org/extensions/xep-0272.html>.
[5] RFC 4575: A Session Initiation Protocol (SIP) Event Package for Conference State <http://tools.ietf.org/html/rfc4575>.

element.
Note that presence of the <conference-info/> element is only determines whether the party sending it is currently acting as a mixer or not. If multiple peers in a call are independently acting as mixers they should all indicate their status accordingly.

6 Delivering conference information

Once a conference call has been established and advertised as such, a mixer MAY at any point send information describing the state of the call and its current participants.

```
<iq from='romeo@montague.lit/orchard'
    id='zid615d9'
    to='juliet@capulet.lit/balcony'
    type='set'>
    <jingle xmlns='urn:xmpp:jingle:1' sid='a73sjjvkla37jfea'/>
    <conference-info xmlns="urn:ietf:params:xml:ns:conference-info"
                     entity="xmpp:romeo@monague.lit/orchard"
                     state="full"
                     version="1">

     <!-- CONFERENCE INFO -->
     <conference-description>
       <subject>Ending a relationship</subject>
     </conference-description>

     <!-- CONFERENCE STATE -->
     <conference-state>
       <user-count>3</user-count>
     </conference-state>

     <!-- USERS -->
     <users>
       <user entity="xmpp:romeo@montague.lit" state="full">
         <display-text>Romeo</display-text>

         <!-- ENDPOINTS -->
         <endpoint entity="xmpp:romeo@montague.lit/orchard">
           <display-text>Romeo's␣smartphone</display-text>
␣␣␣␣␣␣␣␣␣␣␣<status>disconnected</status>
␣␣␣␣␣␣␣␣␣␣␣<disconnection-info>
␣␣␣␣␣␣␣␣␣␣␣␣␣<when>2011-01-31T20:00:00Z</when>
␣␣␣␣␣␣␣␣␣␣␣␣␣<reason>poisoned</reason>
␣␣␣␣␣␣␣␣␣␣␣</disconnection-info>

␣␣␣␣␣␣␣␣␣␣␣<!--␣MEDIA␣-->
␣␣␣␣␣␣␣␣␣␣␣<media␣id="1">
```

```
␣␣␣␣␣␣␣␣␣␣␣␣<display-text>main␣audio</display-text>
␣␣␣␣␣␣␣␣␣␣␣␣<type>audio</type>
␣␣␣␣␣␣␣␣␣␣␣␣<src-id>432424</src-id>
␣␣␣␣␣␣␣␣␣␣␣</media>
␣␣␣␣␣␣␣␣␣␣</endpoint>
␣␣␣␣␣␣␣␣</user>

␣␣␣␣␣␣␣␣<user␣entity="xmpp:juliet@capulet.lit"␣state="full">
␣␣␣␣␣␣␣␣␣<display-text>Juliet</display-text>

␣␣␣␣␣␣␣␣␣<!--␣ENDPOINTS␣-->
␣␣␣␣␣␣␣␣␣<endpoint␣entity="juliet@capulet.lit/balcony">
␣␣␣␣␣␣␣␣␣␣␣<display-text>Juliet's netbook</display-text>
            <status>connected</status>

            <!-- MEDIA -->
            <media id="1">
              <type>audio</type>
              <src-id>2124</src-id>
            </media>
          </endpoint>
        </user>

        <!-- USER -->
        <user entity="sip:alice@example.com" state="full">
          <display-text>Alice</display-text>

          <!-- ENDPOINTS -->
          <endpoint entity="sip:4kfk4j392jsu@example.com;grid=433kj4j3u
              ">
            <status>connected</status>

            <!-- MEDIA -->
            <media id="1">
              <type>audio</type>
              <src-id>534232</src-id>
            </media>
          </endpoint>
        </user>
      </users>
    </conference-info>
</iq>
```

The IQ message containing the conference info document MAY also contain a jingle element with the session id attribute indicting the session to which the conference information refers to.

7 Determining Support

If an entity supports Coin, it SHOULD advertise that fact by returning a feature of "urn:xmpp:coin:1" in response to a Service Discovery (XEP-0030) [6] information request.

