Internet Engineering Task Force (IETF)
Request for Comments: 6505
Category: Standards Track
ISSN: 2070-1721

S. McGlashan
Hewlett-Packard
T. Melanchuk
Rainwillow
C. Boulton
NS-Technologies

March 2012

A Mixer Control Package for the Media Control Channel Framework

Abstract

This document defines a Media Control Channel Framework Package for managing mixers for media conferences and connections. The package defines request elements for managing conference mixers, managing mixers between conferences and/or connections, as well as associated responses and notifications. The package also defines elements for auditing package capabilities and mixers.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6505.

Table of Contents

1. Introduction

The Media Control Channel Framework [RFC6230] provides a generic approach for establishment and reporting capabilities of remotely initiated commands. The Control Framework -- an equivalent term for the Media Control Channel Framework -- utilizes many functions provided by the Session Initiation Protocol (SIP) [RFC3261] for the rendezvous and establishment of a reliable channel for control interactions. The Control Framework also introduces the concept of a Control Package. A Control Package is an explicit usage of the Control Framework for a particular interaction set. This specification defines a package for media conference mixers and media connection mixers.

This package defines mixer management elements for creating, modifying, and deleting conference mixers, elements for joining, modifying, and unjoining media streams between connections and conferences (including mixers between connections), as well as associated responses and notifications. The package also defines elements for auditing package capabilities and mixers.

This package has been designed to satisfy media-mixing requirements documented in the Media Server Control Protocol Requirements document [RFC5167]; more specifically REQ-MCP-22, REQ-MCP-23, REQ-MCP-24, REQ-MCP-25, REQ-MCP-26, and REQ-MCP-27.

The package provides the major conferencing functionality of SIP media server languages such as MSCML [RFC5022] and MSML [RFC5707]. A key differentiator is that this package provides such functionality using the Media Control Channel Framework.

Out of scope for this mixer package are more advanced functions including personalized video mixes for conference participants, support for floor control protocols, as well as support for video overlays and text insertion. Such functionality can be addressed by extensions to this package (through addition of foreign elements or attributes from another namespace) or use of other Control Packages that could build upon this package.

The functionality of this package is defined by messages, containing XML [XML] elements and transported using the Media Control Channel Framework. The XML elements can be divided into two types: mixer management elements and audit elements (for auditing package capabilities and mixers managed by the package).

The document is organized as follows. Section 3 describes how this Control Package fulfills the requirements for a Media Control Channel Framework Control Package. Section 4 describes the syntax and

semantics of defined elements, including mixer management (Section 4.2) and audit elements (Section 4.3). Section 5 describes an XML schema for these elements and provides extensibility by allowing attributes and elements from other namespaces. Section 6 provides examples of package usage. Section 7 describes important security considerations for use of this Control Package. Section 8 provides information on IANA registration of this Control Package, including its name, XML namespace, and MIME media type.

2. Conventions and Terminology

In this document, BCP 14 [RFC2119] defines the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL". In addition, BCP 15 indicates requirement levels for compliant implementations.

The following additional terms are defined for use in this document:

Application Server: A SIP [RFC3261] application server (AS) is a control client that hosts and executes services such as interactive media and conferencing in an operator's network. An AS controls the media server (MS), influencing and impacting the SIP sessions terminating on an MS, which the AS can have established, for example, using SIP third-party call control.

Media Server: A media server (MS) processes media streams on behalf of an AS by offering functionality such as interactive media, conferencing, and transcoding to the end user. Interactive media functionality is realized by way of dialogs, which are identified by a URI and initiated by the application server.

MS Conference: An MS Conference provides the media-related mixing resources and services for conferences. In this document, an MS Conference is often referred to simply as a conference.

MS Connection: An MS connection represents the termination on a media server of one or more RTP [RFC3550] sessions that are associated to a single SIP dialog. A media server receives media from the output(s) of a connection, and it transmits media on the input(s) of a connection.

Media Stream: A media stream on a media server represents a media flow between either a connection and a conference, between two connections, or between two conferences. Streams can be audio or video and can be bidirectional or unidirectional.

3. Control Package Definition

This section fulfills the mandatory requirement for information that MUST be specified during the definition of a Control Framework Package, as detailed in Section 8 of [RFC6230].

3.1. Control Package Name

The Control Framework requires a Control Package definition to specify and register a unique name. The name and version of this Control Package is "msc-mixer/1.0" (Media Server Control - Mixer - version 1.0). Its IANA registration is specified in Section 8.1.

Since this is the initial ("1.0") version of the Control Package, there are no backwards compatibility issues to address.

3.2. Framework Message Usage

The Control Framework requires a Control Package to explicitly detail the control messages that can be used as well as provide an indication of directionality between entities. This will include which role type is allowed to initiate a request type.

This package specifies CONTROL and response messages in terms of XML elements defined in Section 4, where the message bodies have the MIME media type defined in Section 8.4. These elements describe requests, responses, and notifications, and all are contained within a root <mscmixer> element (Section 4.1).

In this package, the MS operates as a Control Server in receiving requests from, and sending responses to, the AS (operating as a Control Client). Mixer management requests and responses are defined in Section 4.2. Audit requests and responses are defined in Section 4.3. Mixer management and audit responses are carried in a framework 200 response or REPORT message bodies. This package's response codes are defined in Section 4.6.

Note that package responses are different from framework response codes. Framework error response codes (see Section 7 of [RFC6230]) are used when the request or event notification is invalid, for example, a request is invalid XML (400) or not understood (500).

The MS also operates as a Control Client in sending event notification to the AS (Control Server). Event notifications (Section 4.2.4) are carried in CONTROL message bodies. The AS MUST respond with a Control Framework 200 response.

3.3. Common XML Support

The Control Framework requires a Control Package definition to specify if the attributes for media dialog or conference references are required.

This package requires that the XML schema in Appendix A.1 of [RFC6230] MUST be supported for media dialogs and conferences.

The package uses 'connectionid' and 'conferenceid' attributes for various element definitions (Section 4). The XML schema (Section 5) imports the definitions of these attributes from the framework schema.

3.4. CONTROL Message Body

The Control Framework requires a Control Package to define the control body that can be contained within a CONTROL command request and to indicate the location of detailed syntax definitions and semantics for the appropriate body types.

When operating as a Control Server, the MS receives CONTROL messages with the MIME media type defined in Section 8.4 and a body containing a <mscmixer> element (Section 4.1) with either a mixer management or audit request child element.

The following mixer management request elements are carried in CONTROL message bodies to MS: <createconference> (Section 4.2.1.1), <modifyconference> (Section 4.2.1.2), <destroyconference> (Section 4.2.1.3), <join> (Section 4.2.2.2), <modifyjoin> (Section 4.2.2.3), and <unjoin> (Section 4.2.2.4) elements.

The <audit> request element (Section 4.3.1) is also carried in CONTROL message bodies.

When operating as a Control Client, the MS sends CONTROL messages with the MIME media type defined in Section 8.4 and a body containing a <mscmixer> element (Section 4.1) with a notification <event> child element (Section 4.2.4).

3.5. REPORT Message Body

The Control Framework requires a Control Package definition to define the REPORT body that can be contained within a REPORT command request, or to indicate that no report package body is required. This section indicates the location of detailed syntax definitions and semantics for the appropriate body types.

When operating as a Control Server, the MS sends REPORT bodies with the MIME media type defined in Section 8.4 and a <mscmixer> element with a response child element. The response element for mixer management requests is a <response> element (Section 4.2.3). The response element for an audit request is a <auditresponse> element (Section 4.3.2).

3.6. Audit

The Control Framework encourages Control Packages to specify whether auditing is available, how it is triggered, as well as the query/response formats.

This Control Package supports auditing of package capabilities and mixers on the MS. An audit request is carried in a CONTROL message and an audit response in a REPORT message (or a 200 response to the CONTROL if it can execute the audit in time).

The syntax and semantics of audit request and response elements are defined in Section 4.3.

3.7. Examples

The Control Framework recommends Control Packages to provide a range of message flows that represent common flows using the package and this framework document.

This Control Package provides examples of such message flows in Section 6.

4. Element Definitions

This section defines the XML elements for this package. The elements are defined in the XML namespace specified in Section 8.2.

The root element is <mscmixer> (Section 4.1). All other XML elements (requests, responses, and notification elements) are contained within it. Child elements describe mixer management (Section 4.2) and audit (Section 4.3) functionality. Response status codes are defined in Section 4.6 and type definitions in Section 4.7.

Implementation of this Control Package MUST address the security considerations described in Section 7.

Implementation of this Control Package MUST adhere to the syntax and semantics of XML elements defined in this section and the schema (Section 5). The XML schema supports extensibility by allowing attributes and elements from other namespaces. Implementations MAY

support attributes and elements from other (foreign) namespaces. If an MS implementation receives a <mscmixer> element containing attributes or elements from another namespace, which it does not support, the MS sends a 428 response (Section 4.6).

Extensible attributes and elements are not described in this section. In all other cases where there is a difference in constraints between the XML schema and the textual description of elements in this section, the textual definition takes priority.

