XCON WG
Internet-Draft
Expires: January 10, 2005

C. Jennings
Cisco Systems
B. Rosen
Marconi
July 12, 2004

Media Conference Server Control for XCON
draft-jennings-xcon-media-control-01

Status of this Memo

Copyright Notice

Abstract

Media Policy is the mechanism by which a client manipulates the media streams of a conference, within limits established by the convener of the conference and the conference server the conference is established on. This document describes how media policy is realized, using a combination of predefined templates a convener can select from that specify interactive controls clients can manipulate

and flow graphs that allow a customized template to be dynamically defined by the convener.

This work 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. TODO Items

Note - the issue of switching from presenter mode to Q and A mode (etc.) is essentially one of floor control? Need much more on how MPCP and floor control work.

Note - using panel for now - may later replace with media neutral term such as placement

3. Introduction

This work tries to solve the problem of allowing a conference participant to manipulate the media flow in a conference server. It defines a protocol between the centralized conference server and the end user's software that manipulates the conference. This protocol needs to be rich enough for a conference server to express what information it wants, yet simple enough to allow the client to render a useful user interface. This work takes into account that real conference servers 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 conference servers or conference systems made by other vendors.

A convener wishing to create a conference uses CPCP (or some other means) to create a conference and obtain a Conference URI. The conference convener can then query the server to find out the media capabilities of that server. This includes the set of templates that a server supports, and the limits the server imposes on the parameters for these templates. A template defines a type of conferencing service that a conference server can provide. The template includes what media streams can flow in and out of the conference, the roles that are possible in the conference and what controls a client can manipulate on the conference to affect the media. A set of standardized templates that a server may support is defined [REFERENCE]. In addition, conference servers that support flow graphs work in TODO REF can dynamically define new templates. Note that templates contain media specific information, so to know which templates are supported is also to know what media types (audio, video, text) are supported.

Each template can define parameters, whose values must be specified by the convener. Parameters limit what media manipulation the

conference can provide over a more general range that the template might define. Parameters are fixed when the conference is instantiated and can not be changed after that. The conference server can limit allowed values for parameters, to, for example, enforce limits of physical hardware, or policy of the conference server. Parameters primarily reduce the number of templates needed.

The conference convener can then choose a template, populate the parameter values and upload using CPCP to the server. If the chosen parameter values are acceptable to the server, the update is accepted and the media policy instantiated. If not, an error message indicating the failure is returned.

Templates define the available roles a participant might assume. The simplest template will have just a single role: Participant. By default, each participant will join a conference as a Participant. More interesting templates will have multiple roles. For example, a template might have two roles: Presenter and Participant. A template role definition will indicate if there can be more than one participant having that role. For this example, there may be only one Presenter but there can be many Participants.

The conference convener can assign roles to participants. This can be done in advance of the conference or dynamically during the conference. For example, the conference convener can assign the role of Lecturer in advance if it is known. When this participant joins the conference, they will be automatically assigned the role of Lecturer. This is Conference Policy not Media Policy but it does relate to the templates. Roles may also be changed by other mechanisms. For example, the floor control mechanism might change a participant's role from Participant to Speaker. The convener may limit the roles a participant may assume. The conference package TODO REF includes the Role of each participant in the conference.

Once a conference starts, a participant can use CPCP to learn the media policy template in use and the parameter values for it. The downloaded object contains the controls available to the participant in the role he is presently in. This template may have controls that allow a participant to control their view of/input to the conference. These controls may be rendered by the participant's endpoint, and any changes to the controls result in MPCP commands being sent to the conference server. A template may define different controls for different roles. For example, a Participant may have only a very small set of controls, a Presenter a larger set. If a participant's role changes during a conference, their set of controls may change, and the user interface will need to be updated accordingly. Controls defined with a type (integer, string, Boolean, ..) and can have constraints on their values. The endpoint (or the conference server)

can change the current value of the controls. Controls can be defined as an array of similar controls.

