XCON WG
Internet-Draft
Expires: August 9, 2004

C. Jennings
Cisco Systems
B. Rosen
Marconi

February 9, 2004

Media Mixer Control for XCON
draft-jennings-xcon-media-control-00

Status of this Memo

Copyright Notice

Abstract

Conference mixers have many controls that change how the media is combined for each participant in the conference. There is a need to describe these to the clients connected to the a centralized conference so that the clients can render a user interface and allow the user to manipulate them.

This work is very early and far from complete. This draft sketched the outline of a solution for consideration. It is being discussed on the xcon@ietf.org mailing list.

Table of Contents

1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [1].

2. Introduction to the Problem

This work tries to solve the problem of allowing a conference participant to manipulate the media flow in a mixer. It defines a protocol between the end user's software manipulating the conference and the centralized conference mixer. This needs to be rich enough for a mixer to express what information it wants from a mixer yet simple enough to allow the client to render a useful user interface to the user. This work takes into account that real mixers have constraints on what media flows are possible and that UIs have buttons, knobs, etc that users manipulate. The goal is for a conferencing end point made by one vendor to work with mixers or conference systems made by another vendor.

2.1 Non Problems

There are several topics that are completely internal to the conference systems and are out of scope for this this work. These include:

How the focus manipulates the mixer.

How one describes what a mixer is capable of doing.

3. Overview

When a conference is created, it is instantiated from a template. The template describes what controls are available for the client to manipulate the media. The conference also describes roles that the client can take on, such as Moderator. The template can have parameters that are set when it is instantiated to allow one template to describe variations of similar flow models.

This document describes the templates and ways for the client to understand and manipulate the media in the conference. It allows for the following:

A conference consists of several participants and multiple streams of media flowing between the participant and the mixer.

Sidebars are mini conferences that are just like conferences

except that a sidebar cannot itself contain sidebars.

Clients can discover the template chosen for use in a conference, and the Values of the parameters set for the conference

Clients can discover the available streams in a conference.

Clients can send media on a participant stream and receive media and receive media on a mixer stream.

Clients can discover the Participants in a conference and their role (this is more conference policy than media policy).

Clients can join a conference as a participant and assume a particular role.

Conferences, Streams, and Participants can have controls that manipulate the media sent and received.

The role of the participant will control what view of the conference they have and which media streams they can manipulate.

3.1 Semantic information in a Conference

The conference has a list of Participants. Each Participant has a set of Controls That he can manipulate. Each conference has a list of sidebars. Each conference has a list of Streams. Each Stream has attributes such as name, type, priority and list of contributing participants.

3.2 The Protocol

The protocol between the client and the conference server allows the client to get the semantic information in the conference, find out when it changes, and make changes to it. It's probably something like XCAP. [TODO add ref]

3.3 Templates

Templates define a model for the reception, manipulation and transmission of streams. A template provides enough information that the client can intelligently render a useful GUI to the end user to manipulate the model. There is a registry of well known templates, but a conference server can define new ones. A convener can find all the templates a conference server supports and select one to use when creating the conference.

A template for a very basic audio conference, for example, may indicate that there is one audio stream for each participant, and one output mixer stream named "primary". Each participant in the stream has a single binary control called "Mute". There is only one Role that can be used, called "participant".

3.4 Parameters

Parameters are variables in the template that are set when the conference is created. For example, in the audio conference, the maximum number of participants might be a parameter. If the value was set to 10 when the conference is instantiated, then up to 10 participant streams can be accepted into the mixer. The template can indicate the valid range for max number of participants, perhaps from 2 to 128.

3.5 Controls

Controls are variables participants may manipulate to control the media streams of the conference. Conferences can have controls, participants in a conference can have controls, and streams in a conference can have controls. Controls can also be implicitly created by stream action, for example a selector control based on the loudest speaker. Controls have a name, and a value. Controls are defined in the template.

3.6 Roles

Participants in a conference can take on different Roles that change what ccontrols they may manipulate. The template defines what Roles are available for the client. The moderator (which itself is a role) can change the role of a particular participant.