Some elements in this Control Package contain attributes whose value is descriptive text primarily for diagnostic use. The implementation can indicated the language used in the descriptive text by means of a 'desclang' attribute [RFC2277]. The 'desclang' attribute can appear on the root element as well as selected subordinate elements (see Section 4.1). The 'desclang' attribute value on the root element applies to all 'desclang' attributes in subordinate elements unless the subordinate element has an explicit 'desclang' attribute that overrides it.

Usage examples are provided in Section 6.

4.1. <mscmixer>

The <mscmixer> element has the following attributes (in addition to standard XML namespace attributes such as 'xmlns'):

version: a string specifying the mscmixer package version. The value is fixed as "1.0" for this version of the package. The attribute is mandatory.

desclang: specifies the language used in descriptive text attributes of subordinate elements (unless the subordinate element provides a 'desclang' attribute that overrides the value for its descriptive text attributes). The descriptive text attributes on subordinate elements include: the 'reason' attribute on <response> (Section 4.2.3), <unjoin-notify> (Section 4.2.4.2), <conferenceexit> (Section 4.2.4.3), and <auditresponse> (Section 4.3.2). A valid value is a language identifier (Section 4.7.7). The attribute is optional. The default value is "i-default" (BCP 47 [RFC5646]).

The <mscmixer> element has the following defined child elements, only one of which can occur:

1. mixer management elements defined in Section 4.2:

   <createconference>: create and configure a new conference mixer. See Section 4.2.1.1

   <modifyconference>: modify the configuration of an existing conference mixer. See Section 4.2.1.2

   <destroyconference>: destroy an existing conference mixer. See Section 4.2.1.3

   <join>: create and configure media streams between connections and/or conferences (for example, add a participant to a conference). See Section 4.2.2.2

   <modifyjoin>: modify the configuration of joined media streams. See Section 4.2.2.3

   <unjoin>: delete a media stream (for example, remove a participant from a conference). See Section 4.2.2.4

   <response>: response to a mixer request. See Section 4.2.3

   <event>: mixer or subscription notification. See Section 4.2.4

2. audit elements defined in Section 4.3:

   <audit>: audit package capabilities and managed mixers. See Section 4.3.1

   <auditresponse>: response to an audit request. See Section 4.3.2

For example, a request to the MS to create a conference mixer is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <createconference/>
</mscmixer>
```

And a response from the MS that the conference was successfully created is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer"
  desclang="en">
 <response status="200" conferenceid="conference1"
   reason="conference created"/>
</mscmixer>
```

4.2. Mixer Elements

This section defines the mixer management XML elements for this Control Package. These elements are divided into requests, responses, and notifications.

Request elements are sent to the MS to request a specific mixer operation to be executed. The following request elements are defined:

<createconference>: create and configure a new a conference mixer. See Section 4.2.1.1

<modifyconference>: modify the configuration of an existing conference mixer. See Section 4.2.1.2

<destroyconference>: destroy an existing conference mixer. See Section 4.2.1.3

<join>: create and configure media streams between connections and/or conferences (for example, add a participant to a conference). See Section 4.2.2.2

<modifyjoin>: modify the configuration of joined media streams. See Section 4.2.2.3

<unjoin>: delete a media stream (for example, remove a participant from a conference). See Section 4.2.2.4

Responses from the MS describe the status of the requested operation. Responses are specified in a <response> element (Section 4.2.3) that includes a mandatory attribute describing the status in terms of a numeric code. Response status codes are defined in Section 4.6. The MS MUST respond to a request message with a response message. If the MS is not able to process the request and carry out the mixer operation (in whole or in part), then the request has failed: the MS MUST ensure that no part of the requested mixer operation is carried out, and the MS MUST indicate the class of failure using an appropriate 4xx response code. Unless an error response code is specified for a class of error within this section, implementations follow Section 4.6 in determining the appropriate status code for the response.

Notifications are sent from the MS to provide updates on the status of a mixer operation or subscription. Notifications are specified in an <event> element (Section 4.2.4).

4.2.1. Conference Elements

4.2.1.1. <createconference>

The <createconference> element is sent to the MS to request creation of a new conference (multiparty) mixer.

The <createconference> element has the following attributes:

conferenceid: string indicating a unique name for the new conference. If this attribute is not specified, the MS MUST create a unique name for the conference. The value is used in subsequent references to the conference (e.g., as conferenceid in a <response>). The attribute is optional. There is no default value.

reserved-talkers: indicates the requested number of guaranteed speaker slots to be reserved for the conference. A valid value is a non-negative integer (see Section 4.7.2). The attribute is optional. The default value is 0.

reserved-listeners: indicates the requested number of guaranteed listener slots to be reserved for the conference. A valid value is a non-negative integer (see Section 4.7.2). The attribute is optional. The default value is 0.

The <createconference> element has the following sequence of child elements:

<codecs>: an element to configure the codecs supported by the conference (see Section 4.4). If codecs are specified, then they impose limitations on media capability when the MS attempts to join the conference to other entities (see Sections 4.2.2.2 and 4.2.2.3). The element is optional.

<audio-mixing>: an element to configure the audio mixing characteristics of a conference (see Section 4.2.1.4.1). The element is optional.

<video-layouts>: an element to configure the video layouts of a conference (see Section 4.2.1.4.2). The element is optional.

<video-switch>: an element to configure the video switch policy for the layout of a conference (see Section 4.2.1.4.3). The element is optional.

<subscribe>: an element to request subscription to conference events. (see Section 4.2.1.4.4). The element is optional.

If the 'conferenceid' attribute specifies a value that is already used by an existing conference, the MS reports an error (405) and MUST NOT create a new conference and MUST NOT affect the existing conference.

If the MS is unable to configure the conference according to specified 'reserved-talkers' or 'reserved-listeners' attributes, the MS reports an error (420) and MUST NOT create the conference.

If the MS is unable to configure the conference according to a specified <audio-mixing> element, the MS reports an error (421) and MUST NOT create the conference.

If the MS is unable to configure the conference according to a specified <video-layouts> element, the MS reports an error (423) and MUST NOT create the conference.

If the MS is unable to configure the conference according to a specified <video-switch> element, the MS reports an error (424) and MUST NOT create the conference.

If the MS is unable to configure the conference according to a specified <codecs> element, the MS reports an error (425) and MUST NOT create the conference.

When a MS has finished processing a <createconference> request, it MUST reply with an appropriate <response> element (Section 4.2.3).

For example, a request to create an audio video conference mixer with specified codecs, video layout, video switch, and subscription is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <createconference conferenceid="conference1"
       reserved-talkers="1" reserved-listeners="10">
   <codecs>
    <codec name="video">
     <subtype>H264</subtype>
    </codec>
    <codec name="audio">
     <subtype>PCMA</subtype>
    </codec>
   </codecs>
   <audio-mixing type="nbest"/>
   <video-layouts>
    <video-layout min-participants="1"><single-view/></video-layout>
    <video-layout min-participants="2"><dual-view/></video-layout>
    <video-layout min-participants="3"><quad-view/></video-layout>
   </video-layouts>
   <video-switch interval="5"><vas/></video-switch>
   <subscribe>
    <active-talkers-sub interval="4"/>
   </subscribe>
 </createconference>
</mscmixer>
```

A response from the MS if the conference was successfully created is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <response status="200" conferenceid="conference1"/>
</mscmixer>
```

Alternatively, a response if the MS could not create the conference due to a lack of support for the H264 codec is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <response status="425" conferenceid="conference1"
           reason="H264 codec not supported"/>
</mscmixer>
```

4.2.1.2. <modifyconference>

The <modifyconference> element is sent to the MS to request modification of an existing conference.

The <modifyconference> element has the following attribute:

conferenceid: string indicating the name of the conference to modify. This attribute is mandatory.

The <modifyconference> element has the following sequence of child elements (one or more):

<codecs>: an element to configure the codecs supported by the conference (see Section 4.4). If codecs are specified, then they impose limitations in media capability when the MS attempts to join the conference to other entities (see Sections 4.2.2.2 and 4.2.2.3). Existing conference participants are unaffected by any policy change. The element is optional.

<audio-mixing>: an element to configure the audio mixing characteristics of a conference (see Section 4.2.1.4.1). The element is optional.

<video-layouts>: an element to configure the video layouts of a conference (see Section 4.2.1.4.2). The element is optional.

<video-switch>: an element to configure the video switch policy for the layout of a conference (see Section 4.2.1.4.3). The element is optional.

<subscribe>: an element to request subscription to conference events. (see Section 4.2.1.4.4). The element is optional.

If the 'conferenceid' attribute specifies the name of a conference that does not exist, the MS reports an error (406).

If the MS is unable to configure the conference according to a specified <audio-mixing> element, the MS reports an error (421) and MUST NOT modify the conference in any way.

If the MS is unable to configure the conference according to a specified <video-layouts> element, the MS reports an error (423) and MUST NOT modify the conference in any way.

If the MS is unable to configure the conference according to a specified <video-switch> element, the MS reports an error (424) and MUST NOT modify the conference in any way.

If the MS is unable to configure the conference according to a specified <codecs> element, the MS reports an error (425) and MUST NOT modify the conference.

When a MS has finished processing a <modifyconference> request, it MUST reply with an appropriate <response> element (Section 4.2.3).

4.2.1.3. <destroyconference>

The <destroyconference> element is sent to the MS to request destruction of an existing conference.

The <destroyconference> element has the following attribute:

conferenceid: string indicating the name of the conference to destroy. This attribute is mandatory.

The <destroyconference> element does not specify any child elements.

If the 'conferenceid' attribute specifies the name of a conference that does not exist, the MS reports an error (406).

When a MS has finished processing a <destroyconference> request, it MUST reply with an appropriate <response> element (Section 4.2.3).

Successfully destroying the conference (status code 200) will result in all connection or conference participants being removed from the conference mixer, <unjoin-notify> notification events (Section 4.2.4.2) being sent for each conference participant, and a <conferenceexit> notification event (Section 4.2.4.3) indicating that conference has exited. A <response> with any other status code indicates that the conference mixer still exists and participants are still joined to the mixer.

4.2.1.4. Conference Configuration

The elements in this section are used to establish and modify the configuration of conferences.

4.2.1.4.1. <audio-mixing>

The <audio-mixing> element defines the configuration of the conference audio mix.

The <audio-mixing> element has the following attributes:

type: is a string indicating the audio stream mixing policy. Defined values are: "nbest" (where the N best (loudest) participant signals are mixed) and "controller" (where the contributing participant(s) is/are selected by the controlling AS via an external floor control protocol). The attribute is optional. The default value is "nbest".

n: indicates the number of eligible participants included in the conference audio mix. An eligible participant is a participant who contributes audio to the conference. Inclusion is based on having the greatest audio energy. A valid value is a non-negative integer (see Section 4.7.2). A value of 0 indicates that all participants contributing audio to the conference are included in the audio mix. The default value is 0. The element is optional.

If the 'type' attribute does not have the value "nbest", the MS ignores the 'n' attribute.

The <audio-mixing> element has no child elements.

For example, a fragment where the audio-mixing policy is set to "nbest" with 3 participants to be included is as follows:

```
<audio-mixing type="nbest" n="3"/>
```

If the conference had 200 participants of whom 30 contributed audio, then there would be 30 eligible participants for the audio mix. Of these, the 3 loudest participants would have their audio included in the conference.

4.2.1.4.2. <video-layouts>

The <video-layouts> element describes the video presentation layout configuration for participants providing a video input stream to the conference. This element allows multiple video layouts to be specified so that the MS automatically changes layout depending on the number of video-enabled participants.

The <video-layouts> element has no attributes.

The <video-layouts> element has the following sequence of child elements (one or more):

<video-layout>: element describing a video layout (Section 4.2.1.4.2.1).

If the MS does not support video conferencing at all, or does not support multiple video layouts, or does not support a specific video layout, the MS reports an 423 error in the response to the request element containing the <video-layouts> element.

An MS MAY support more than one <video-layout> element, although only one layout can be active at a time. A <video-layout> is active if the number of participants in the conference is equal to or greater than the value of its 'min-participants' attribute, but less than the

value of the 'min-participants' attribute for any other <video-layout> element. An MS reports an error (400) if more than one <video-layout> has the same value for the 'min-participants' attribute. When the number of regions within the active layout is greater than the number of participants in the conference, the display of unassigned regions is implementation-specific.

The assignment of participant video streams to regions within the layout is according to the video switch policy specified by the <video-switch> element (Section 4.2.1.4.3).

For example, a fragment describing a single layout is as follows:

```
<video-layouts>
  <video-layout><single-view/></video-layout>
</video-layouts>
```

A fragment describing a sequence of layouts is as follows:

```
<video-layouts>
  <video-layout min-participants="1"><single-view/></video-layout>
  <video-layout min-participants="2"><dual-view/></video-layout>
  <video-layout min-participants="3"><quad-view/></video-layout>
  <video-layout min-participants="5"><multiple-3x3/></video-layout>
</video-layouts>
```

When the conference has one participant providing a video input stream to the conference, then the single-view format is used. When the conference has two such participants, the dual-view layout is used. When the conference has three or four participants, the quad-view layout is used. When the conference has five or more participants, the multiple-3x3 layout is used.

4.2.1.4.2.1. <video-layout>

The <video-layout> element describes a video layout containing one or more regions in which participant video input streams are displayed.

The <video-layout> element has the following attribute:

min-participants: the minimum number of conference participants needed to allow this layout to be active. A valid value is a positive integer (see Section 4.7.3). The attribute is optional. The default value is 1.

The <video-layout> element has one child element specifying the video layout. An MS MAY support the predefined video layouts defined in the conference information data model for centralized conferencing

(XCON) [RFC6501]: <single-view>, <dual-view>, <dual-view-crop>, <dual-view-2x1>, <dual-view-2x1-crop>, <quad-view>, <multiple-3x3>, <multiple-4x4>, and <multiple-5x1>.

The MS MAY support other video layouts. Non-XCON layouts MUST be specified using an element from a namespace other than the one used in this specification, for example:

```
<video-layout>
 <mylayout xmlns='http://example.com/foo'>my-single-view</mylayout>
</video-layout>
```

If the MS does not support the specified video layout configuration, then the MS reports a 423 error (Section 4.6) in the response to the request element containing the <video-layout> element.

Each video layout has associated with it one or more regions. The XCON layouts are associated with the following named regions:

<single-view/>: layout with one stream in a single region as shown in Figure 1.

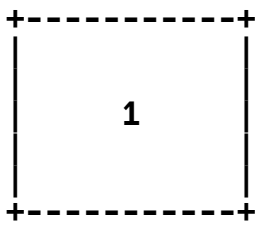


Figure 1: single-view video layout

<dual-view/>: layout presenting two streams side-by-side in two regions as shown in Figure 2. The MS MUST NOT alter the aspect ratio of each stream to fit the region, and hence the MS might need to blank out part of each region.

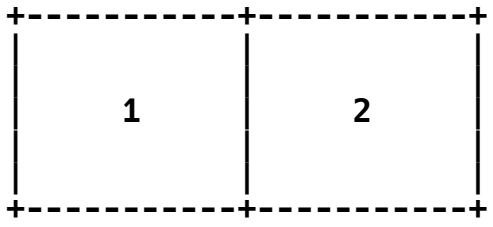


Figure 2: dual-view video layout

<dual-view-crop/>: layout presenting two streams side-by-side in two regions as shown in Figure 3. The MS MUST alter the aspect ratio of each stream to fit its region so that no blanking is required.

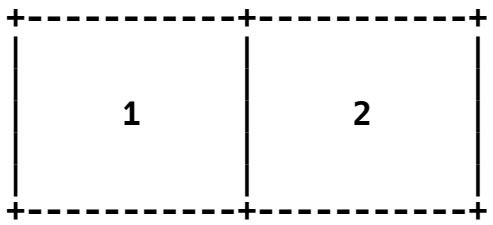


Figure 3: dual-view-crop video layout

<dual-view-2x1/>: layout presenting two streams, one above the other, in two regions as shown in Figure 4. The MS MUST NOT alter the aspect ratio of each stream to fit its region, and hence the MS might need to blank out part of each region.

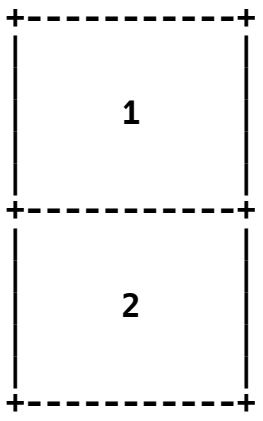


Figure 4: dual-view-2x1 video layout

<dual-view-2x1-crop/>: layout presenting two streams one above the other in two regions as shown in Figure 5. The MS MUST alter the aspect ratio of each stream to fit its region so that no blanking is required.

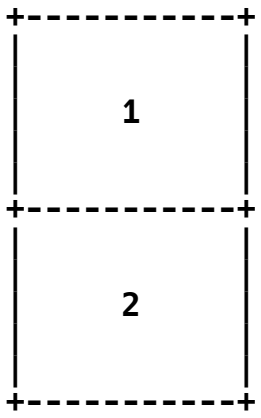


Figure 5: dual-view-2x1-crop video layout

<quad-view/>: layout presenting four equal-sized regions in a 2x2 layout as shown in Figure 6. Typically, the aspect ratio of the streams is preserved, so blanking is required.

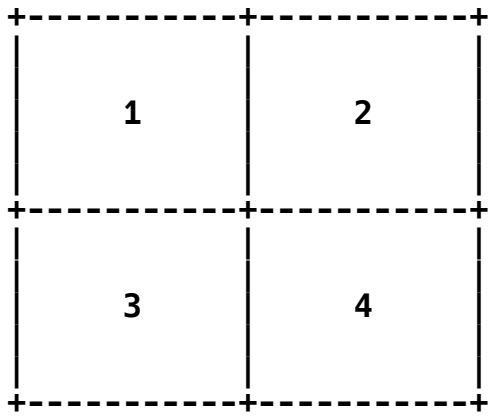


Figure 6: quad-view video layout

<multiple-3x3/>: layout presenting nine equal-sized regions in a 3x3 layout as shown in Figure 7. Typically, the aspect ratio of the streams is preserved, so blanking is required.

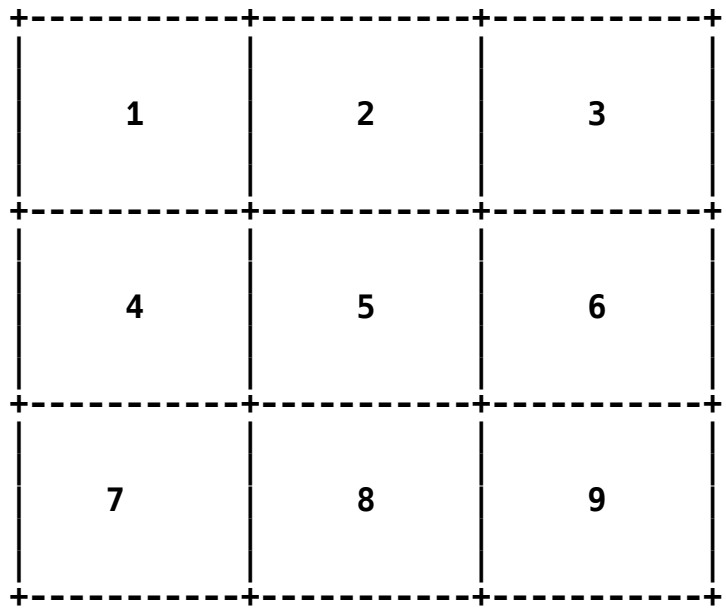


Figure 7: multiple-3x3 video layout

<multiple-4x4/>: layout presenting 16 equal-sized regions in a 4x4 layout as shown in Figure 8. Typically, the aspect ratio of the streams is preserved, so blanking is required.

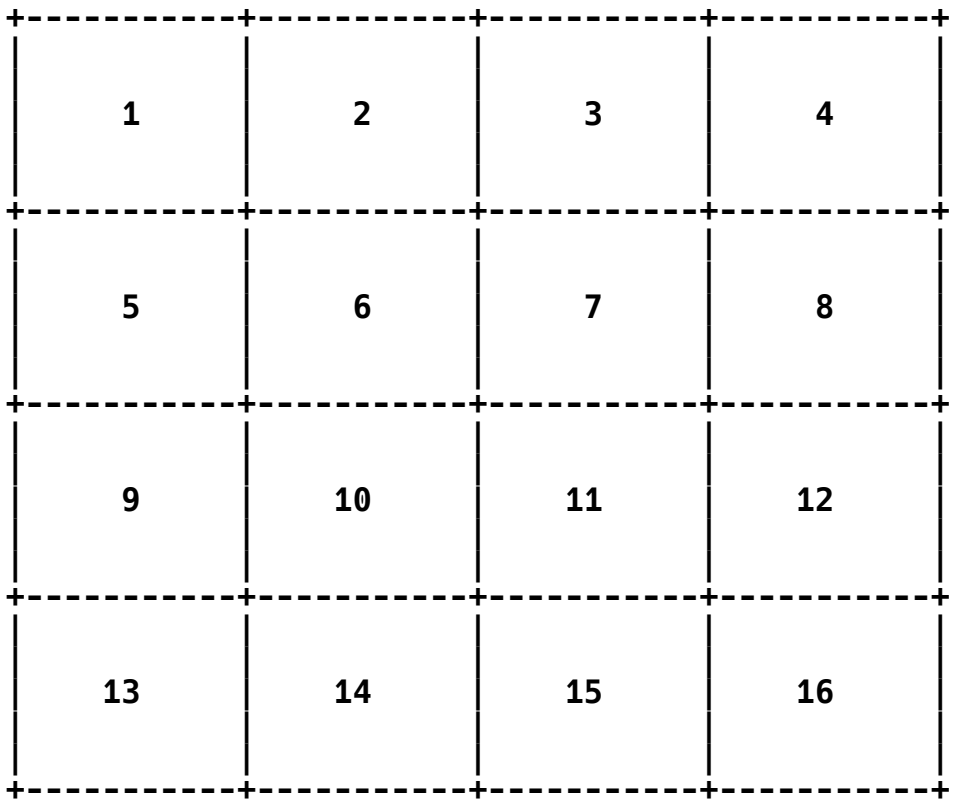


Figure 8: multiple-4x4 video layout

<multiple-5x1/>: layout presents a 5x1 layout as shown in Figure 9 where one region will occupy 4/9 of the mixed video stream, while the others will each occupy 1/9 of the stream. Typically, the aspect ratio of the streams is preserved, so blanking is required.

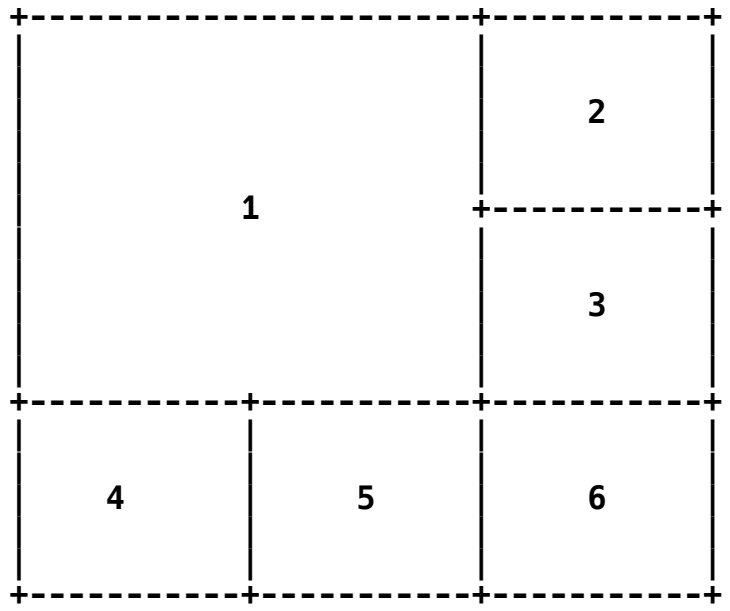


Figure 9: multiple-5x1 video layout

4.2.1.4.3. <video-switch>

The <video-switch> element describes the configuration of the conference policy for how participants' input video streams are assigned to regions within the active video layout.

The <video-switch> element has the following child elements defined (one child occurrence only) to indicate the video-switching policy of the conference:

<vas/>: (Voice-Activated Switching) enables automatic display of the loudest speaker participant that is contributing both audio and video to the conference mix. Participants who do not provide an audio stream are not considered for automatic display. If a participant provides more than one audio stream, then the policy for inclusion of such a participant in the VAS is implementation-specific; an MS could select one stream, sum audio streams, or ignore the participant for VAS consideration. If there is only one region in the layout, then the loudest speaker is displayed there. If more than one region is available, then the loudest speaker is displayed in the largest region (if any), and then in

the first region from the top-left corner of the layout. The MS assigns the remaining regions based on the priority mechanism described in Section 4.2.1.4.3.1.

<controller/>: enables manual control over video switching. The controller AS determines how the regions are assigned based on an external floor control policy. The MS receives <join>, <modifyjoin>, and <unjoin> commands with a <stream> element (Section 4.2.2.5) indicating the region where the stream is displayed. If no explicit region is specified, the MS assigns the region based on the priority mechanism described in Section 4.2.1.4.3.1.

An MS MAY support other video-switching policies. Other policies MUST be specified using an element from a namespace other than the one used in this specification. For example:

```
<video-switch>
 <mypolicy xmlns='http://example.com/foo'/>
</video-switch>
```

The <video-switch> element has the following attributes:

interval: specifies the period between video switches as a number of seconds. In the case of <vas/> policy, a speaker needs to be the loudest speaker for the interval before the switch takes place. A valid value is a non-negative integer (see Section 4.7.2). A value of 0 indicates that switching is applied immediately. The attribute is optional. The default value is 3 (seconds).

activespeakermix: indicates whether or not the active (loudest) speaker participant receives a video stream without themselves displayed in the case of the <vas/> switching policy. If enabled, the MS needs to generate two video streams for each conference mix: one for the active speaker participant without themselves displayed (details of this video layout are implementation-specific) and one for other participants (as described in the <vas/> switching policy above). A valid value is a boolean (see Section 4.7.1). A value of "true" indicates that a separate video mix is generated for the active speaker without themselves being displayed. A value of "false" indicates that all participants receive the same video mix. The attribute is optional. The default value is "false". If the 'type' attribute is not set to <vas/>, the MS ignores this attribute.

If the MS does not support the specified video-switching policy or other configuration parameters (including separate active speaker video mixes), then the MS reports a 424 error (Section 4.6) in the response to the request element containing the <video-switch> element.

If the MS receives a <join> or <modifyjoin> request containing a <stream> element (Section 4.2.2.5) that specifies a region and the conference video-switching policy is set to <vas/>, then the MS ignores the region (i.e., conference-switching policy takes precedence).

If the MS receives a <join> or <modifyjoin> request containing a <stream> element (Section 4.2.2.5) specifying a region that is not defined for the currently active video layout, the MS MUST NOT report an error. Even though the participant is not currently visible, the MS displays the participant if the layout changes to one that defines the specified region.

For example, a fragment specifying a <vas/> video-switching policy with an interval of 2s

```
<video-switch interval="2"><vas/></video-switch>
```

For example, a fragment specifying a <controller/> video-switching policy where video switching takes place immediately is as follows:

```
<video-switch interval="0"><controller/></video-switch>
```

4.2.1.4.3.1. Priority Assignment

In cases where the video-switching policy does not explicitly determine the region to which a participant is assigned, the following priority assignment mechanism applies:

1. Each participant has a (positive integer) priority value: the lower the value, the higher the priority. The priority value is determined by the <priority> child element (Section 4.2.2.5.4) of <stream>. If not explicitly specified, the default priority value is 100.

2. The MS uses priority values to assign participants to regions in the video layout which remain unfilled after application of the video-switching policy. The MS MUST dedicate larger and/or more prominent portions of the layout to participants with higher priority values first (e.g., first, all participants with priority 1, then those with 2, 3, etc.).

3. The policy for displaying participants with the same priority is implementation-specific.

The MS applies this priority policy each time the video layout is changed or updated. It is RECOMMENDED that the MS does not move a participant from one region to another unless required by the video-switching policy when an active video layout is updated.

This model allows the MS to apply default video layouts after applying the video-switching policy. For example, region 2 is statically assigned to Bob, so the priority mechanism only applies to regions 1, 3, 4, etc.

4.2.1.4.4. <subscribe>

The <subscribe> element is a container for specifying conference notification events to which a controlling entity subscribes. Notifications of conference events are delivered using the <event> element (see Section 4.2.4).

The <subscribe> element has no attributes, but has the following child element:

<active-talkers-sub>: subscription to active talker events (Section 4.2.1.4.4.1). The element is optional.

The MS MUST support a <active-talkers-sub> subscription. It MAY support other event subscriptions (specified using attributes and child elements from a foreign namespace). If the MS does not support a subscription specified in a foreign namespace, it sends a <response> with a 428 status code (see Section 4.6).

4.2.1.4.4.1. <active-talkers-sub>

The <active-talkers-sub> element has the following attribute:

interval: the minimum amount of time (in seconds) that elapses before further active talker events can be generated. A valid value is a non-negative integer (see Section 4.7.2). A value of 0 suppresses further notifications. The attribute is optional. The default value is 3 (seconds).

The <active-talkers-sub> element has no child elements.

Active talker notifications are delivered in the <active-talkers-notify> element (Section 4.2.4.1).

4.2.2. Joining Elements

This section contains definitions of the joining model (Section 4.2.2.1) as well as the <join> (Section 4.2.2.2), <modifyjoin> (Section 4.2.2.3), <unjoin> (Section 4.2.2.4) and <stream> (Section 4.2.2.5) elements.

4.2.2.1. Joining Model

The <join> operation creates a media stream between a connection and a conference, between connections, or between conferences. This section describes the model of conferences and connections and specifies the behavior for join requests to targets that already have an associated media stream.

Conferences support multiple inputs and have resources to mix them together. A media server conference in essence is a mixer that combines media streams. A simple audio mixer simply sums its input audio signals to create a single common output. Conferences, however, use a more complex algorithm so that participants do not hear themselves as part of the mix. That algorithm, sometimes called an "n-minus mix", subtracts each participants input signal from the summed input signals, creating a unique output for each contributing participant. Each <join> operation to a conference uses one of the conference's available inputs and/or outputs, to the maximum number of supported participants.

A connection is the termination of one or more RTP sessions on a media server. It has a single input and output for each media session established by its SIP dialog. The output of a connection can feed several different inputs such as both a conference mixer and a recording of that participant's audio.

Joining two connections that are not joined to anything else simply creates a media stream from the outputs(s) of one connection to the corresponding inputs(s) of the other connection. It is not necessary to combine media from multiple sources in this case. There are, however, several common scenarios where combining media from several sources to create a single input to a connection is needed.

In the first case, a connection can be receiving media from one source (for example, a conference), and it is necessary to play an announcement to the connection so that both the conference audio and announcement can be heard by the conference participant. This is sometimes referred to as a "whisper announcement". An alternative to a whisper announcement is to have the announcement preempt the conference media.

Another common case is the call-center coaching scenario where a supervisor can listen to the conversation between an agent and a customer, and provide hints to the agent that are not heard by the customer.

Both of these cases can be solved by having the controlling AS create one or more conferences for audio mixing, and then join and unjoin the media streams as required. A better solution is to have the media server automatically mix media streams that are requested to be joined to a common input when only the simple summing of audio signals as described above is required. This is the case for both the use cases presented above.

Automatically mixing streams has several benefits. Conceptually, it is straightforward and simple, requiring no indirect requests on the part of the controlling AS. This increases transport efficiency and reduces the coordination complexity and the latency of the overall operation. Therefore, it is RECOMMENDED that a media server be able to automatically mix at least two audio streams where only the simple summing of signals is required.

When a media server receives a <join> request, it MUST automatically mix all of the media streams included in the request with any streams already joined to one of the entities identified in the request, or it MUST fail the request and MUST NOT join any of the streams (and MUST NOT change existing streams of the entities). A controlling AS uses the <createconference> request for generic conferences where the complex mixing algorithm is required.

Specifications that extend this package to handle additional media types such as text MUST define the semantics of the join operation when multiple streams are requested to be joined to a single input, such as that for a connection with a single RTP session per media type.

4.2.2.2. <join>

The <join> element is sent to the MS to request creation of one or more media streams either between a connection and a conference, between connections, or between conferences. The two entities to join are specified by the attributes of <join>.

Streams can be of any media type and can be bidirectional or unidirectional. A bidirectional stream is implicitly composed of two unidirectional streams that can be manipulated independently. The streams to be established are specified by child <stream> elements (see Section 4.2.2.5).

The <join> element has the following attributes:

id1: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

id2: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

Note: Appendix A.1 of [RFC6230] defines the semantics for a conference identifier but not its syntax. Media server implementations need to distinguish between conferences and connections based upon the values of the 'id1' and 'id2' attributes.

If id1 or id2 specify a conference identifier and the conference does not exist on the MS, the MS reports an error (406). If id1 or id2 specify a connection identifier and the connection does not exist on the MS, the MS reports an error (412).

The <join> element has the following child element (zero or more):

<stream>: an element that both identifies the media streams to join and defines the way that they are to be joined (see Section 4.2.2.5). The element is optional.

If no <stream> elements are specified, then the default is to join all streams between the entities according to the media configuration of the connection or conference.

One or more <stream> elements can be specified so that individual media streams can be controlled independently. For example, if a connection supports both audio and video streams, a <stream> element could be used to indicate that only the audio stream is used in receive mode. In cases where there are multiple media streams of the same type for a connection or conference, the configuration MUST be explicitly specified using <stream> elements.

Multiple <stream> elements can be specified for precise control over the media flow in different directions within the same media stream. One <stream> element can be specified for the receiving media flow and another element for the sending media flow, where each independently controls features such as volume (see child element of <stream> in Section 4.2.2.5). If there is only one <stream> element for a given media specifying a 'sendonly' or 'recvonly' direction, then the media flow in the opposite direction is inactive (established but there's no actual flow of media) unless this leads to a stream conflict.

If the MS is unable to execute the join as specified in <stream> because a <stream> element is in conflict with (a) another <stream> element, (b) specified connection or conference media capabilities (including supported or available codec information), or (c) an Session Description Protocol (SDP) label value as part of the connection-id (see Appendix A.1 of [RFC6230]), then the MS reports an error (407) and MUST NOT join the entities and MUST NOT change existing streams of the entities.

If the MS is unable to execute the join as specified in <stream> elements because the MS does not support the media stream configuration, the MS reports an error (422) and MUST NOT join the entities and MUST NOT change existing streams of the entities.

If the MS is unable to join an entity to a conference because it is full, then the MS reports an error (410).

If the specified entities are already joined, then the MS reports an error (408).

If the MS does not support joining two specified connections together, the MS reports an error (426).

If the MS does not support joining two specified conferences together, the MS reports an error (427).

If the MS is unable to join the specified entities for any other reason, the MS reports an error (411).

When the MS has finished processing a <join> request, it MUST reply with an <response> element (Section 4.2.3).

For example, a request to join two connections together is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="1536067209:913cd14c" id2="1536067209:913cd14c"/>
</mscmixer>
```

