

XCON Working Group
Internet-Draft
Expires: April 26, 2006

C. Boulton
Ubiquity Software Corporation
U. Chandra
Nokia Research Center
October 23, 2005

Media Policy Templates for XCON
draft-boulton-xcon-media-template-02

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 26, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

The xcon framework[1] specifies the object model for centralized conferencing. The Conference Object, which is defined in the framework data model, comprises of two distinct components - the Common Conference Information and Conference Templates. This memo specifies the Conference Templates that describe various common conference scenarios. The templates define controls (and media properties like type of media supported, what streams are supported

etc) through which participants of the conference can manipulate the media they receive from the conference server. This document provides a minimum set of media templates that can be instantiated during conference creation and manipulated during the life cycle of a conference instance.

This draft is currently under major revision and should be considered a work in progress as it aligns with current technical direction of the Working Group. A revision of this draft will be submitted very soon.

This work is being discussed on the xcon@ietf.org mailing list.

Table of Contents

1.	Introduction	4
2.	XML Conventions and Practices	4
3.	Guidelines for writers of Media Policy Templates	5
4.	Templates	5
4.1.	Basic-Audio Template	5
4.1.1.	Description	5
4.1.2.	Roles	5
4.1.3.	Parameters	6
4.1.4.	Controls	6
4.1.5.	Control Arrays	7
4.1.6.	Streams	8
4.1.7.	Blueprint - XML example	9
4.1.8.	XML Schema Definition	11
4.2.	Advanced Audio Template	14
4.3.	Basic-Video Template	14
4.3.1.	Description	14
4.3.2.	Roles	14
4.3.3.	Parameters	15
4.3.4.	Controls	15
4.3.5.	Control Array	16
4.3.6.	Streams	16
4.3.7.	Blueprint - XML Example	17
4.3.8.	XML Schema Definition	19
4.4.	Advanced Video Template	22
4.5.	Basic Text Template	22
4.5.1.	Description	22
4.5.2.	Roles	22
4.5.3.	Parameters	23
4.5.4.	Controls	23
4.5.5.	Control Array	23
4.5.6.	Streams	23
4.5.7.	Blueprint - XML Example	25

4.5.8.	XML Schema Definition	26
4.6.	Advanced Text Template	29
4.7.	Basic Audio-Video Template	29
4.7.1.	Description	29
4.7.2.	Roles	29
4.7.3.	Parameters	30
4.7.4.	Controls	30
4.7.5.	Control Array	30
4.7.6.	Streams	30
4.7.7.	Blueprint - XML Example	31
4.7.8.	XML Schema Definition	33
4.8.	Audio-text Template	37
4.9.	Basic Multimedia Template	37
4.10.	Advanced Multimedia Template	37
5.	Security Considerations	37
6.	IANA Considerations	37
7.	Acknowledgements	37
8.	References	37
8.1.	Normative References	37
8.2.	Informative References	38
	Authors' Addresses	39
	Intellectual Property and Copyright Statements	40

1. Introduction

The XCON architecture defined in [1] is built around a fundamental concept of Conference Object. A Conference Object is a logical representation of a conference instance. For conference creation, a conference object provides a "blueprint" representing the system capabilities which consists of a Common Conference Information part [4] and a Conference Template (as defined in this document). A conference object also provides the logical representation of a conference during each of the various stages of a conference (e.g. reservation, active, completed etc).

[Editors Note: Need explicit reference relating to where the CCI is derived from.]

The mechanism described in the 'Media Conference Server Control[2]' draft defines a protocol between the Centralized Conference System and individual Conference Participants. The mechanism details the construction of media templates that provide a wide range of functionality to a client for the purpose of manipulating a media profile. The templates are constructed using an XML rule set described in [2].

This document supplies a a set of core media templates that should be used in conjunction with [2]. The templates specified SHOULD be supported at a minimum but MAY NOT be supported if a core related feature is not available on the Conference System.

The editors of this document would like to point out that this document is currently 'Work In Progress' and fully identify that more work is required on the text and XML for the next revision. In the current draft revision we have defined only four templates - basic audio, basic video, basic audio & video and basic-text. The intent of defining these templates is to get a complete picture of conference object which comprises of the common conference information which is defined in [4] and the conference template. The idea is to get early feedback from the working group regarding templates which would help in defining the more complex templates for different conference scenarios in the next revision.

2. XML Conventions and Practices

It is intended that future iterations of this document will provide an individual namespace for each template schema definition. The schema will then be registered in an appropriate IANA registry. In the future, when new templates are constructed, they can also be registered.

It is also intended that common components of this draft will be removed and incorporated in a separate namespace. Templates will then import the common components and use appropriately. This will save on repetition of common elements that appear in multiple media templates. This section will be expanded to include all the relevant details of these changes.

3. Guidelines for writers of Media Policy Templates

[Editors Note: TODO]

4. Templates

4.1. Basic-Audio Template

4.1.1. 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 consists mix of all the participants audio excluding the input stream of the receiving entity. If a participant needs to talk in the conference, it first needs to get a floor from the moderator of the conference. The floor control can be enabled or disabled by the moderator during the conference creation stage. When the floor control is not enabled then any participant can speak in the conference at any time. In this conference very basic controls like muting the output stream, changing the volume of the output stream are provided to end participants. If more advanced control (or user defined control) are required, it is recommended to use the advance-audio template.