4. Introductory Example

4.1 Simple Audio

The client selects the basic audio template that looks like:

```
<template name="basic-audio">
     <parameter type="integer" name="max-participants"
             min="2" max="128"/>
     <role name="Participant">
         <stream type="audio" dir="in" name="input[]" priority="1.0">
         </stream>
         <stream type="audio" dir="out" name="mix[]" priority="1.0">
             <control type="boolean" name="mute"/>
         </stream>
```

```
     </role>
</template>
```

The client retrieves this template and uses it to create a conference where it sets the max-participants to 10. Alice and Bob join this conference and the conference server tells Bob about the state of the conference media. There is only one role "participamt". Each participant contributes one input stream. There is also an output stream per participant. There is a single control, called mute, for each participant.

After Alice and Bob have joined, the conference server informs Bob that the current state of the conference is as shown in the xml below.

```
<conference type="basic-audio" name="Weekly Conference">
        <parameter type="integer" name="max-participants"> 10
     </parameter>
        <role name="Participant"/>
        <participant name="Alice" role="Participant">
        <stream type="audio" dir="in" name="input[Alice]"
             url="sip:alice22-audio-primary@cs.example.com"
             priority="1.0"/>
        <stream type="audio" dir="out" name="mix[Alice]"
             url="sip:alice22-audio-primary@cs.example.com"
             priority="1.0">
                <control type="boolean" name="mute"
                     perm="readonly"> 0 </control>
            </stream>
        </participant>
        <participant name="Bob" role="Participant">
        <stream type="audio" dir="in" name="input[Bob]"
             url="sip:bob5-audio-primary@cs.example.com"
             priority="1.0"/>
        <stream type="audio" dir="out" name="mix[Bob]"
             url="sip:bob5-audio-primary@cs.example.com"
             priority="1.0">
                <control type="boolean" name="mute"
                     perm="readwrite"> 0 </control>
            </stream>
        </participant>
</conference>
```

There are two participants, Alice and Bob, who both contribute input streams and receive Mix streams and neither is muted.

Bob's client decides to change the Mute state for its audio stream and sends the following to the conference server to change the state

of the conference.

```
<conference type="basic-audio" name="Weekly Conference">
        <stream type="audio" dir = "out" name="mix[Bob]">
                 <control type="Boolean" name="mute"> 1
                 </control>
        </stream>
</conference>
```

A key part of this is that Bob's client may have known about this basic audio template and what the semantics of the "mute" control implied. The client may have connected this up with a button of the client's that was labeled mute. On the other hand, Bob's client may not have known anything about this template and simply rendered a button on the screen and labeled it "mute" with no idea what this would do. A third client may not have been table to deal with the control at all and may have just ignored it. Clearly the user interface can be better if the client understands the semantics of what the template means, but the user interface is still functional when the client does not.

5. Names and terminology

5.1 Templates

Templates contain a list of stream, roles for participants, parameters that need to be set, and controls for the conference.

5.2 Participants

Participants are the logical user entities participating in a conference.

5.3 Streams

The stream is a named stream of media. An example is a simple audio conference with 6 participants and a mixer that mixes the loudest three. Each participant contributes an input stream. There is a single logical output stream, but every participant gets a "custom" version of this stream, because, in normal mixers, each participants can hear all inputs except his own. This is commonly referred to as "mix-minus". If the output steam also has a control (mute), the output streams for each participant may also vary depending on the state of the control.

Streams all have a type, a name, a direction (in or out), one or more URLs, and a priority. The URL is the source or sink of the stream. The priority indicates how important this particular stream is to the

conference and the type indicates the type of media carried in this steam.

Streams have types. These correspond to the major MIME types of the media they send.

5.3.1 Stream Types

5.3.1.1 Audio

Streams originate as participant contributions (dir="in") that are mixed using some kind of algorithm. Intermediate streams may be created, which are subsequently mixed with other streams yielding streams which are sent to participants (dir="out"). Controls commonly available on audio streams include input or output faders (volume controls), stereo balance, and mute.