An advanced conference server may support the definition of custom templates using flow graphs. If so, the conference policy will indicate this capability. If it is supported, a conference convener may upload a flow graph using CPCP. This flow graph will contain enough information for the conference server to create a custom template: it will contain stream level media mixing information and information about parameters, roles, controls, and support for floor control. If the server can process the flow graph and support the mixing defined by the template, the server returns a success response. If it is not able, it returns an error indicating how the flow graph might be fixed. A custom template created using flow graphs will be identical to the set of standardized templates - it will just have a different name, roles, parameters, controls, etc. The same methods that allow a participant to render an unknown standardized template will be used to render a custom template.

Once a conference begins, the template and parameters are fixed and MAY NOT be manipulated during the conference. As a result, flow graphs can only be uploaded prior to the start of the conference, although they could be downloaded by a participant during a conference using CPCP. In general, however, flow graphs will only be used by the convener of the conference prior to the start of the conference.

Templates define media streams. Streams may have controls, such as gain, associated with them. The conference has a set of physical streams that get contributed to the conference by the client and a set of physical streams that are sent from the conference to the client.

Output streams may be formed by the mixer by combining multiple input component streams. For video the output is often some composited form of the input component streams. Similarly for audio multiple mono audio streams are additively summed into an output stream. Mixers can create individual streams for each participant (for example, a mix-minus audio mix wher each participant gets the sum of all inputs except his own), or a single stream sent to all participants (for example, a simple video mixed shows the same composite view of participants to each participant).

4. Non Problems

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

How the focus manipulates the conference server.
How one describes what a conference server is capable of doing.
Managing resource allocation and how busy a given DSP is, and
checking whether more work can be allocated to a media processor.

5. Defining a Template

5.1 Overview

A template defines 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.

Templates contain parameters, a list of streams, roles for participants, and controls for the conference.

A template for a very basic audio conference, for example, may indicate that there is one audio stream for each participant named "mainIn", and one output stream group named "mainOut". Each participant in the stream has a single binary control called "Mute" applied to his input stream. There is only one Role that can be used, called "Participant".

This template definition would look like

```
<template name=basicAudio>
  <role name=Participant min=1 max=16>
    <stream name=mainIn dir=in type=audio>
      <control name=mute type=boolean value=false/>
    </stream>
    <stream name=mainOut dir=out type=audio/>
  </role>
</template>
```

Templates and parameter values are specified by the convener and cannot be changed after the conference starts. Some conference servers may limit changes to templates and parameters between the initial creation of the Media Policy and conference start.

Templates are defined in a language that this document describes. The language includes constructs for defining parameters, streams, roles and controls.

5.2 Parameters

TODO - need better name for Parameters. Perhaps Instantiation Values

Parameters are variables in the template that are fixed when the conference is created. Parameters reduce the number of templates required. For example, in the audio conference, whether or not sidebars are supported might be a parameter. The template can indicate the valid range for parameters, which can be further restricted by the conference server to reflect resource limits or local policy. The convener sets the parameters to a specific value when instantiating a conference to limit what capabilities it will use for that conference. Parameter values may be modified by the convener until the conference starts. Once started, the parameter values may not change.

Parameters have a name, a type, a value and, optionally, a min and a max element. The value specified in the template definition is the default value. The value specified in the downloaded template by a participant is the value set by the convener. An example definition of a parameter is:

<parameter name=max-participants type=integer min=1 max=32 value=16/>

Parameters can be used elsewhere in the template when values are needed. For example, a parameter for "max-participants" might be used to define the number of elements in a controlArray. When used this way, the name of the parameter Is prefixed by the "%" character, for example:

< controlArray name=mute[1:%max-participants] type=boolean &gt ;

Parameter Types:
- Boolean
- 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.