Listing 1: 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 2: Service Discovery Information Response

```
<iq from='laertes@shakespeare.lit/castle'
    id='ku6e51v3'
    to='kingclaudius@shakespeare.lit/castle'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:coin:1'/>
  </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) [7]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

8 Security Considerations

PENDING: RFC 4575 mostly talks about authentication conference-info subscriptions but these are not part of this specification. The authors are hence currently unaware of any other Coin specific security considerations

9 Open Issues

This document provides a basic description of a simple way to support tightly coupled conference calls. It is in many respects still a stub and a number of open issues require the attention of the community:

[6] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[7] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

1. Need to define best practices for user agents to easily determine whether the request of user to establish a conference call should result in a Muji or a Coin conference.

10 XML Schemas

10.1 Advertising Conf Calls

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

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

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0298: http://www.xmpp.org/extensions/xep-0298.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name="conference-info" type="empty">
    <xs:complexType>
      <xs:attribute name='isfocus' type='xs:boolean' required='true'/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

10.2 Conference Info

```
<?xml version="1.0" encoding="UTF-8" ?>
   <xs:schema
   targetNamespace="urn:ietf:params:xml:ns:conference-info"
   xmlns:tns="urn:ietf:params:xml:ns:conference-info"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns="urn:ietf:params:xml:ns:conference-info"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified">

   <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      RFC 4575: http://tools.ietf.org/html/rfc4575  and reused by
      XEP-0298 http://www.xmpp.org/extensions/xep-0298.html
    </xs:documentation>
```

```
</xs:annotation>

 <!--
   This imports the xml:language definition
 -->
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
     schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
 <!--
   CONFERENCE ELEMENT
 -->
    <xs:element name="conference-info" type="conference-type"/>
    <!--
       CONFERENCE TYPE
    -->
    <xs:complexType name="conference-type">
     <xs:sequence>
      <xs:element name="conference-description"
       type="conference-description-type" minOccurs="0"/>
      <xs:element name="host-info"
       type="host-type" minOccurs="0"/>
      <xs:element name="conference-state"
       type="conference-state-type" minOccurs="0"/>
      <xs:element name="users"
       type="users-type" minOccurs="0"/>
      <xs:element name="sidebars-by-ref"
       type="uris-type" minOccurs="0"/>
      <xs:element name="sidebars-by-val"
       type="sidebars-by-val-type" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax"
       minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="entity"
      type="xs:anyURI" use="required"/>
     <xs:attribute name="state"
      type="state-type" use="optional" default="full"/>
     <xs:attribute name="version"
      type="xs:unsignedInt" use="optional"/>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <!--
       STATE TYPE
    -->
    <xs:simpleType name="state-type">
     <xs:restriction base="xs:string">
      <xs:enumeration value="full"/>
      <xs:enumeration value="partial"/>
      <xs:enumeration value="deleted"/>
     </xs:restriction>
    </xs:simpleType>
```

```
<!--
   CONFERENCE DESCRIPTION TYPE
-->
<xs:complexType name="conference-description-type">
 <xs:sequence>
  <xs:element name="display-text"
   type="xs:string" minOccurs="0"/>
  <xs:element name="subject"
   type="xs:string" minOccurs="0"/>
  <xs:element name="free-text"
   type="xs:string" minOccurs="0"/>
  <xs:element name="keywords"
   type="keywords-type" minOccurs="0"/>
  <xs:element name="conf-uris"
   type="uris-type" minOccurs="0"/>
  <xs:element name="service-uris"
   type="uris-type" minOccurs="0"/>
  <xs:element name="maximum-user-count"
   type="xs:unsignedInt" minOccurs="0"/>
  <xs:element name="available-media"
   type="conference-media-type" minOccurs="0"/>
  <xs:any namespace="##other" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<!--
   HOST TYPE
-->
<xs:complexType name="host-type">
 <xs:sequence>
  <xs:element name="display-text" type="xs:string"
   minOccurs="0"/>
  <xs:element name="web-page" type="xs:anyURI"
   minOccurs="0"/>
  <xs:element name="uris" type="uris-type"
   minOccurs="0"/>
  <xs:any namespace="##other" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<!--
   CONFERENCE STATE TYPE
-->
<xs:complexType name="conference-state-type">
 <xs:sequence>
  <xs:element name="user-count" type="xs:unsignedInt"
   minOccurs="0"/>
```