5.3.1.2 Video

Streams originate as participant contributions (dir="in") that are combined with some kind of algorithm. Intermediate streams may be created, which are subsequently combined with other streams yielding streams which are sent to participants (dir="out"). Controls commonly available on video streams might include selectors for choosing a tiling format, selectors which input streams appear on output tiles, and video mutes.

5.3.1.3 Text

Streams originate as participant contributions (dir="in") (Instant Messages). Messages from all participants are combined using some algorithm. Intermediate streams may be created, which are subsequently combined with other text streams yielding streams which are sent to participants (dir="out").

5.3.1.4 Application

At a minimal level, this consist of a URL that defines the application. Many systems will simply update an http URL that fetches an HTML page that shows the current presentation.

5.3.2 Stream URLs

Streams have URLs that specify the source or sink of the stream. These would typically be a SIP, H323 or XMPP URL.

5.3.3 Stream Priority

Streams have a priority from 0 to 1. Zero indicates that a client, by default, should not play/display this stream unless the user specifically requests it. A priority of 1 indicates that, by default, the client should render this stream and should warn the user if it cannot. Other values only define an ordering, and clients should attempt to use their resources to display the higher priority streams before the lower.

5.4 Roles

Roles are defined as part of Conference Policy but are used here so that the Media Policy can define separate streams and controls depending on role. Roles are defined by in the template. Some templates may allow a participant to take on more than one role at a time. Each template must define a role named "participant", which is the default role. "Moderator" is a typical role, as is "Floor-Holder", but templates do not intrinsically define or require such roles.

5.5 Controls

Controls manipulate the state of the conference while it is instantiated. All controls have a name, a type, a current value and permissions that indicate whether or not the current client can modify them. They may also have, optionally, a min and max value.

A control can be defined as being part of a role. In that case, all participants who assume that role have an instance of the control. A control may also be defined as part of a stream, in which case all contributors of that stream (dir="in") have an instance of the control, or all sinks of the stream (dir="out") have an instance of the control. There can be global controls, which are available to all participants. Implicit controls extract values from streams (or other controls), such as choosing video inputs based on loudest speakers

5.6 Parameters

Parameters are variables that modify the function of the template. They are fixed when the conference is instantiated. Parameters allow a single template definition to describe a range of possible mixer capabilities.

Parameters have a name, a type, a value and, optionally, a mix and max value.

6. Solution

6.1 Templates

A template is an xml document. The template definition includes a name, which is a string, for example:

```
<template name="audio-basic">
```

6.1.1 Parameters

The parameters in the templates customize a generic template for a specific conference. Parameters have name, type, value, and optionally min/max. Parameters are defined in the template description. Only conveners can set template parameters

One typical template parameter is "max-participants". When the CS generates the template for the client, it can customize the min and max value of this parameter to match what it is capable of. When the client instantiates the template and creates the conference, it can specify the value that has been requested. The value typically represents the limits the mixer is capable of. Resource availability may limit the actual value that can be achieved.

Parameter names are strings.

Parameter Types:

Integer

Real

Enumeration

String

Values of course must be conformant to the type. Min and Max, if defined, must also be conformant to the declared type.

Example:

```
<parameter name="Master Volume", type="integer", min="0",
max="100">75</parameter>
```

6.1.2 Roles

Templates define all the Roles that a participant can take and (optionally) the max number of participants of each role. Each role is defined in a role element. A Role element includes a name and optionally a "max-participants" value. Role elements may also

contain stream elements, which define per-participant-in-role streams.

Example:

```
<role name="moderator" max-participants="1" />
```

6.1.3 Streams

Templates also define all the streams available. A stream element has a name, a type, a direction ("in" or "out"), priority and URL. Certain streams may actually be a set of streams, for example, one per participant. A specific member of the set can be referenced using an array notation with square brackets. For example, if an input stream is available named foo, and there is a participant named "Bob", then foo["Bob"] would be the name of the foo stream Bob contributes. If a stream is defined within a role element, the stream is a set of streams, one per participant in the role. If a stream is defined in more than one role with the same name, the stream set is the same, and participants in any roles that have that stream defined with that name contribute/sink a stream to the set.