A parameter's default value and min/max may be defined in the template definition by an expression that includes other parameters (as variables). Such expressions must evaluate to a fixed value at instantiation time, circular definitions are not permitted. If not defined, min is zero and max is infinite.

5.3 Roles

Participants in a conference can take on multiple different Roles that will change what controls they may manipulate and which media streams they have access to. The template defines what Roles are available for the client. Manipulation of Roles is done in CPCP and is not directly part of MPCP, but the various Roles that are possible are found in the template. Some common roles include:

- Participant
- Presenter
- Moderator
- Observer

A participant may only be in one role at a time.

Roles are defined in Media Policy. Conference policy defines which roles a particular participant may assume, and provide the mechanisms to assume roles. Roles are defined in the template. Each template must define a role named "Participant", which is the default role.

A role defined in a template includes the name of the role, (optionally) max participants, streams defined for the role and controls defined for the role. Role definitions may be nested. When a role is defined as part of another role, the streams and controls defined in the outer role(s) are available to participants in the inner role. It is common, for example, for a Moderator role to also have all of the streams and controls a Participant can have.

A simple example:

```
<template name=simple>
  <parameter name=max-participants min=1 max=32 value=16>
  <role name=Participant>
     <stream name=mainIn dir=in type=audio/>
     <stream name=mainOut dir=out type=audio/>
     <role name=Moderator max=1>
        <controlArray name=mute[1:%maxParticipants] type=Boolean
value=false>
     </role>
  </role>
</template>
```

In this example there are two roles defined, Participant and Moderator. Participants have one input and one output stream. Moderators have the same input and output streams as a Participant would, and in addition has a set of mute controls for each participant.

Roles can be nested to arbitrary depth. Streams and controls defined

before a nested role are inherited by the nested role. Streams and controls defined after a nested role are not inherited.

5.4 Streams

Streams correspond to a given flow of media. They are named and can be manipulated by controls. The conference package is used to understand the relationships between users or participants, dialog or session, and physical streams.

The physical streams are the actual media streams sent and/or received by or on behalf of conference participants. Media streams can be established when conference participants join a conference and are described by the SDP media lines in the offer/answer exchange between the participants and the focus, or the analogous exchange in other protocols (ex: H.245 logical channel establishment). As a participant's role changes, the streams that participant contributes or consumes with the conference may change, necessitating signaling changes with the focus.

Within the template definition, each stream is described by a media type, direction and a name. Initially media types considered include audio, video, and text. Other media types can also be considered in the future. The direction "in" corresponds to streams originating from the conference participants to the conference, and "out" for streams originating from the conference and terminating at the conference participants.

NOTE: Is this REALLY what we want? A server view? Or do we want an endpoint view?

Streams have types. These correspond to the major MIME types of the media the stream carries.

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

Video Streams originate as participant contributions (dir="in"), which are combined with some kind of algorithm that 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.

Text Streams originate as participant contributions (dir="in") in the form of Instant Messages or interactive text streams. Messages from all participants are combined using some algorithm that are sent to participants (dir="out").

NOTE: There is some discussion of separating interactive text from text, having two separate types. Message mode IM, MSRP IM and session mode interactive text are all intended to be supported with this version of MPCP.

Examples of a stream definitions were given above in Section 5.3.

5.5 Controls

Controls are variables in a template that participants may manipulate to control the media streams of the conference. Controls are defined with a type to assist the client render an appropriate user interface. A control definition has a name, a type, a default value, and may have constraints on its values. A control in an instantiated conference has a current value, which may be changed within the limits of the definition. The controls that are available are defined in the template.

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 "controlArray" can be defined as a group of controls with common characteristics.