4.1.2. Roles

Roles define different privileges that are available to the conference participants. In [3] different types of roles that the participants can assume in a conference are defined. For the basic audio template, there can be only two types of roles - participants and moderator.

Issue : Do we limit that there can be only two types of participants or we remove this restriction and let the conference creator to define as many roles as possible and name them as they wish.

4.1.2.1. Participant

The basic audio template specifies the role of 'Participant' to

signify an entry level user with no privileges by default. The participant role could obtain access to all controls available for this template if set accordingly by a privileged user. This is achieved during the blueprint creation phase by setting the enable attribute of the control element to TRUE.

4.1.2.2. 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 the same controls as a 'Participant' with an additional control for muting any participant in a conference instance. The primary difference being that the default value for the 'enable' attribute on the 'mute' control on an output media stream is set to 'true'. See the individual control definitions for the impact of attribute settings.

4.1.3. Parameters

4.1.3.1. max-participants

The 'max-participants' parameter specifies the maximum number of participants (Entity with signaling connection to the focus) that are permitted to be involved in an instantiated instance of the template (or sub-element, depending on where specified) for the specified role. The example Blue-Print provides a minimum value permitted of "1" and the maximum value permitted is "128".

4.1.3.2. max-audio-streams

The 'max-audio-streams' parameter specifies the maximum number of audio media streams that are permitted to be involved in an instantiated instance of the template for the specified role. The example Blue-Print provides a minimum value permitted of "1" and the maximum value permitted is "128".

4.1.4. Controls

In the basic-audio template, only two types of controls are defined - mute and gain. It is expected that for most of the basic audio conferencing, these two controls are sufficient. If the conference server wants to support more advanced control, then it is recommended that advance-audio template to be used. In those templates the schema for controls is extensible, hence new control types can be added.

Issue : For the basic audio template, we are restricting the type of the control to boolean and real (for mute and gain control). The

motivation for that was you cannot define new control types in the basic-audio template. Should we keep the restriction or let the capability to define new control in a basic-audio template.

[4.1.4.1.](#) **mute**

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

type: The 'mute' control has a type of 'Boolean'.

name: The control is named 'mute'.

enable: The enable attribute specifies if access to the control is available to the client. The value is of type 'boolean' and is set to 'true' as default.

The Control also consists of the following additional elements:-

label: Provides a Control label that can be rendered to the client. For the Blue-Print example the vlaue is 'Mute-Audio'.

value: Provides a default value for the Control. For the Blue-Print example the value is 'True'.

[4.1.4.2.](#) **gain**

The 'gain' control is used in conjunction with a media output stream to indicate the amount of amplification of an audio stream. It has a 'Real' number value. The control consists of the following attributes:-

type: The 'gain' control has a type of 'Real'.

name: The control is named 'gain'.

enable: The enable attribute specifies if access to the control is available to the client.

The Control also consists of the following additional elements:-

default: Indicates the default value for the 'gain' control when not set.

[4.1.5.](#) **Control Arrays**

The definition for Templates is derived from [\[2\]](#) which permits for

Control Arrays. The `controlArray` defines a control for an logical stream. In the actual conference object, there would be one instance of control for each of the stream in the logical stream. Logical streams are defined in [2]. With the `controlArray` an control is defined which applies to all the streams which are contributing to the logical stream. The logical stream element has an attribute type which defines the media type of the logical stream. The attribute name in the `Logicalstream` has a semantic meaning. The conference server based on the name of the logical stream knows what streams constitutes the logical stream. For example in the basic audio we have an logical stream called `MainAudioIn`, which comprises of the incoming streams from the participants. We also define another logical stream in basic audio called `MainAudioOut` which comprises of the output stream from the server to the end points.

In the basic-audio template, for the role of the moderator, two `controlArray` elements are defined which work on the two logical streams - `MainAudioIn`, `MainAudioOut`. For both these logical streams, the `controlArray` element defines the control `Mute`, which enables the moderator to mute any participants incoming audio stream or out going audio stream

4.1.6. Streams

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

4.1.6.1. AudioIn

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

type: The stream type is 'audio'.

name: The descriptive name for this stream is 'AudioIn'.

dir: The direction of the stream is set to 'in'.

stream-id : The identifier which uniquely identifies the stream. It is optional because during the blueprint creation phase there is no need to specify the stream-id. But once the conference starts and the streams are created in the server, each stream element is assigned an stream-id, through which the server and the client can uniquely identify the streams.

4.1.6.1.1. Floor

The 'AudioIn' media stream consists of the following floor:

[4.1.6.1.1.1.](#) **Send_audio**

The 'Send-audio' floor is associated with the 'AudioIn' stream. It provides a mechanism to control the input to a media mix. A participant with a role of 'Moderator' should always have floor preference - this is considered out of scope for this document. The following attributes are defined:

floor_id: Uniquely identifies the floor within the context of this template and has a value of '1'.

instances: Value of '1' implies only one occurrence of this floor at any one time.

enable: A value of 'false' indicates that this floor is not enabled by default.

value: A value of false specifies that the instance represented by this template does not have control of the floor.

[4.1.6.2.](#) **AudioOut**

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

type: The stream type is 'audio'.

name: The descriptive name for this stream is 'AudioOut'.

dir: The direction of the stream is set to 'out'.