```
    <xs:element name="active" type="xs:boolean"
     minOccurs="0"/>
    <xs:element name="locked" type="xs:boolean"
     minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax"
     minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!--
     CONFERENCE MEDIA TYPE
  -->
  <xs:complexType name="conference-media-type">
   <xs:sequence>
    <xs:element name="entry" type="conference-medium-type"
     maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!--
     CONFERENCE MEDIUM TYPE
  -->
  <xs:complexType name="conference-medium-type">
   <xs:sequence>
    <xs:element name="display-text" type="xs:string"
     minOccurs="0"/>
    <xs:element name="type" type="xs:string"/>
    <xs:element name="status" type="media-status-type"
     minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax"
     minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="label" type="xs:string"
    use="required"/>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!--
     URIs TYPE
  -->
  <xs:complexType name="uris-type">
   <xs:sequence>
    <xs:element name="entry" type="uri-type"
     maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="state" type="state-type"
    use="optional" default="full"/>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!--
```

```
    URI TYPE
-->
<xs:complexType name="uri-type">
 <xs:sequence>
  <xs:element name="uri" type="xs:anyURI"/>
  <xs:element name="display-text" type="xs:string"
   minOccurs="0"/>
  <xs:element name="purpose" type="xs:string"
   minOccurs="0"/>
  <xs:element name="modified" type="execution-type"
   minOccurs="0"/>
  <xs:any namespace="##other" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<!--
    KEYWORDS TYPE
-->
<xs:simpleType name="keywords-type">
 <xs:list itemType="xs:string"/>
</xs:simpleType>
<!--
    USERS TYPE
-->
<xs:complexType name="users-type">
 <xs:sequence>
  <xs:element name="user" type="user-type"
   minOccurs="0" maxOccurs="unbounded"/>
  <xs:any namespace="##other" processContents="lax"
   minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="state" type="state-type"
  use="optional" default="full"/>
 <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
<!--
    USER TYPE
-->
<xs:complexType name="user-type">
 <xs:sequence>
  <xs:element name="display-text" type="xs:string"
   minOccurs="0"/>
  <xs:element name="associated-aors" type="uris-type"
   minOccurs="0"/>
  <xs:element name="roles" type="user-roles-type"
   minOccurs="0"/>
  <xs:element name="languages" type="user-languages-type"
   minOccurs="0"/>
```

```
    <xs:element name="cascaded-focus" type="xs:anyURI"
     minOccurs="0"/>
    <xs:element name="endpoint" type="endpoint-type"
     minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax"
     minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="entity" type="xs:anyURI"/>
   <xs:attribute name="state" type="state-type"
    use="optional" default="full"/>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!--
     USER ROLES TYPE
  -->
  <xs:complexType name="user-roles-type">
   <xs:sequence>
    <xs:element name="entry" type="xs:string"
     maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!--
     USER LANGUAGES TYPE
  -->
  <xs:simpleType name="user-languages-type">
   <xs:list itemType="xs:language"/>
  </xs:simpleType>
  <!--
     ENDPOINT TYPE
  -->
  <xs:complexType name="endpoint-type">
   <xs:sequence>
    <xs:element name="display-text" type="xs:string"
     minOccurs="0"/>
    <xs:element name="referred" type="execution-type"
     minOccurs="0"/>
    <xs:element name="status" type="endpoint-status-type"
     minOccurs="0"/>
    <xs:element name="joining-method" type="joining-type"
     minOccurs="0"/>
    <xs:element name="joining-info"
     type="execution-type"
     minOccurs="0"/>
    <xs:element name="disconnection-method"
     type="disconnection-type"
     minOccurs="0"/>
    <xs:element name="disconnection-info"
     type="execution-type"
```