A control is defined with the following elements

- Name
- label (display name for the control)
- type (integer, real, string, enumeration)
- value - current value (in conference object) or default value in template object mix/max/increment - the minimum and maximum value of the control and the allowable increment for the value. Applies to integer and real types
- regex - an expression limiting the value of a string. Applies to string and multiline string types
- editable - a flag to indicate if the value can be changed
- enable - a flag set dynamically to indicate if the control is currently active
- private - a flag indicating that the value entered should not be displayed, as in a password text box. Applies to strings and multiline strings.
- help - a help text string

There are control types for:

```
string
multiline String
integer
real
boolean
date
time
dateTime
URI
enumeration - choose one from a list of enumerated values
enumerationMultiple - choose one or more from an enumeration
```

If an unknown control is encountered, it should be treated as a string type. If min is not specified, it is negative infinity, and if max is not specified, it is infinity.

enumerated values can be supplied in ordered <item> elements each of which contain label and value elements. The template definition does not need to specify the values of an enumeration; the conference server may supply the values. For example, a channel selector may be an enumeration of the available television channel names.

5.5.1 Strings

This is typically rendered as a text input field.

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

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

5.5.2 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>
                <min>-18</min> <max>6</max> <increment>1</increment>
        </control>
```

5.5.3 Boolean

This is typically rendered as a toggle button.

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

5.5.4 Selection

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

```
<control type=select 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>
```

5.5.5 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.

5.5.6 Control Arrays

A controlArray is a one dimensional set of controls with common characteristics. A controlArray is defined like a simple control with the addition of bounds. For example:

```
<controlArray name=muteInputs[1:16] type=Boolean value=false>
```

When a controlArray is manipulated, a specific element is denoted with an index in square brackets, e.g. <control name=muteInput[7] value=true >.

For convenience, the notation "*" denotes all rows in the control. A parameter might be used as a bounds using %foo notation. ControlArrays cannot be nested (arrays of arrays are not supported). Each member of a controlArray of enumation type has the same labels (i.e. <item> elements apply to all elements of the array). Each element of a controlArray has its own enable.

5.5.7 Frames

A frame element provides a hint to groups of roles, streams and controls. UIs are not constrained to follow the frame construct and MAY ignore it. Frame elements may occur anywhere in a template definition.

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

5.5.8 Lists

It is common to have a list of participants or other entities that often have some kind of implied order. An example would be a list of the most active speakers. Such lists are useful when a label on a control is to be filled in by the conference server. A list may be defined in a template and then subsequently used elsewhere, especially in an item element as part of an enumeration control. Lists have subscripts to select one member (row) of the list.

For example, the "active-speaker[0]" would likely be the current speaker, and "active-speaker[1]" might be the previous speaker. In general, [0] is the most import member or first member of the list if there is an implied ordering.

A list is defined with a name and bounds. Bounds can be declared to be dynamic, by using an "*". In that case, the value changes as the conference state changes. If there was a role called Presenter, and

there could be 1 to 3 Presenters, one could declare a list as <list name=Presenters[*]> If at some time during the conference, the number of presenters was 2, the list would consist of the two participants currently in that role.

When used subsequently in the template, the list name is prefixed by "%". For example:

```
<list name=active-speaker[0:7]/>
...
<control name=videoSelect type=enumeration value=1>
  <item label=%active-speaker[0] value=0>
  <item label=%active-speaker[1] value=1>
       ...
  <item label=%active-speaker[7] value=7>
</control>
```

As a convenience the star notation can be used to simplify such a construction:

```
<list name=active-speaker[0:7]/>
...
<control name=videoSelect type=enumeration value=1>
  <item label=%active-speaker[*] value=*>
</control>
```

The star means "all". This notation can further be extended when it is desired to concatenate lists. For example:

```
<list name=active-speaker[*]/>
<list name=presenters[*]/>
...
<control name=videoSelect type=enumeration value=1>
  <item label=%presenters[*] value=*>
  <item label=%active-speaker[*] value=*>
</control>
```

Suppose, using the above example, the current presenters were Alice and Bob, the current speaker is Carol and the previous speaker was Dave. Then the enumeration choices would be Alice, Bob, Carol and Dave in that order.

After declaring a list, the bounds of the list and number of elements can be used in other definitions. For example, with a list defined as: <list name=foo[0:10]> %foo.mix is 1, %foo.max is 10 and %foo.length is 11.

5.6 Objects

A template is defined in a template object. When a conference server is queried for the templates it supports, it will return a template object. The original definition of a template object is a document that is included in the IANA registration for the template name.

The convener of the conference uses CPCP to create a conference, specifying the template to be used, and the values of the parameters.

A participant joins the conference and can download the conference object which is a version of the template that returns only the role, streams and controls for the role the participant is in. The participant can upload a conference object to change the values of the controls.

5.6.1 Template Object

The template object is defined in a document, which includes the XML descriptions of the parameters, roles, streams and controls as well as a human readable description of how the template works.

The template object is returned by the CS at any time from when the convener first creates the conference until the conference is over. The only difference permitted between the template object in the document and the template object the CS returns are the min and max values of parameters. The CS may not specify a min value less that the document min, and may not specify a max value greater than the document max.

5.6.2 Conference Object

The conference object is the XML object that contains all the states about the instantiated conference. It closely mirrors the XML template for the conference but represents an instantiated version of the template with all the appropriate streams, control values, and other states appropriate for the role of the participant that uploads or download it. It may contain Sidebar Objects.

5.6.3 Sidebar Object

Sidebars have all the properties of a full fledged Conference Object except they must exist inside an Conference Object and can not contain Sidebar objects themselves.

6. Examples

6.1 Simple Audio Example

The client selects the basic audio template that looks like:

```
<template name=BasicAudio>
  <parameter name=maxParticipants min=1 max=16/>
  <role name=Participant>
     <stream name=mainIn dir=in type=audio/>
     <stream name=mainOut dir=out type=audio>
        <control name=gain label="Volume" type=real value=1.0/>
     </stream>
   </role>
</template>
```

The client retrieves this template. This templates defines that this conference has one (required) role, Participant, which has one input stream called mainIn and one output stream called mainOut. There is a single integer control called gain for the output stream.

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

```
<conference template=BasicAudio>
 <role name=Participant>
  <stream name=mainIn dir=in type=audio/>
  <stream name=mainOut dir=out type=audio>
     <control name=gain label="Volume" type=real value=1.0/>
  </stream>
 </role>
</conference>
```

Bob's client decides to change the gain control for its audio stream and sends the following to the conference server to change the state of the conference.

```
<conference template BasicAudio>
 <role name=Participant>
  <stream name=mainOut>
     <control name=gain value=0.75/>
  </stream>
 </role>
</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 "gain" control implied. The client may have connected this up with a knob of the client's that was labeled Volume. On the other hand, Bob's client

may not have known anything about this template and simply rendered a slider on the screen and labeled it "gain" with no idea what this would do. A third client may not have been able 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.

6.2 Simple Audio Video Example

A more complex video example is given below.

```
<template name=AudioVideo>
  <parameter name=max-participants min 1 max 32>
  <list Participants[1:%maxParticipants]>
  <role name=Listner>
    <stream name=audioOut dir=out type=audio>
      <control name=volume type=real value=1.0/>
    </stream>

    <role name=Participant>

      <stream name=videoIn dir=in type=video/>

      <stream name=audioIn dir=in type=audio>
        <control name=mute type=Boolean value=false/>
      </stream>

      <stream name=videoOut dir=out type=video/>
         <conrol name=layout type=enumeration value=1>
            <item label="1x1" value=1/>
            <item label="2x1" value=1/>
            <item label="2x2" value=1/>
            <item label="4x4" value=1/>
         </control>
         <controlArray name=inputSelect[1:16] type=enumeration>
            <item label=%Participants[*] value=*>
         </controlArray>
      </stream>
    </role>
  </role>
</template
```

This template defines two roles, listener and participant. The listener has only one stream, the audio output of the mix, and one control, a volume control for his output. Participants have the same stream and control and in addition contribute an input audio stream to the mixer, on which they have a mute control. Participants also

contribute a video input stream and receive a video output stream. There are two controls associated with the video output stream. One is a selector that chooses a layout (1x1. 2x1, ...). For each of the up to 16 (4x4) tiles, the user can select one of the participants. The selector is labeled with the participant names via the Participant list.