[4.1.7.](#) **Blueprint - XML example**

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

  <role name="Participant">
    <parameter type="integer" name="max-participants"
value="1" min="1" max="128"/>

    <stream type="audio" name="AudioIn" dir="in">
      <floor name="Send_audio" floor_id="1" instances="1"
enable="true" value="false">
        <control type="boolean" name="mute" enable="true">
```

```
<label> Mute-Audio </label>  
<value>True</value>  
</control>
```

```

    </floor>
  </stream>

  <stream type="audio" name="AudioOut" dir="out">
    <control type="real" name="gain" enable="true" >
      <label>Volume</label>
      <value>0</value>
    </control>
  </stream>
</role>

<role name="moderator">
  <parameter type="integer" name="max-participants"
value="0" min="0" max="128"/>

  <controlArray >
    <LogicalStream type="audio" name="MainAudioIn" />
    <control type="boolean" name="mute" enable="false">
      <label>MuteInputAudio</label>
      <value>False</value>
    </control>
  </controlArray>

  <controlArray>
    <LogicalStream type="audio" name="MainAudioOut" />
    <control name="mute" type="boolean" enable="false">
      <label>MuteOutputAudio</label>
      <value>False</value>
    </control>
  </controlArray>

  <stream type="audio" name="AudioIn" dir="in">
    <floor name="Send_audio" floor_id="1" instances="1"
enable="true" value="true">
      <control type="boolean" name="mute" enable="true">
        <label> Mute-Audio </label>
        <value>True</value>
      </control>
    </floor>
  </stream>

  <stream type="audio" name="AudioOut" dir="out">
    <control type="real" name="gain" enable="true" >
      <label>Volume</label>
      <value>1.0</value>
    </control>
  </stream>
</role>

```

</template>

Boulton & Chandra

Expires April 26, 2006

[Page 10]

[4.1.1.8.](#) XML Schema Definition

This section provides the XML schema definition for the 'Basic-Audio' template.

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:role="urn:ietf:params:xml:ns:role-schema">

    <import namespace="urn:ietf:params:xml:ns:role-schema" />

    <xs:element name="template-basic-audio">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="role" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="parameter">
      <xs:complexType>
        <xs:attribute name="type" type="parameter-defined" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="value" type="parameter-value" use="required"/>
        <xs:attribute name="min" type="xs:positiveInteger" use="optional"/>
        <xs:attribute name="max" type="xs:positiveInteger" use="optional"/>
      </xs:complexType>
    </xs:element>

    <xs:simpleType name="parameter-value">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:integer"/>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:real"/>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:enumeration"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>

    <xs:simpleType name="parameter-defined">
      <xs:restriction base="xs:string">
```



```

        <xs:enumeration value="integer"/>
        <xs:enumeration value="real"/>
        <xs:enumeration value="enumeration"/>
    </xs:restriction>
</xs:simpleType>

<xs:element name="role">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="controlArray" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="stream" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="rtype" use="required"/>
  </xs:complexType>
</xs:element>

  <xs:simpleType name="rtype">
    <xs:restriction base="role:role-type">
      <xs:enumeration value="Participant"/>
      <xs:enumeration value="Moderator"/>
    </xs:restriction>
  </xs:simpleType>
<xs:element name="controlArray">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="LogicalStream" minOccurs="1" maxOccurs="unbounded" />
      <xs:element ref="control" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="stream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="floor" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="control" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="media-type" use="required"/>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="dir" type="direction-type" use="required"/>
    <xs:attribute name="stream-id" type="xs:positiveInteger" use="optional"/
>
  </xs:complexType>
</xs:element>

```

```
<xs:element name="LogicalStream">
```

```
<xs:complexType>
  <xs:attribute name="type" type="media-type" />
  <xs:attribute name="name" type="logical-Streamtype" />
</xs:complexType>
</xs:element>

<xs:simpleType name="logical-streamType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="MainAudioIn"/>
    <xs:enumeration value="MainAudioOut"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="media-type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="audio"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="direction-type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="floor">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="control" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="floor_id" type="xs:positiveInteger" use="required"/>
    <xs:attribute name="instances" type="xs:positiveInteger" use="required"/>
  </xs:complexType>
</xs:element>

>
  <xs:attribute name="enable" type="xs:boolean" use="required"/>
  <xs:attribute name="value" type="xs:boolean" use="required"/>
</xs:complexType>
</xs:element>

<xs:element name="control">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="label" type="xs:string"/>
      <xs:element name="value" type="control-value"/>
    </xs:sequence>
    <xs:attribute name="type" type="control-type" use="required"/>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="enable" type="xs:boolean" use="required"/>
  </xs:complexType>
</xs:element>
```



```
<xs:simpleType name="control-value">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:real"/>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:boolean"/>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

<xs:simpleType name="control-type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="real"/>
    <xs:enumeration value="boolean"/>
  </xs:restriction>
</xs:simpleType>

</xs:schema>
```

[4.2.](#) **Advanced Audio Template**

TODO

[4.3.](#) **Basic-Video Template**

[4.3.1.](#) **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 a control that allows input to be paused. The conference server chooses the output video stream for each participant based on local policy. The local policy for example can be active-speaker or the video of the host of the conference or the moderator of the conference. The template also defines a moderator role who has the ability to pause a users video input.