The URL is typically not given a value in the template definition. The mixer assigns URL values as participants assume roles. Most implementations would not allow the URL to be changed by the media policy mechanism. The value of the URL would be included in the media policy conference state document.

Example:

```
<stream name="input-audio" type="audio" dir="in" priority="1.0" >
```

6.1.4 Controls

A control can be inside the template, participant, or stream. The control will apply to the appropriate context. By including stream definitions in multiple roles that have the same name, different controls can be provided to different roles affecting streams contributed or sunk from multiple roles. For example, a moderator may be given a set of input volume controls controlling a mix, and every participant can be given an output master mix control for the output stream sent to him

6.1.5 Conference State

Conference state can be requested by any participant. A document will be returned elucidating the complete current conference state, which would contain all the participants, all the streams, and the

values of all the controls. The form of the document mirrors the template definition. The conference can also contain sidebars.

6.1.5.1 Conference State Update

The client can attempt to change the state of various controls in the CS by sending a document that contains just the things it wants to change.

6.1.5.2 Change Notification

The client can request that conference state be automatically sent when it changes.

6.1.6 Transport Protocol

TODO: Need to define how the information is sent between the client and the conference server. XCAP?

6.2 Controls

6.2.1 Requirements

Controls need to collect information. This can be classified into several types. It should be possible to provide default values, a name for the control and text it displays, help text, control if a value is required, and control of whether or not the value is editable. It should be possible to express constraints on the form an input can take by specifying a minimum or maximum for types where that makes sense, or specifying a regular expression that must be satisfied. For numeric values in a constrained range, it should be possible to provide an increment value used by the control. For strings it should be possible to indicate that they should not be displayed when they are entered for things like passwords. Need the ability to internationalize any text that is displayed to the user.

There are control types for:

Strings

Multi-line Strings

Integer

Real

Boolean

Date

Time

Date Time

URI

File Selection

Select Single

Select Multiple

If an unknown control is encountered, it should be treated as a string type. The <label> element controls what is displayed to the user and the <value> element contains the current setting of the control. If set in the template definition, it represents the default value. An optional <description> element provides some text that can be used as help text for the control.

6.2.2 Strings

This is typically rendered as a text input field.

```
<control type="string" name="Host" private="true" >
   <label> Meeting Host </label>
   <value>Richard</value>
   <description>Host for this weeks meeting</description>
   <regex>.*[rR].*</regex>
</control>
```

The "private" attribute indicates that the string should not be displayed as it is entered.

6.2.3 Integer

This can be rendered as a slider or volume knob if it has a constrained range; otherwise it is a text field. The text field may have increment or decrement buttons.

```
        <control type="integer" name="gain">
                <label> Volume </label>
                <value>0</value>
                <range min="-18" max="6" increment="3"/>
        </control>
```

6.2.4 Boolean

This is typically rendered as a toggle button.

```
        <control type="boolean" name="mute">
                <label> Mute </label>
                <value>True</value>
        </control>
```

6.2.5 Selection

This is typically rendered as a pull down menu or as a radio button box.

```
        <control type="select1" name="foo">
                <label> the thing </label>
                <value>2</value>
                <item>
                        <label>one</label>
                        <value>1</value>
                </item>
                <item>
                        <label>two</label>
                        <value>2</value>
                </item>
        </control>
```

The list of items that can be selected is contained in <item> elements. Each item has a label that is displayed and a value that is returned when it is selected.

6.2.6 Multiple Selection

This is typically rendered as a combo box or list.

This is the same as a selection, except that the type is selected and the initial value is a space-separated list of values.

6.2.7 Frame

Provides a hint to groups of controls. Uis are NOT constrained to follow the frame construct.

```
<frame name="Address">
   <control type="string" name="addr" private="true" >
     <label> Street Address </label>
     <regex>.*[rR].*</regex>
```

```
      </control>
      <control type="string" name="city" private="true" >
        <label> City </label>
        <regex>.*[rR].*</regex>
      </control>
      <control type="string" name="state" private="true" >
        <label> State </label>
        <regex>.*[rR].*</regex>
      </control>
   </frame>
```

7. Examples

7.1 Audio Video Presentation

The following is a more complex template with bits of text explaining it:

In this template, there are three roles, Participant, Presenter and Moderator. There is an input and output audio stream for each role. All roles have mute and master volume controls for their outputs. The moderator has input controls for each input.