```
    minOccurs="0"/>
   <xs:element name="media" type="media-type"
    minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="call-info" type="call-type"
    minOccurs="0"/>
   <xs:any namespace="##other" processContents="lax"
    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="entity" type="xs:string"/>
  <xs:attribute name="state" type="state-type"
   use="optional" default="full"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
 </xs:complexType>
 <!--
    ENDPOINT STATUS TYPE
 -->
 <xs:simpleType name="endpoint-status-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="pending"/>
   <xs:enumeration value="dialing-out"/>
   <xs:enumeration value="dialing-in"/>
   <xs:enumeration value="alerting"/>
   <xs:enumeration value="on-hold"/>
   <xs:enumeration value="connected"/>
   <xs:enumeration value="muted-via-focus"/>
   <xs:enumeration value="disconnecting"/>
   <xs:enumeration value="disconnected"/>
  </xs:restriction>
 </xs:simpleType>
 <!--
    JOINING TYPE
 -->
 <xs:simpleType name="joining-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="dialed-in"/>
   <xs:enumeration value="dialed-out"/>
   <xs:enumeration value="focus-owner"/>
  </xs:restriction>
 </xs:simpleType>
 <!--
    DISCONNECTION TYPE
 -->
 <xs:simpleType name="disconnection-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="departed"/>
   <xs:enumeration value="booted"/>
   <xs:enumeration value="failed"/>
   <xs:enumeration value="busy"/>
  </xs:restriction>
```

```
    </xs:simpleType>
    <!--
       EXECUTION TYPE
    -->
    <xs:complexType name="execution-type">
     <xs:sequence>
      <xs:element name="when" type="xs:dateTime"
       minOccurs="0"/>
      <xs:element name="reason" type="xs:string"
       minOccurs="0"/>
      <xs:element name="by" type="xs:anyURI"
       minOccurs="0"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
   <!--
       CALL TYPE
    -->
    <xs:complexType name="call-type">
     <xs:choice>
      <xs:element name="sip" type="sip-dialog-id-type"/>
      <xs:any namespace="##other" processContents="lax"
       minOccurs="0" maxOccurs="unbounded"/>
     </xs:choice>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <!--
       SIP DIALOG ID TYPE
    -->
    <xs:complexType name="sip-dialog-id-type">
     <xs:sequence>
      <xs:element name="display-text" type="xs:string"
       minOccurs="0"/>
      <xs:element name="call-id" type="xs:string"/>
      <xs:element name="from-tag" type="xs:string"/>
      <xs:element name="to-tag" type="xs:string"/>
      <xs:any namespace="##other" processContents="lax"
       minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <!--
       MEDIA TYPE
    -->
    <xs:complexType name="media-type">
     <xs:sequence>
      <xs:element name="display-text" type="xs:string"
       minOccurs="0"/>
      <xs:element name="type" type="xs:string"
```

```
        minOccurs="0"/>
       <xs:element name="label" type="xs:string"
        minOccurs="0"/>
       <xs:element name="src-id" type="xs:string"
        minOccurs="0"/>
       <xs:element name="status" type="media-status-type"
        minOccurs="0"/>
       <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string"
       use="required"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
     </xs:complexType>
     <!--
        MEDIA STATUS TYPE
     -->
     <xs:simpleType name="media-status-type">
      <xs:restriction base="xs:string">
       <xs:enumeration value="recvonly"/>
       <xs:enumeration value="sendonly"/>
       <xs:enumeration value="sendrecv"/>
       <xs:enumeration value="inactive"/>
      </xs:restriction>
     </xs:simpleType>
      <!--
        SIDEBARS BY VAL TYPE
      -->
      <xs:complexType name="sidebars-by-val-type">
       <xs:sequence>
        <xs:element name="entry" type="conference-type"
         minOccurs="0" maxOccurs="unbounded"/>
       </xs:sequence>
       <xs:attribute name="state" type="state-type"
        use="optional" default="full"/>
       <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:complexType>
     </xs:schema>

</xs:schema>
```

11 Acknowledgements

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