[4.3.2.](#) **Roles**

Participant: The basic video template specifies the role of 'Participant' to signify an entry level user with no privileges by default. The participant role could obtain access to all controls

available for this template if set accordingly by a privileged user.

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 'pause-video' control of the output video stream is set to 'false'.

4.3.3. Parameters

4.3.3.1. 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". The parameter can appear at either a template or role level.

4.3.3.2. max-video-streams

The 'max-video-streams' parameter specifies the maximum number of video 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.

4.3.4. Controls

In the basic-video template, a pause-video control is defined which basically pauses the input and the output stream from and to the conference server. It is expected that for a very basic video conferencing this control is sufficient. If the conference server wants to support more advance features, then it is recommended that advance-video template to be used. In advance-video template the schema for controls is extensible, hence new control types can be defined.

Issue : For the basic video template we are restricting the type of the control to boolean (for pause-video control). The reason for doing is the basic-audio can only have very basic control like pause-video. Should we keep the restriction or let the capability to define new control in basic-video template.

4.3.4.1. 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: The 'pause-video' control has a type of 'Boolean'.

name: The control is named 'pause-video'.

default: Indicates the default value for the 'pause-video' control when not set. The attribute type is 'Boolean' with a default value of '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: The enable attribute specifies if access to the control is available to the client. The value is of type 'boolean' and is set to 'true' as default.

4.3.5. Control Array

In the basic-video template, two logical streams are defined - mainvideoIn, mainvideoOut. The mainvideoIn comprises of the input video stream from the participants and the mainvideoout stream comprises of the output video stream from the server to the participants. For the role of moderator, two controlArrays are defined which defines the control pause-video for each of the logical streams. With the help of this controls, the moderator can pause any participants video being mixed in the server and can pause any output video stream to any of the participants of the conference.

4.3.6. Streams

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

4.3.6.1. VideoIn

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

type: The stream type is 'video'.

name: The descriptive name for this stream is 'VideoIn'.

dir: The direction of the stream is set to 'in'.

stream-id : The unique identifier of the stream.

[4.3.6.1.1.](#) Floor

[4.3.6.1.1.1.](#) Send_Video

The 'Send_Video' floor is associated with the 'VideoIn' stream. It provides a mechanism to control the Video input to a media mix. A participant with a role of 'Moderator' should always have floor preference - this is considered out of scope for this document. The following attributes are defined:

floor_id: Uniquely identifies the floor within the context of this template and has a value of '1'.

instances: Value of '1' implies only one occurrence of this floor at any one time.

enable: A value of 'false' indicates that this floor is not enabled by default.

value: A value of 'false' specifies that the instance represented by this template does not have control of the floor. A value of 'true' indicates control of the floor.

enable: A value of 'true' specifies that floor control is in operation while 'false' specifies that floor control is not in operation.

[4.3.6.2.](#) VideoOut

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

type: The stream type is 'video'.

name: The descriptive name for this stream is 'VideoOut'.

dir: The direction of the stream is set to 'out'.

stream-id : The unique identifier of the stream.

[4.3.7.](#) Blueprint - XML Example

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



```

        <role name="Participant">
            <parameter type="integer" name="max-participants"
value="1" min="1" max="128"/>

            <stream type="video" name="VideoIn" dir="in">
                <floor name="Send_Video" floor_id="1" instances="1"
enable="true" value="false">
                    <control name ="pause-video" type="boolean"
enable="true" >
                        <label>Pause-Video</label>
                        <value>False</value>
                    </control>
                </floor>
            </stream>

            <stream type="video" name="VideoOut" dir="out"/>
        </role>

        <role name="Moderator">

            <parameter type="integer" name="max-participants"
value="1" min="0" max="1"/>

            <controlArray>
                <LogicalStream type="video" name="MainVideoIn" />
                <control type="boolean" name="pause-video"
enable="false">
                    <label>PauseInputVideo</label>
                    <value>False</value>
                </control>
            </controlArray>

            <controlArray>
                <LogicalStream type="video" name="MainVideoOut" />
                <control type="boolean" name="pause-video"
enable="false">
                    <label>PauseOutputVideo</label>
                    <value>False</value>
                </control>
            </controlArray>

            <stream type="video" name="VideoIn" dir="in">
                <floor name="Send_video" floor_id="1" instances="1"
enable="true" value="true">
                    <control name ="pause-video" type="boolean"
enable="true" >
                        <label>Pause-Video</label>

```

```
        <value>False</value>
      </control>
    </floor>
  </stream>

  <stream type="video" name="VideoOut" dir="out"/>
</role>
```

```
</template>
```

4.3.8. XML Schema Definition

This section provides the XML schema definition for the 'Basic-Video' template.