7. 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.

7.1 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.

7.2 Change Notification

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

7.3 Transport Protocol

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

8. Control Declarations

9. Template Registry

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

[Still need TODO this].

10. IANA

TODO - will need template registry

11. Security

TODO

12. Acknowledgments

Many thanks to Nermeen Ishmail, Rohan Mahy, Alan Johnston, Orit Levin, Roni Evin for many helpful comments. Rohan in particular has spent a huge amount of time improving this document. Large chunks of text were contributed by Alan Johnston. Chris Boulton and Umesh Chandra contributed the template appendix

13. References

13.1 Normative References

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

13.2 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

Appendix A. Guidelines for writers of Media Policy Templates

NOTE: This is the beginning of what will eventually be a separate document with the set of predefined templates. It was written by Chris Boulton and Umesh Chandra.

TODO

Appendix B. Templates

B.1 Audio Templates

B.1.1 Basic-Audio Template

B.1.1.1 Description

The Basic-Audio template can be used to achieve the simplest form of audio interaction between multiple voice input streams from participating entities. The output stream will consist of either a mix of the participants input streams excluding the input stream of the receiving entity, or the current active participant.

B.1.1.2 Roles

Participant: The basic audio template specifies the role of 'Participant' to signify an entry level user with no privileges by default. It is defined with the same set of controls as Moderator, but they are disabled by default

Moderator: The basic audio template specifies the role of 'Moderator' to signify a user with advanced privileges by default. The 'Moderator' role for this template has exactly the same controls as a 'Participant'. The major difference being that the default value for the 'enable' attribute on the 'mute' control on an output media stream is set to 'true' and the default value for the 'enable' attribute on the 'mix-type' control is set to 'true'. See the individual control definitions for impact of the attribute settings.

B.1.1.3 Parameters

max-streams: The 'max-streams' parameter specifies the maximum number of entities that are permitted to be involved in an instantiated instance of the template for the specified role. The minimum value permitted is "2" and the maximum value permitted is "128".

B.1.1.4 Controls

B.1.1.4.1 mute

The 'mute' control is used in conjunction with a media stream to cease transmission of associated media. It has a 'Boolean' value. The 'mute control can exist in two forms, depending on if the control appears on input or output media streams. When appearing as a control on an input media stream, the control locally manipulates the clients media stream to be active or inactive. When muting locally, the control consists of the following attributes:-

type: 'Boolean'.

name: 'mute'.

default: 'false'. Setting the 'mute' attribute to 'false' specifies that media should be transported for the associated media stream. When set to the value of 'true', media should not be transported for the associated media stream.

enable: 'true'.

The 'mute' control can also appear on an output stream. This indicates that the mute will manipulate outgoing media streams from the mixer to alternative clients. When used in this instance, the attributes for the control are identical to that of the input definition with the exception that the 'enable' and 'default' values are both set to 'true'. This instance of the control defines an additional 'controlArray' that represents other participating clients. This provides a privileged user the ability to mute/un-mute media being distributed to every client defined.

B.1.1.4.2 mix-type

The 'mix-type' control is used to configure the type of media associated with an output stream. The control can have one of two possible values:

Value "1" - Indicates an output stream of 'MixMinus'. This results in a mix of all output streams with the exception of the receiving

participants input.

Value "2" - Indicates an output stream of 'Active-Speaker'. This results in a mix containing the current active speaker.