The response if the MS doesn't support joining media streams between connections is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <response status="426" reason="mixing connections not supported"/>
</mscmixer>
```

4.2.2.3. <modifyjoin>

The <modifyjoin> element is sent to the MS to request changes in the configuration of media stream(s) that were previously established between a connection and a conference, between two connections, or between two conferences.

The <modifyjoin> element has the following attributes:

id1: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

id2: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

The <modifyjoin> element has the following child elements (one or more):

<stream>: an element that both identifies the media streams to modify and defines the way that each stream is to be configured from this point forward (see Section 4.2.2.5).

The MS MUST support <modifyjoin> for any stream that was established using <join>.

The MS MUST configure the streams that are included within <modifyjoin> to that stated by the child elements.

If the MS is unable to modify the join as specified in <stream> elements because a <stream> element is in conflict with (a) another <stream> element, (b) specified connection or conference media capabilities (including supported or available codec information), or (c) a SDP label value as part of the connection-id (see Appendix A.1 of [RFC6230]), then the MS reports an error (407) and MUST NOT modify the join between the entities and MUST NOT change existing streams of the entities.

If the MS is unable to modify the join as specified in <stream> elements because the MS does not support the media stream configuration, the MS reports an error (422) and MUST NOT modify the join between the entities and MUST NOT change existing streams of the entities.

If the specified entities are not already joined, then the MS reports an error (409).

If the MS is unable to modify the join between the specified entities for any other reason, the MS reports an error (411).

When an MS has finished processing a <modifyjoin> request, it MUST reply with an appropriate <response> element (Section 4.2.3).

In cases where stream characteristics are controlled independently for each direction, then a <modifyjoin> request needs to specify a child element for each direction in order to retain the original stream directionality. For the example, if a <join> request establishes independent control for each direction of an audio stream (see Section 4.2.2.5):

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="1536067209:913cd14c" id2="conference1">
  <stream media="audio" direction="sendonly">
   <volume controltype="setgain" value="-3"/>
  </stream>
  <stream media="audio" direction="recvonly">
   <volume controltype="setgain" value="+3"/>
  </stream>
 </join>
</mscmixer>
```

then the following <modifyjoin> request

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <modifyjoin id1="1536067209:913cd14c" id2="conference1">
  <stream media="audio" direction="sendonly">
    <volume controltype="setgain" value="0"/>
  </stream>
  </modifyjoin>
</mscmixer>
```

would cause, in addition to the modification of the sendonly volume, the overall stream directionality to change from sendrecv to sendonly since there is no <stream> element in this <modifyjoin> request for the recvonly direction. The following would change the sendonly volume and retain the recvonly stream together with its original characteristics such as volume:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <modifyjoin id1="1536067209:913cd14c" id2="conference1">
  <stream media="audio" direction="sendonly">
    <volume controltype="setgain" value="0"/>
  </stream>
  <stream media="audio" direction="recvonly"/>
  </modifyjoin>
</mscmixer>
```

4.2.2.4. <unjoin>

The <unjoin> element is sent to the MS to request removal of previously established media stream(s) from between a connection and a conference, between two connections, or between two conferences.

The <unjoin> element has the following attributes:

id1: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

id2: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Section 15.1 of [RFC6230]. The attribute is mandatory.

The <unjoin> element has the following child element (zero or more occurrences):

<stream>: an element that identifies the media stream(s) to remove (see Section 4.2.2.5). The element is optional. When not present, all currently established streams between "id1" and "id2" are removed.

The MS MUST support <unjoin> for any stream that was established using <join> and that has not already been removed by a previous <unjoin> on the same stream.

If the MS is unable to terminate the join as specified in <stream> elements because a <stream> element is in conflict with (a) another <stream> element, (b) specified connection or conference media capabilities, or (c) a SDP label value as part of the connection-id (see Appendix A.1 of [RFC6230]), then the MS reports an error (407) and MUST NOT terminate the join between the entities and MUST NOT change existing streams of the entities.

If the MS is unable to terminate the join as specified in <stream> elements because the MS does not support the media stream configuration, the MS reports an error (422) and MUST NOT terminate the join between the entities and MUST NOT change existing streams of the entities.

If the specified entities are not already joined, then the MS reports an error (409).

If the MS is unable to terminate the join between the specified entities for any other reason, the MS reports an error (411).

When an MS has successfully processed a <unjoin> request, it MUST reply with a successful <response> element (Section 4.2.3).

4.2.2.5. <stream>

<join>, <modifyjoin>, and <unjoin> require the identification and manipulation of media streams. Media streams represent the flow of media between a participant connection and a conference, between two connections, or between two conferences. The <stream> element is used (as a child to <join>, <modifyjoin>, and <unjoin>) to identify the media stream(s) for the request and to specify the configuration of the media stream.

The <stream> element has the following attributes:

media: a string indicating the type of media associated with the stream. A valid value is a MIME type name as defined in Section 4.2 of [RFC4288]. The following values MUST be used for common types of media: "audio" for audio media, and "video" for video media. See [IANA] for registered MIME type names. The attribute is mandatory.

label: a string indicating the SDP label associated with a media stream [RFC4574]. The attribute is optional.

direction: a string indicating the allowed media flow of the stream relative to the value of the 'id1' attribute of the parent element. Defined values are: "sendrecv" (media can be sent and received), "sendonly" (media can only be sent), "recvonly" (media can only be received), and "inactive" (stream established but no media flow). The default value is "sendrecv". The attribute is optional.

The <stream> element has the following sequence of child elements:

<volume>: an element (Section 4.2.2.5.1) to configure the volume or gain of the media stream. The element is optional.

<clamp>: an element (Section 4.2.2.5.2) to configure filtering and removal of tones from the media stream. The element is optional.

<region>: an element (Section 4.2.2.5.3) to configure a region within a video layout where the media stream is displayed. The element is optional.

<priority>: an element (Section 4.2.2.5.4) to configure priority associated with the stream in the media mix. The element is optional.

In each child element, the media stream affected is indicated by the value of the 'direction' attribute of the parent element.

If the 'media' attribute does not have the value of "audio", then the MS ignores <volume> and <clamp> elements.

If the 'media' attribute does not have the value of "video", then the MS ignores a <region> element.

For example, a request to join a connection to conference in both directions with volume control is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="1536067209:913cd14c" id2="conference1">
  <stream media="audio" direction="sendrecv">
   <volume controltype="setgain" value="-3"/>
  </stream>
 </join>
</mscmixer>
```

where audio flow from the connection (id1) to the conference (id2) has the volume lowered by 3 dB, and likewise the volume of the audio flow from the conference to the connection is lowered by 3 dB.

In this example, the volume is independently controlled for each direction.

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="1536067209:913cd14c" id2="conference1">
  <stream media="audio" direction="sendonly">
   <volume controltype="setgain" value="-3"/>
  </stream>
  <stream media="audio" direction="recvonly">
   <volume controltype="setgain" value="+3"/>
  </stream>
 </join>
</mscmixer>
```

where audio flow from the connection (id1) to the conference (id2) has the volume lowered by 3 dB, but the volume of the audio flow from the conference to the connection is raised by 3 dB.

4.2.2.5.1. <volume>

The <volume> element is used to configure the volume of an audio media stream. It can be set to a specific gain amount, to automatically adjust the gain to a desired target level, or to mute the volume.

The <volume> element has no child elements but has the following attributes:

controltype: a string indicating the type of volume control to use for the stream. Defined values are: "automatic" (the volume will be adjusted automatically to the level specified by the 'value' attribute), "setgain" (use the value of the 'value' attribute as a specific gain measured in dB to apply), and "setstate" (set the state of the stream to "mute" or "unmute" as specified by the value of the 'value' attribute). The attribute is mandatory.

value: a string specifying the amount or state for the volume control defined by the value of the 'controltype' attribute. The attribute is optional. There is no default value.

If the audio media stream is in a muted state, then the MS also changes automatically the state to unmuted with an "automatic" or "setgain" volume control. For example, assume an audio stream has been muted with <volume controltype="setstate" value="mute"/>. If the gain on the same stream is changed with <volume controltype="setgain" value="+3"/>, then the volume is increased and stream state is also changed to unmuted.

4.2.2.5.2. <clamp>

The <clamp> element is used to configure whether tones are filtered and removed from a media stream.

The <clamp> element has no child elements but has the following attribute:

tones: A space-separated list of the tones to remove. The attribute is optional. The default value is "1 2 3 4 5 6 7 8 9 0 * # A B C D" (i.e., all DTMF (Dual-Tone Multi-Frequency) tones are removed).

4.2.2.5.3. <region>

As described in Section 4.2.1.4.2.1, each <video-layout> is composed of one or more named regions (or areas) in which video media can be presented. For example, the XCON layout <dual-view> has two regions named "1" and "2", respectively.

The <region> element is used to explicitly specify the name of the area within a video layout where a video media stream is displayed.

The <region> element has no attributes, and its content model specifies the name of the region.

4.2.2.5.4. <priority>

The <priority> element is used to explicitly specify the priority of a participant. The MS uses this priority to determine where the media stream is displayed within a video layout (Section 4.2.1.4.3.1).

The <priority> element has no attributes, and its content model specifies a positive integer (see Section 4.7.3). The lower the value, the higher the priority.

4.2.3. <response>

Responses to requests are indicated by a <response> element.

The <response> element has following attributes:

status: numeric code indicating the response status. Valid values are defined in Section 4.6. The attribute is mandatory.

reason: string specifying a reason for the response status. The attribute is optional.

desclang: specifies the language used in the value of the 'reason' attribute. A valid value is a language identifier (Section 4.7.7). The attribute is optional. If not specified, the value of the 'desclang' attribute on <mscmixer> (Section 4.1) applies.

conferenceid: string identifying the conference (see Appendix A.1 of [RFC6230]). The attribute is optional.

connectionid: string identifying the SIP dialog connection (see Appendix A.1 of [RFC6230]). The attribute is optional.

For example, a response when a conference was created successfully is as follows:

```
<response code="200"/>
```

If conference creation failed due to the requested conference ID already existing, the response is:

```
<response code="405" reason="Conference already exists"/>
```

4.2.4. <event>

When a mixer generates a notification event, the MS sends the event using an <event> element.