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:role="urn:ietf:params:xml:ns:role-schema">

    <import namespace="urn:ietf:params:xml:ns:role-schema" />

    <xs:element name="template-basic-video">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="role" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="parameter">
      <xs:complexType>
        <xs:attribute name="type" type="parameter-defined" use="required"/>
        <xs:attribute name="name" type="parameter-name" use="required"/>
        <xs:attribute name="value" type="parameter-value" use="required"/>
        <xs:attribute name="min" type="xs:positiveInteger" use="optional"/>
        <xs:attribute name="max" type="xs:positiveInteger" use="optional"/>
      </xs:complexType>
    </xs:element>

    <xs:simpleType name="parameter-value">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:integer"/>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:real"/>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:enumeration"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
```



```
<xs:simpleType name="parameter-defined">
  <xs:restriction base="xs:string">
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="enumeration"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="role">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="controlArray" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="stream" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="rtype" use="required"/>
  </xs:complexType>
</xs:element>

  <xs:simpleType name="rtype">
    <xs:restriction base="role:role-type">
      <xs:enumeration value="Participant"/>
      <xs:enumeration value="Moderator"/>
    </xs:restriction>
  </xs:simpleType>

<xs:element name="controlArray">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="LogicalStream" minOccurs="1" maxOccurs="unbounded" />
      <xs:element ref="control" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="stream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="floor" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="control" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="media-type" use="required"/>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="dir" type="direction-type" use="required"/>
    <xs:attribute name="stream-id" type="xs:positiveInteger" use="optional"/>
  </xs:complexType>
</xs:element>
```

>

Boulton & Chandra

Expires April 26, 2006

[Page 20]

```

    </xs:complexType>
  </xs:element>

  <xs:element name="LogicalStream">
    <xs:complexType>
      <xs:attribute name="type" type="media-type" />
      <xs:attribute name="name" type="logical-Streamtype" />
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="logical-streamType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MainVideoIn"/>
      <xs:enumeration value="MainVideoOut"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="media-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="video"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="direction-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="in"/>
      <xs:enumeration value="out"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="floor">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="control" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="floor_id" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="instances" type="xs:positiveInteger" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="control">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="value" type="xs:boolean"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```
<xs:attribute name="type" type="control-type" use="required"/>
```

```
<xs:attribute name="name" type="xs:string" use="required"/>
<xs:attribute name="enable" type="xs:boolean" use="required"/>
</xs:element>

<xs:simpleType name="control-type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="boolean"/>
  </xs:restriction>
</xs:simpleType>

</xs:schema>
```

[4.4.](#) **Advanced Video Template**

TODO

[4.5.](#) **Basic Text Template**

[4.5.1.](#) **Description**

The Basic-Text template can be used to achieve the simplest form of interaction between multiple text input streams from participating entities. The output stream will consist of all input messages except those injected by the participant in question. Controls are defined to limit who can inject into a conference and a privilege user has the ability to block participants receiving outgoing media.

[4.5.2.](#) **Roles**

Participant: The basic audio template specifies the role of 'Participant' to signify an entry level user with no privileges by default. The participant role could obtain access to all controls available for this template if set accordingly by a privileged user. Access to the controls available to a participant can be controlled using floor element.

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 the same controls as a 'Participant'. The moderator also has the capability to block any input stream from being mixed in the output stream and can also block the output stream to any participant. See the individual control definitions for the impact of attribute settings.

[4.5.3.](#) Parameters

max-participants: The 'max-participants' parameter specifies the maximum number of participants (Entity with signalling connection to the focus) that are permitted to be involved in an instantiated instance of the template (or sub-element, depending on where specified) for the specified role. The minimum value permitted is "1" and the maximum value permitted is "128".

[4.5.4.](#) Controls

[4.5.4.1.](#) block

The 'block' control is used in conjunction with a input (and output) stream from a participant (and from the conference server) to cease transmission of associated media. It has a 'Boolean' value. The 'block' control consists of the following attributes:-

type: The 'block' control has a type of 'Boolean'.

name: The control is named 'block'.

default: Indicates the default value for the 'block' control when not set. The attribute type is 'Boolean' with a default value of 'false'. Setting the 'block-incoming' 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: The enable attribute specifies if access to the control is available to the client. The value is of type 'boolean' and is set to 'true' as default.

[4.5.5.](#) Control Array

In the basic-text template, two logical streams are defined - mainTextIn, mainTextOut. The mainTextIn comprises of the input text stream from the participants and the maintextout stream comprises of the output text stream from the server to the participants. For the role of moderator, two controlArrays are defined which defines the control "block" for each of the logical streams. With the help of this controls, the moderator can block any participants text stream from being mixed in the server and can block any output text stream to any of the participants of the conference.

[4.5.6.](#) Streams

The 'Basic-Text' Template consists of two text media streams:

[4.5.6.1.](#) **TextIn**

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

type: The stream type is 'text'.

name: The descriptive name for this stream is 'TextIn'.

dir: The direction of the stream is set to 'in'.

[4.5.6.1.1.](#) **Floor**

[4.5.6.1.1.1.](#) **send_text**

The 'send_text' floor is associated with the 'TextIn' stream. It provides a mechanism to control the text input to a text media mix. A participant with a role of 'Moderator' should always have floor preference - this is considered out of scope for this document. The following attributes are defined:

floor_id: Uniquely identifies the floor within the context of this template and has a value of '1'.

instances: Indicates the number of occurrences of this floor at any one time.

enable: A value of 'false' indicates that this floor is not enabled. A value of 'true' indicates that this floor is enabled.

value: A value of 'false' specifies that the instance represented by this template does not have control of the floor. A value of 'true' indicates control of the floor.

[4.5.6.2.](#) **TextOut**

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

type: The stream type is 'text'.

name: The descriptive name for this stream is 'TextOut'.

dir: The direction of the stream is set to 'out'.

[4.5.7.](#) **Blueprint - XML Example**


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

  <parameter type="integer" name="max-participants" min="1"
max="128"/>

  <role name="Participant">
    <parameter type="integer" name="max-participants" value="1"
min="1" max="128"/>

    <stream type="text" name="TextIn" dir="in">
      <floor name="send_text" floor_id="1" instances="1"
enable="true" value="false">
        <control type="boolean" name="block" enable="true">
          <label>Block-Incoming-Text</label>
          <value>True</value>
        </control>
      </floor>
    </stream>

    <stream type="text" name="TextOut" dir="out"/>
  </role>

  <role name="moderator">
    <parameter type="integer" name="max-participants"
value="0" min="1" max="128"/>

    <controlArray>
      <LogicalStream name="MainTextIn" type="Text"/>
      <control name="block" type="boolean" enable="false">
        <label>Stop-Input-Text</label>
        <value>False</value>
      </control>
    </controlArray>

    <controlArray>
      <LogicalStream name="MainTextOut" type="Text" />
      <control name="block" type="boolean" enable="false">
        <label>StopOutputText</label>
        <value>False</value>
      </control>
    </controlArray>

    <stream type="text" name="TextIn" dir="in">
      <floor name="Right_to_inject" floor_id="1"
instances="1" enable="true" value="true">
        <control type="boolean" name="block" enable="true">
          <label>Block-Incoming-Text</label>
          <value>False</value>
        </control>
```

```
</floor>  
</stream>  
  
<stream type="text" name="TextOut" dir="out"/>
```