B.1.1.5 Streams

The 'Basic Audio Template' consists of two audio streams:

B.1.1.5.1 AudioIn

The 'AudioIn' media stream details properties associated with the incoming audio to the mixer. The 'AudioIn' stream has the following attributes:

type: 'audio'

name: 'AudioIn'

dir: 'in'

B.1.1.5.2 AudioOut

The 'AudioOut' media stream details properties associated with the outgoing audio from the mixer. The 'AudioOut' stream has the following attributes:

type: 'audio'

name: 'AudioOut'

dir: 'out'

B.1.1.6 XML Definition

-

```
<template name="basic-audio">
    <parameter type="integer" name="max-streams" min="1" max="128"/>
        <role name="Participant">

             <stream type="audio" name="AudioIn" dir="in">
               <control  name="mute" default="false" enable="true"/>
             </stream>

             <stream type="audio" name="AudioOut" dir="out">
             <control name="mix-type" type="enumeration", enable="false"
default="1">
               <item label="MixMinus" value="1">
```

```
                  <item label="Active-Speaker" value="2">
                <control name="mute" type="boolean" enable="false"
default="false">
                <controlArray name="sourceSelector" label="source"
type="enumeration" enable="false" default="false">
                      <item label="participant 1" value="1"/>
                      <item label="participant 2" value="2"/>
                      <item label="participant 3" value="3"/>
                      ....
                      ....
                      <item label="participant n" value="n"/>
                     </control>
                </stream>
           </role>

           <role name="moderator">
                <parameter type="integer" name="max-streams" min="1" max="128"/
>

                <stream type="audio" name="AudioIn[]" dir="in">
                 <control type="boolean" name="mute" default="false"
enable="true"/>
                </stream>

                <stream type="audio" name="AudioOut[]" dir="out">
                <control name="mix-type" type="enumeration", enable="true"
default="1">
                  <item label="MixMinus" value="1">
                  <item label="Active-Speaker" value="2">
                <control type="boolean" name="mute" enable="true" default="0">
                <controlArray name="sourceSelector" label="source"
type="enumeration" enable="true" default="false">
                      <item label="participant 1" value="1"/>
                      <item label="participant 2" value="2"/>
                      <item label="participant 3" value="3"/>
                      ....
                      ....
                      <item label="participant n" value="n"/>
                     </control>
                </stream>
           </role>
   </template>
```

B.2 Video Templates

B.2.1 Basic-Video Template

B.2.1.1 Description

The 'Basic-Video Template' is used to convey the basic set of video

functionality. The template allows participants to send and receive video media with various controls that allow for manipulation of output format and other control functionality. The template also defines a moderator role who has the ability to control the functionality available to participants.

B.2.1.2 Roles

Participant: The basic video template specifies the role of 'Participant' to signify an entry level user with no privileges by default. It is defined with the same set of controls as Moderator, but they are disabled by default.

Moderator: The basic video template specifies the role of 'Moderator' to signify a user with advanced privileges by default. The 'Moderator' role for this template has exactly the same controls as a 'Participant'. The major difference being that the default value for the 'enable' attribute on the 'layout' control of the output video stream is set to 'true' which enables a 'Moderator' to force the layout if required.

B.2.1.3 Parameters

max-participants: The 'max-participants' parameter specifies the maximum number of entities that are permitted to be involved in an instantiated instance of the template for the specified template category. The minimum value permitted is "1" and the maximum value permitted is "128".

max-video-streams: The 'max-video-streams' parameter specifies the maximum number of media streams that are permitted to be involved in an instantiated instance of the template. The minimum value permitted is "1" and the maximum value permitted is "128". This parameter is required as participant of the mix can contribute more than one video stream.

max-video-input-streams: The parameter "max-video-stream-from-participant" indicates the number of input video streams each participant can inject into the conference. The convener of the conference can set this value when instantiating a conference. This reduces the requirement to define multiple templates for a number of media stream of same type that participants can send to the conference.