The video presentation is "Hollywood Squares". Each role contributes one video stream. The moderator can select the presentation (tile format) from an enumeration. All viewers see the same output presentation.

Finally, there is an application sharing channel which is provided by the Presenter, and received by all roles.

```
   <template name="audio-video-presentation">
      <parameter type="integer" name="max-participants"
                  min="2" max="16"/>
      <role name="Participant">
            <parameter type="integer" name="max-participants"
                          min="0" max="16"/>
            <stream type="audio" name="AudioIn" dir="in"/>
            <stream type="video" name="VideoIn" dir="in"/>
            <stream type="video" name="VideoOut" dir="out"/>
            <stream type="application" name="AppShare" dir="out"/>
            <stream type="audio" name="MixOut" dir="out">
                    <control type="binary" name="mute"/>
                    <control type="real" name="master"
                        min="-18" max="+6" note="Master Gain in DB"/>
            </stream>
      </role>
```

```
      <role name="Presenter">
            <parameter type="integer" name="max-participants"
                          min="0" max="16"/>
            <stream type="audio" name="AudioIn" dir="in"/>
            <stream type="video" name="VideoIn" dir="in"/>
            <stream type="video" name="VideoOut" dir="out"/>
            <stream type="application" name="AppShareIn" dir="in"/>
            <stream type="application" name="AppShare" dir="out"/>
            <stream type="audio" name="MixOut" dir="out">
                    <control type="binary" name="mute"/>
                    <control type="real" name="master"
                        min="-18" max="+6" note="Master Gain in DB"/>
            </stream>
      </role>
      <role name="Moderator">
            <parameter type="integer" name="max-participants"
                          min="0" max="1"/>
            <stream type="audio" name="AudioIn" dir="in">
                    <control type="real" name="gain"
                        min="-18" max="+6" note="Input Gain in DB"/>
            </stream>
            <stream type="video" name="VideoIn" dir="in"/>
            <stream type="video" name="VideoOut" dir="out">
                    <control type=selector name="Tile Format">
                          <item name="1x1" value="0"/>
                          <item name="2x1" value="1"/>
                          <item name="2x2" value="2"/>
                          <item name="3x3" value="3"/>
                          <item name="4x4" value="4"/>
                    </control>
            <stream type="application" name="AppShare" dir="out"/>
            <stream type="audio" name="MixOut" dir="out">
                    <control type="binary" name="mute"/>
                    <control type="real" name="master"
                        min="-18" max="+6" note="Master Gain in DB"/>
            </stream>
      </role>
   </template>
```

8. Template Registry

An IANA registry will be created for commonly encountered template definitions. This document will include some starter templates

[Still need TODO this].

9. Comparison to other solutions

[TODO]

10. CPCP vs. MPCP vs. CCP vs. MCP

What is the boundary between conference control, media control, and policy control for both of them.

11. IANA

12. Security

13. Acknowledgments

Many thanks to Nermeen Ishmail and Rohan Mahy

Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Informative References

[3] Mahy, R. and N. Ismail, "Media Policy Manipulation in the Conference Policy Control Protocol", draft-mahy-xcon-media-policy-control-00 (work in progress), June 2003.

[4] Even, R., "Conferencing Scenarios", draft-even-xcon-conference-scenarios-00 (work in progress), June 2003.

[5] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol", draft-ietf-sipping-conferencing-framework-00 (work in progress), May 2003.

Authors' Addresses

Cullen Jennings
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421 9990
EMail: fluffy@cisco.com

Brian Rosen
Marconi
2000 Marconi Drive
Warrendale, PA 15086
USA

Phone: +1 724 742 6826
EMail: brian.rosen@marconi.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.