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

4.5.8. XML Schema Definition

This section provides the XML schema definition for the 'Basic-Text' template.

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:role="urn:ietf:params:xml:ns:role-schema">

    <import namespace="urn:ietf:params:xml:ns:role-schema" />

    <xs:element name="template-basic-text">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="role" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="parameter">
      <xs:complexType>
        <xs:attribute name="type" type="parameter-defined" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="value" type="parameter-value" use="required"/>
        <xs:attribute name="min" type="xs:positiveInteger" use="optional"/>
        <xs:attribute name="max" type="xs:positiveInteger" use="optional"/>
      </xs:complexType>
    </xs:element>

    <xs:simpleType name="parameter-value">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:integer"/>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:real"/>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:enumeration"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
```



```
<xs:simpleType name="parameter-defined">
  <xs:restriction base="xs:string">
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="enumeration"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="role">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="controlArray" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="stream" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="rtype" use="required"/>
  </xs:complexType>
</xs:element>

  <xs:simpleType name="rtype">
    <xs:restriction base="role:role-type">
      <xs:enumeration value="Participant"/>
      <xs:enumeration value="Moderator"/>
    </xs:restriction>
  </xs:simpleType>

<xs:element name="controlArray">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="LogicalStream" minOccurs="1" maxOccurs="unbounded" />
      <xs:element ref="control" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="stream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="floor" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="control" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="media-type" use="required"/>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="dir" type="direction-type" use="required"/>
    <xs:attribute name="stream-id" type="xs:positiveInteger" use="optional"/>
  </xs:complexType>
</xs:element>
```

>

Boulton & Chandra

Expires April 26, 2006

[Page 27]

```
    </xs:complexType>
  </xs:element>

  <xs:element name="LogicalStream">
    <xs:complexType>
      <xs:attribute name="type" type="media-type" />
      <xs:attribute name="name" type="logical-Streamtype" />
    </xs:complexType>
  </xs:element>

  <xs:simpleType name="logical-streamType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="MainTextIn"/>
      <xs:enumeration value="MainTextOut"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="media-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="text"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="direction-type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="in"/>
      <xs:enumeration value="out"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="floor">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="control" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="floor_id" type="xs:positiveInteger" use="required"/>
      <xs:attribute name="instances" type="xs:positiveInteger" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="control">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="value" type="control-value"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:attribute name="type" type="control-type" use="required"/>
```

```
<xs:attribute name="name" type="xs:string" use="required"/>
<xs:attribute name="enable" type="xs:boolean" use="required"/>
</xs:element>

<xs:simpleType name="control-type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="boolean"/>
  </xs:restriction>
</xs:simpleType>