B.2.1.4 Controls

B.2.1.4.1 pause-video

The 'pause-video' control is used in conjunction with a media stream to cease transmission of associated media. It has a 'Boolean' value. The 'pause-video' control consists of the following attributes:-

type: 'Boolean'

name: 'pause-video'

default: 'false. Setting the 'default' attribute to 'false' specifies that media should be transported for the associated media stream. When set to the value of 'true', media should not be transported for the associated media stream.

enable: 'true'

B.2.1.4.2 layout

The 'layout' control is used to specify both the format of outgoing video mix to an entity and the source configuration for the media. It has an 'Enumeration' value. The 'layout' control consists of the following attributes:-

type: 'Enumeration'

name: 'layout'

default: '0' which corresponds to the 1x1 tile format. The other formats available are - '2x1' which has a value of '1', '1x2' which has a value of '2', '2x2' which has a value of '3' and finally '3x3' which has a value of '4'.

enable: 'false' as for the 'participant' role and 'true' for the 'moderator' role.

B.2.1.5 Streams

The 'Basic-Video Template' consists of two video streams:

B.2.1.5.1 VideoIn

The 'VideoIn' media stream details properties associated with the incoming video to the mixer. The 'VideoIn' stream has the following attributes:

type: 'video'

name: 'VideoIn'

dir: 'in'

B.2.1.5.2 VideoOut

The 'VideoOut' media stream details properties associated with the outgoing audio from the mixer. The 'VideoOut' stream has the following attributes:

type: 'video'

name: 'VideoOut'

dir: 'out'

B.2.1.6 XML Definition

-

```
<template name="basic-video">
            <parameter type="integer" name="max-participants" min="1"
max="128"/>
            <parameter type="integer" name="max-video-streams" min="1"
max="128"/>

             <role name="Participant" max=%max-participants min="1">
                 <parameter type="integer" name="max-video-input-streams"
min="1" max="2"/>
                  <stream type="video" name="VideoIn" dir="in">
                      <control name ="pause-video" type="boolean"
enable="true" default="false"/>
                  </stream>
                 <stream type="video" name="VideoOut" dir="out">
                        <control name="Layout" type="enumeration"
enable="false" default="0">
                          <item label="1x1" value="0"/>
                          <item label="2x1" value="1"/>
                          <item label="1x2" value="2"/>
                          <item label="2x2" value="3"/>
                          <item label="3x3" value="4"/>
                    </control>
                    <controlArray name="sourceSelector" label="source"
type="enumeration" enable="false" default="0">
                      <item label="voice-activated switching" value="0"/>
                      <item label="participant 1" value="1"/>
                      <item label="participant 2" value="2"/>
                      ....
                      ....
```

```
                    <item label="participant n" value="n"/>
                    </control>

                </stream>
            </role>
```

```
              <role name="Moderator" min="0" max="1"/>
                  <parameter type="integer" name="max-video-input-streams"
min="1" max="2"/>
                  <stream type="video" name="VideoIn" dir="in">
                       <control name ="pause-video" type="boolean"
default="false" enable="true" />
                   </stream>
                  <stream type="video" name="VideoOut" dir="out">
                         <control name="Layout" type="enumeration"
enable="true" default="0">
                         <item label="1x1" value="0"/>
                         <item label="2x1" value="1"/>
                         <item label="1x2" value="2"/>
                         <item label="2x2" value="3"/>
                         <item label="3x3" value="4"/>
                         </control>
                       <controlArray name="sourceSelector" label="source"
type="enumeration" enable="true" default="0">
                       <item label="voice-activated switching" value="n" />
                       <item label="participant 1" value="0"/>
                       <item label="participant 2" value="1"/>
                       <item label="participant 3" value="2"/>
                       ....
                       ....
                       <item label="participant n" value="n"/>
                       </control>
                  </stream>
   </template>
```

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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