</xs:schema>
```

[4.6.](#) **Advanced Text Template**

TODO

[4.7.](#) **Basic Audio-Video Template**

[4.7.1.](#) **Description**

The Basic Audio-Video template enables a conference to provide both audio and video interactions between entities that provide input stream to a Conference mix. The Audio and Video output streams consist of a mix of all participants audio and video input. This particular template and definition creates a union of functionality from the basic audio and basic video templates.

[4.7.2.](#) **Roles**

[4.7.2.1.](#) **Participant**

The basic Audio-Video template specifies the role of 'Participant' to signify an entry level user with no privileges by default. The participant role could obtain access to all controls available for this template if set accordingly by a privileged user. This is achieved during the blueprint creation phase by setting the enable attribute of the control element to TRUE.

[4.7.2.2.](#) **Moderator**

The basic Audio-Video template specifies the role of 'Moderator' to signify a user with advanced privileges by default. The 'Moderator' role for this template has the same controls as a 'Participant' with

an additional controls for muting audio and stopping video for any participant in a conference instance. Another primary difference between a 'Participant' and a 'Moderator' being that the default value for the 'enable' attribute on the 'mute' and 'pause-video' controls on an output media stream is set to 'true'. See the individual control definitions for the impact of attribute settings.

4.7.3. Parameters

4.7.3.1. max-participants

See [Section 4.1.3.1](#) for definition.

4.7.3.2. max-audio-streams

See [Section 4.1.3.2](#) for definition.

4.7.3.3. max-video-streams

See [Section 4.3.3.2](#) for definition.

4.7.4. Controls

4.7.4.1. mute

See [Section 4.1.4.1](#) for definition.

4.7.4.2. pause-video

See [Section 4.3.4.1](#) for definition.

4.7.4.3. gain

See [Section 4.1.4.2](#) for definition.

4.7.5. Control Array

See [Section 4.1.5](#) for information relating to audio Control arrays and [Section 4.3.5](#) for information relating to video related Control arrays in this template.

4.7.6. Streams

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

4.7.6.1. AudioIn

See [Section 4.1.6.1](#) for definition.

4.7.6.2. AudioOut

See [Section 4.1.6.2](#) for definition.

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

4.7.6.3. VideoIn

See [Section 4.3.6.1](#) for definition.

4.7.6.4. VideoOut

See [Section 4.3.6.2](#) for definition.

4.7.7. Blueprint - XML Example

```
<template name="basic-audio-video">
    <parameter type="integer" name="max-participants" value="1"
min="1" max="128"/>
    <parameter type="integer" name="max-audio-streams" value="1"
min="1" max="128"/>
    <parameter type="integer" name="max-video-streams" value="1"
min="1" max="128"/>
    <role name="Participant">
        <parameter type="integer" name="max-participants"
value="1" min="1" max="128"/>
        <stream type="audio" name="AudioIn" dir="in">
            <floor name="Send_audio" floor_id="1"
instances="1" enable="true" value="false">
                <control type="boolean" name="mute"
enable="true">
                    <label> Mute-Audio </label>
                    <value>True</value>
                </control>
            </floor>
        </stream>
        <stream type="video" name="VideoIn" dir="in">
            <floor name="Send_Video" floor_id="1"
instances="1" enable="true" value="false">
                <control name="pause-video" type="boolean"
enable="true" >
                    <label>Pause-Video</label>
                    <value>False</value>
                </control>
            </floor>
        </stream>
    </role>
</template>
```

```
</floor>
</stream>

<stream type="audio" name="AudioOut" dir="out">
  <control type="real" name="gain" enable="true" >
    <label>Volume</label>
    <value>0</value>
```

```

        </control>
    </stream>

    <stream type="video" name="VideoOut" dir="out"/>
</role>

<role name="moderator">
    <parameter type="integer" name="max-participants"
value="0" min="0" max="128"/>

    <controlArray >
        <LogicalStream type="audio" name="MainAudioIn" />
        <control type="boolean" name="mute"
enable="false">

            <label>MuteInputAudio</label>
            <value>False</value>
        </control>
    </controlArray>

    <controlArray>
        <LogicalStream type="video" name="MainVideoIn" />
        <control type="boolean" name="pause-video"
enable="false">

            <label>PauseInputVideo</label>
            <value>False</value>
        </control>
    </controlArray>

    <controlArray>
        <LogicalStream type="audio" name="MainAudioOut" /
>

        <control name="mute" type="boolean"
enable="false">

            <label>MuteOutputAudio</label>
            <value>False</value>
        </control>
    </controlArray>

    <controlArray>
        <LogicalStream type="video" name="MainVideoOut" />
        <control type="boolean" name="pause-video"
enable="false">

            <label>PauseOutputVideo</label>
            <value>False</value>
        </control>
    </controlArray>

    <stream type="audio" name="AudioIn" dir="in">

```

```

                                <floor name="Send_audio" floor_id="1"
instances="1" enable="true" value="true">
                                <control type ="boolean" name="mute"
enable="true">
                                <label> Mute-Audio </label>
                                <value>True</value>
                                </control>
                                </floor>

```

```

        </stream>

        <stream type="video" name="VideoIn" dir="in">
          <floor name="Send_video" floor_id="1" instances="1"
enable="true" value="true">
            <control name ="pause-video" type="boolean"
enable="true" >
              <label>Pause-Video</label>
              <value>False</value>
            </control>
          </floor>
        </stream>

        <stream type="audio" name="AudioOut" dir="out">
          <control type="real" name="gain" enable="true" >
            <label>Volume</label>
            <value>1.0</value>
          </control>
        </stream>

        <stream type="video" name="VideoOut" dir="out"/>

      </role>

</template>

```

4.7.8. XML Schema Definition

```

<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:role="urn:ietf:params:xml:ns:role-schema">

    <import namespace="urn:ietf:params:xml:ns:role-schema" />

    <xs:element name="basic-audio-video">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="role" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:element name="parameter">
      <xs:complexType>
        <xs:attribute name="type" type="parameter-defined" use="required"/>
        <xs:attribute name="name" type="parameter-name" use="required"/>
      </xs:complexType>
    </xs:element>

```

```
<xs:attribute name="value" type="parameter-value" use="required"/>
```

```
        <xs:attribute name="min" type="xs:positiveInteger" use="optional"/>
        <xs:attribute name="max" type="xs:positiveInteger" use="optional"/>
    </xs:complexType>
</xs:element>

<xs:simpleType name="parameter-value">
    <xs:union>
        <xs:simpleType>
            <xs:restriction base="xs:integer"/>
        </xs:simpleType>
        <xs:simpleType>
            <xs:restriction base="xs:real"/>
        </xs:simpleType>
        <xs:simpleType>
            <xs:restriction base="xs:enumeration"/>
        </xs:simpleType>
    </xs:union>
</xs:simpleType>

<xs:simpleType name="parameter-name">
    <xs:restriction base="xs:string">
        <xs:enumeration value="max-participants"/>
        <xs:enumeration value="max-audio-streams"/>
        <xs:enumeration value="max-video-streams"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="parameter-defined">
    <xs:restriction base="xs:string">
        <xs:enumeration value="integer"/>
        <xs:enumeration value="real"/>
        <xs:enumeration value="enumeration"/>
    </xs:restriction>
</xs:simpleType>

<xs:element name="role">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="controlArray" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="stream" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="rtype" use="required"/>
    </xs:complexType>
</xs:element>

    <xs:simpleType name="rtype">
        <xs:restriction base="role:role-type">
```



```
        <xs:enumeration value="Participant"/>
        <xs:enumeration value="Moderator"/>
    </xs:restriction>
</xs:simpleType>
```

```
<xs:element name="controlArray">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="LogicalStream" minOccurs="1" maxOccurs="unbounded" />
      <xs:element ref="control" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="stream">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="floor" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="control" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="media-type" use="required"/>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="dir" type="direction-type" use="required"/>
    <xs:attribute name="stream-id" type="xs:positiveInteger" use="optional"/>
>
```

```
    </xs:complexType>
  </xs:element>
```

```
<xs:element name="LogicalStream">
  <xs:complexType>
    <xs:attribute name="type" type="media-type" />
    <xs:attribute name="name" type="logical-Streamtype" />
  <xs:complexType>
</xs:element>
```

```
<xs:simpleType name="logical-streamType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AudioIn"/>
    <xs:enumeration value="AudioOut"/>
    <xs:enumeration value="VideoIn"/>
    <xs:enumeration value="VideoOut"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="media-type">
  <xs:restriction base="xs:string">
```

```
<xs:enumeration value="audio"/>
```

```
        <xs:enumeration value="video"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="direction-type">
      <xs:restriction base="xs:string">
        <xs:enumeration value="in"/>
        <xs:enumeration value="out"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:element name="floor">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="control" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="floor_id" type="xs:positiveInteger" use="required"/>
        <xs:attribute name="instances" type="xs:positiveInteger" use="required"/>
      >
        <xs:attribute name="enable" type="xs:boolean" use="required"/>
        <xs:attribute name="value" type="xs:boolean" use="required"/>
      </xs:complexType>
    </xs:element>

    <xs:element name="control">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="label" type="xs:string"/>
          <xs:element name="value" type="control-value"/>
        </xs:sequence>
        <xs:attribute name="type" type="control-type" use="required"/>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="enable" type="xs:boolean" use="required"/>
      </xs:complexType>
    </xs:element>

    <xs:simpleType name="control-value">
      <xs:union>
        <xs:simpleType>
          <xs:restriction base="xs:real"/>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:boolean"/>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>

    <xs:simpleType name="control-type">
      <xs:restriction base="xs:string">
```

```
<xs:enumeration value="real"/>
```

```
<xs:enumeration value="boolean"/>
</xs:restriction>
</xs:simpleType>

</xs:schema>
```

[4.8.](#) Audio-text Template

TODO

[4.9.](#) Basic Multimedia Template

TODO

[4.10.](#) Advanced Multimedia Template

[5.](#) Security Considerations

[6.](#) IANA Considerations

[7.](#) Acknowledgements

Many thanks to the media policy design team members for helpful comments including Cullen Jennings, Alan Johnston, Roni Even, Brian Rosen and Mark Trayer. Special thanks goes to Alan Johnston for his guidance.

[8.](#) References

[8.1.](#) Normative References

- [1] Barnes, M., "A Framework and Data Model for Centralized Conferencing", [draft-ietf-xcon-framework-01](#) (work in progress), July 2005.
- [2] Jennings, C. and B. Rosen, "Media Conference Server Control for XCON", [draft-jennings-xcon-media-control-03](#) (work in progress), July 2005.
- [3] Morgan, D. and O. Novo, "Role Definitions for Centralized Conferencing", [draft-morgan-xcon-roles-00](#) (work in progress), October 2005.

- [4] Novo, O., "A Common Conference Information Data Model for Centralized Conferencing (XCON)", [draft-novo-xcon-common-data-model-00](#) (work in progress), September 2005.

8.2. Informative References

Authors' Addresses

Chris Boulton
Ubiquity Software Corporation
Building 3
Wern Fawr Lane
St Mellons, Cardiff, Wales UK CF3 5EA

Email: cboulton@ubiquity.net

Umesh Chandra
Nokia Research Center
6000 Connection Drive
Irving, Texas 75063

Email: Umesh.Chandra@nokia.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights 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; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat 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 implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

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