The <event> element has no attributes, but has the following sequence of child elements (zero or more instances of each child):

<active-talkers-notify>: specifies an active talkers notification (Section 4.2.4.1).

<unjoin-notify>: notifies that a connection or conference has been completely unjoined (Section 4.2.4.2).

<conferenceexit>: notifies that a conference has exited (Section 4.2.4.3).

4.2.4.1. <active-talkers-notify>

The <active-talkers-notify> element describes zero or more speakers that have been active in a conference during the specified interval (see Section 4.2.1.4.4.1).

The <active-talkers-notify> element has the following attribute:

conferenceid: string indicating the name of the conference from which the event originated. This attribute is mandatory.

The <active-talkers-notify> element has the following sequence of child elements (zero or more occurrences):

<active-talker>: element describing an active talker (Section 4.2.4.1.1).

4.2.4.1.1. <active-talker>

The <active-talker> element describes an active talker, associated with either a connection or conference participant in a conference.

The <active-talker> element has the following attributes:

connectionid: string indicating the connectionid of the active talker. This attribute is optional. There is no default value.

conferenceid: string indicating the conferenceid of the active talker. This attribute is optional. There is no default value.

Note that the element does not describe an active talker if both the 'connectionid' and 'conferenceid' attributes are specified, or if neither attribute is specified.

The <active-talker> element has no child elements.

4.2.4.2. <unjoin-notify>

The <unjoin-notify> element describes a notification event where a connection and/or conference have been completely unjoined.

The <unjoin-notify> element has the following attributes:

status: a status code indicating why the unjoin occurred. A valid value is a non-negative integer (see Section 4.7.2). The MS MUST support the following values:

0 indicates the join has been terminated by a <unjoin> request.

1 indicates the join terminated due to an execution error.

2 indicates that the join terminated because a connection or conference has terminated.

All other valid but undefined values are reserved for future use, where new status codes are assigned using the Standards Action process defined in [RFC5226]. The AS MUST treat any status code it does not recognize as being equivalent to 1 (join execution error). The attribute is mandatory.

reason: a textual description providing a reason for the status code, e.g., details about an error. A valid value is a string (see Section 4.7.4). The attribute is optional. There is no default value.

desclang: specifies the language used in the value of the 'reason' attribute. A valid value is a language identifier (Section 4.7.7). The attribute is optional. If not specified, the value of the 'desclang' attribute on <mscmixer> (Section 4.1) applies.

id1: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

id2: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

The <unjoin-notify> element has no child elements.

4.2.4.3. <conferenceexit>

The <conferenceexit> element indicates that a conference has exited because it has been terminated or because a error occurred (for example, a hardware error in the conference mixing unit). This event MUST be sent by the MS whenever a successfully created conference exits.

The <conferenceexit> element has the following attributes:

conferenceid: string indicating the name of the conference. This attribute is mandatory.

status: a status code indicating why the conference exited. A valid value is a non-negative integer (see Section 4.7.2). The MS MUST support the following values:

0 indicates the conference has been terminated by a <destroyconference> request.

1 indicates the conference terminated due to an execution error.

2 indicates the conference terminated due to exceeding the maximum duration for a conference.

All other valid but undefined values are reserved for future use, where new status codes are assigned using the Standards Action process defined in [RFC5226]. The AS MUST treat any status code it does not recognize as being equivalent to 1 (conference execution error). The attribute is mandatory.

reason: a textual description providing a reason for the status code, e.g., details about an error. A valid value is a string (see Section 4.7.4). The attribute is optional. There is no default value.

desclang: specifies the language used in the value of the 'reason' attribute. A valid value is a language identifier (Section 4.7.7). The attribute is optional. If not specified, the value of the 'desclang' attribute on <mscmixer> (Section 4.1) applies.

The <conferenceexit> element has no child elements.

When a MS sends a <conferenceexit> event, the identifier for the conference ('conferenceid' attribute) is no longer valid on the MS and can be reused for another conference.

For example, the following notification event would be sent from the MS when the conference with identifier "conference99" exits due to a successful <destroyconference/>:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <event>
  <conferenceexit conferenceid="conference99"
     status="0"/>
 </event>
</mscmixer>
```

4.3. Audit Elements

The audit elements defined in this section allow the MS to be audited for package capabilities as well as mixers managed by the package. Auditing is particularly important for two use cases. First, it enables discovery of package capabilities supported on an MS before an AS creates a conference mixer or joins connections and conferences. The AS can then use this information to create request elements using supported capabilities and, in the case of codecs, to negotiate an appropriate SDP for a user agent's connection. Second, auditing enables discovery of the existence and status of mixers

currently managed by the package on the MS. This could be used when one AS takes over management of mixers if the AS that created the mixers fails or is no longer available (see the security considerations in Section 7).

4.3.1. <audit>

The <audit> request element is sent to the MS to request information about the capabilities of, and mixers currently managed with, this Control Package. Capabilities include supported conference codecs and video layouts. Mixer information includes the status of managed mixers as well as codecs.

The <audit> element has the following attributes:

capabilities: indicates whether package capabilities are to be audited. A valid value is a boolean (see Section 4.7.1). A value of "true" indicates that capability information is to be reported. A value of "false" indicates that capability information is not to be reported. The attribute is optional. The default value is "true".

mixers: indicates whether mixers currently managed by the package are to be audited. A valid value is a boolean (see Section 4.7.1). A value of "true" indicates that mixer information is to be reported. A value of "false" indicates that mixer information is not to be reported. The attribute is optional. The default value is "true".

conferenceid: string identifying a specific conference mixer to audit. It is an error (406) if the 'conferenceid' attribute is specified and the conference identifier is not valid. The attribute is optional. There is no default value.

If the 'mixers' attribute has the value "true" and 'conferenceid' attribute is specified, then only audit information about the specified conference mixer is reported. If the 'mixers' attribute has the value "false", then no mixer audit information is reported even if a 'conferenceid' attribute is specified.

The <audit> element has no child elements.

When the MS receives an <audit> request, it MUST reply with a <auditresponse> element (Section 4.3.2) that includes a mandatory attribute describing the status in terms of a numeric code. Response status codes are defined in Section 4.6. If the request is successful, the <auditresponse> contains (depending on attribute values) a <capabilities> element (Section 4.3.2.1) reporting package

capabilities and a <mixers> element (Section 4.3.2.2) reporting managed mixer information. If the MS is not able to process the request and carry out the audit operation, the audit request has failed and the MS MUST indicate the class of failure using an appropriate 4xx response code. Unless an error response code is specified for a class of error within this section, implementations follow Section 4.6 in determining the appropriate status code for the response.

For example, a request to audit capabilities and mixers managed by the package is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
  <audit/>
</mscmixer>
```

In this example, only capabilities are to be audited:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
  <audit mixers="false"/>
</mscmixer>
```

With this example, only a specific conference mixer is to be audited:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
  <audit capabilities="false" conferenceid="conf4"/>
</mscmixer>
```

4.3.2. <auditresponse>

The <auditresponse> element describes a response to a <audit> request.

The <auditresponse> element has the following attributes:

status: numeric code indicating the audit response status. The attribute is mandatory. Valid values are defined in Section 4.6.

reason: string specifying a reason for the status. The attribute is optional.

desclang: specifies the language used in the value of the 'reason' attribute. A valid value is a language identifier (Section 4.7.7). The attribute is optional. If not specified, the value of the 'desclang' attribute on <mscmixer> (Section 4.1) applies.

The <auditresponse> element has the following sequence of child elements:

<capabilities>: element describing capabilities of the package (see Section 4.3.2.1). The element is optional.

<mixers>: element describing information about managed mixers (see Section 4.3.2.2). The element is optional.

For example, a successful response to an <audit> request for capabilities and mixer information is as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <auditresponse status="200">
  <capabilities>
   <codecs>
    <codec name="video">
     <subtype>H263</subtype>
    </codec>
    <codec name="video">
     <subtype>H264</subtype>
    </codec>
    <codec name="audio">
     <subtype>PCMU</subtype>
    </codec>
    <codec name="audio">
     <subtype>PCMA</subtype>
    </codec>
   </codecs>
  </capabilities>
  <mixers>
   <conferenceaudit conferenceid="conf1">
    <codecs>
     <codec name="audio">
      <subtype>PCMA</subtype>
     </codec>
    </codecs>
    <participants>
     <participant id="1536067209:913cd14c"/>
    </participants>
   </conferenceaudit>
   <joinaudit id1="1536067209:913cd14c" id2="conf1"/>
   <joinaudit id1="1636067209:113cd14c" id2="1836067209:313cd14c"/>
   <joinaudit id1="1736067209:213cd14c" id2="1936067209:413cd14c"/>
  </mixers>
 </auditresponse>
</mscmixer>
```

4.3.2.1. <capabilities>

The <capabilities> element provides audit information about package capabilities.

The <capabilities> element has no attributes.

The <capabilities> element has the following sequence of child elements:

<codecs>: element (Section 4.4) describing codecs available to the package. The element is mandatory.

For example, a fragment describing capabilities is as follows:

```
<capabilities>
 <codecs>
  <codec name="video">
   <subtype>H263</subtype>
  </codec>
  <codec name="video">
   <subtype>H264</subtype>
  </codec>
  <codec name="audio">
   <subtype>PCMU</subtype>
  </codec>
  <codec name="audio">
   <subtype>PCMA</subtype>
  </codec>
 </codecs>
</capabilities>
```

4.3.2.2. <mixers>

The <mixers> element provides audit information about mixers.

The <mixers> element has no attributes.

The <mixers> element has the following sequence of child elements (zero or more occurrences, any order):

<conferenceaudit>: audit information for a conference mixer (Section 4.3.2.2.1). The element is optional.

<joinaudit>: audit information for a join mixer (Section 4.3.2.2.2). The element is optional.

4.3.2.2.1. <conferenceaudit>

The <conferenceaudit> element has the following attribute:

conferenceid: string identifying the conference (see Appendix A.1 of [RFC6230]). The attribute is mandatory.

The <conferenceaudit> element has the following sequence of child elements:

<codecs> element describing codecs used in the conference. See Section 4.4. The element is optional.

<participants> element listing connections or conferences joined to the conference. See Section 4.3.2.2.1.1. The element is optional.

<video-layout> element describing the active video layout for the conference. See Section 4.2.1.4.2.1. The element is optional.

For example, a fragment describing a conference that has been created but has no participants is as follows:

```
<conferenceaudit conferenceid="conference1"/>
```

A fragment when the same conference has three participants (two connections and another conference) joined to it is as follows:

```
<conferenceaudit conferenceid="conference1">
 <codecs>
  <codec name="audio">
   <subtype>PCMU</subtype>
  </codec>
 </codecs>
 <participants>
   <participant id="connection1"/>
   <participant id="connection2"/>
   <participant id="conference2"/>
 </participants>
</conferenceaudit>
```

4.3.2.2.1.1. <participants>

The <participants> element is a container for <participant> elements (Section 4.3.2.2.1.1.1).

The <participants> element has no attributes, but the following child elements are defined (zero or more):

<participant>: specifies a participant (Section 4.3.2.2.1.1.1).

4.3.2.2.1.1.1. <participant>

The <participant> element describes a participant.

The <participant> element has the following attribute:

id: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

The <participant> element has no children.

4.3.2.2.2. <joinaudit>

The <joinaudit> element has the following attributes:

id1: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

id2: an identifier for either a connection or a conference. The identifier MUST conform to the syntax defined in Appendix A.1 of [RFC6230]. The attribute is mandatory.

The <joinaudit> element has no children.

For example, a fragment describing an audit of two join mixers, one between connections and the second between conferences, is as follows:

```
<mixers>
 <joinaudit id1="1536067209:913cd14" id2="1636067209:413cd14"/>
 <joinaudit id1="conference1" id2="conference2"/>
</mixers>
```

4.4. <codecs>

The <codecs> element is a container for one or more codec definitions. Codec definitions are used by an AS to specify the codecs allowed for a conference (e.g., when used as a child of <createconference> or <modifyconference). Codec definitions are used by an MS to provide audit information about the codecs supported by an MS and used in specific conferences.

The <codecs> element has no attributes.

The <codecs> element has the following sequence of child elements (zero or more occurrences):

<codec>: defines a codec and optionally its policy (Section 4.4.1). The element is optional.

For example, a fragment describing two codecs is as follows:

```
<codecs>
  <codec name="audio">
   <subtype>PCMA</subtype>
  </codec>
  <codec name="video">
    <subtype>H263</subtype>
  </codec>
</codecs>
```

4.4.1. <codec>

The <codec> element describes a codec. The element is modeled on the <codec> element in the XCON conference information data model ([RFC6501]) and allows additional information (e.g., rate, speed, etc.) to be specified.

The <codec> element has the following attribute:

name: indicates the type name of the codec's media format as defined in [IANA]. A valid value is a "type-name" as defined in Section 4.2 of [RFC4288]. The attribute is mandatory.

The <codec> element has the following sequence of child elements:

<subtype>: element whose content model describes the subtype of the codec's media format as defined in [IANA]. A valid value is a "subtype-name" as defined in Section 4.2 of [RFC4288]. The element is mandatory.

<params>: element (Section 4.5) describing additional information about the codec. This package is agnostic to the names and values of the codec parameters supported by an implementation. The element is optional.

For example, a fragment with a <codec> element describing the H263 codec is as follows:

```
<codec name="video">
 <subtype>H263</subtype>
</codec>
```

A fragment where the <codec> element describes the H264 video codec with additional information about the profile level and packetization mode is as follows:

```
<codec name="video">
 <subtype>H264</subtype>
 <params>
  <param name="profile-level-id">42A01E</param>
  <param name="packetization-mode">0</param>
 </params>
</codec>
```

4.5. <params>

The <params> element is a container for <param> elements (Section 4.5.1).

The <params> element has no attributes, but the following child elements are defined (zero or more):

<param>: specifies a parameter name and value (Section 4.5.1).

4.5.1. <param>

The <param> element describes a parameter name and value.

The <param> element has the following attributes:

name: a string indicating the name of the parameter. The attribute is mandatory.

type: specifies a type indicating how the in-line value of the parameter is to be interpreted. A valid value is a MIME media type (see Section 4.7.6). The attribute is optional. The default value is "text/plain".

encoding: specifies a content-transfer-encoding schema applied to the in-line value of the parameter on top of the MIME media type specified with the 'type' attribute. A valid value is a content-transfer-encoding schema as defined by the "mechanism" token in Section 6.1 of [RFC2045]. The attribute is optional. There is no default value.

The <param> element content model is the value of the parameter. Note that a value that contains XML characters (e.g., "<") needs to be escaped following standard XML conventions.

4.6. Response Status Codes

This section describes the response codes in Table 1 for the 'status' attribute of mixer management <response> (Section 4.2.3) and <auditresponse> (Section 4.3.2). The MS MUST support the status response codes defined here. All other valid but undefined values are reserved for future use, where new status codes are assigned using the Standards Action process defined in [RFC5226]. The AS MUST treat any responses it does not recognize as being equivalent to the x00 response code for all classes. For example, if an AS receives an unrecognized response code of 499, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 (Syntax error) response code.

4xx responses are definite failure responses from a particular MS. The 'reason' attribute in the response SHOULD identify the failure in more detail, for example, "Mandatory attribute missing: id2 join element" for a 400 (Syntax error) response code.

The AS SHOULD NOT retry the same request without modification (for example, correcting a syntax error or changing the conferenceid to use one available on the MS). However, the same request to a different MS might be successful, for example, if another MS supports a capability required in the request.

4xx failure responses can be grouped into three classes: failure due to a syntax error in the request (400); failure due to an error executing the request on the MS (405-419); and failure due to the request requiring a capability not supported by the MS (420-435).

In cases where more than one request code could be reported for a failure, the MS SHOULD use the most specific error code of the failure class for the detected error. For example, if the MS detects that the conference identifier in the request is invalid, then it uses a 406 status code. However, if the MS merely detects that an execution error occurred, then 419 is used.

Code	Summary	Description	Informational: AS Possible Recovery Action
200	OK	request has succeeded.	
400	Syntax error	request is syntactically invalid: it is not valid with respect to the XML schema specified in Section 5 or it violates a co-occurrence constraint for a request element defined in Section 4.	Change the request so that it is syntactically valid.
405	Conference already exists	request uses an identifier to create a new conference (Section 4.2.1.1) that is already used by another conference on the MS.	Send an <audit> request (Section 4.3.1) requesting the list of conference mixer identifiers already used by the MS and then use a conference identifier that is not listed.
406	Conference does not exist	request uses an identifier for a conference that does not exist on the MS.	Send an <audit> request (Section 4.3.1) requesting the list of conference mixer identifiers used by the MS and then use a conference identifier that is listed.

407	Incompatible stream configuration	request specifies a media stream configuration that is in conflict with itself, the connection, or conference capabilities (see Section 4.2.2.2).	Change the media stream configuration to match the capabilities of the connection or conference.
408	Joining entities already joined	request attempts to create a join mixer (Section 4.2.2.2) where the entities are already joined.	Send an <audit> request (Section 4.3.1) requesting the list of join mixers on the MS and then use entities that are not listed.
409	Joining entities not joined	request attempts to manipulate a join mixer where the entities are not joined.	Send an <audit> request (Section 4.3.1) requesting the list of join mixers on the MS and then use entities that are listed.
410	Unable to join - conference full	request attempts to join a participant to a conference (Section 4.2.2.2) but the conference is already full.	
411	Unable to perform join mixer operation	request attempts to create, modify, or delete a join between entities but fails.	
412	Connection does not exist	request uses an identifier for a connection that does not exist on the MS.	

419	Other execution error	requested operation cannot be executed by the MS.	
420	Conference reservation failed	request to create a new conference (Section 4.2.1.1) failed due to unsupported reservation of talkers or listeners.	
421	Unable to configure audio mix	request to create or modify a conference failed due to unsupported audio mix.	
422	Unsupported media stream configuration	request contains one or more <stream> elements (Section 4.2.2.5) whose configuration is not supported by the MS.	
423	Unable to configure video layouts	request to create or modify a conference failed due to unsupported video layout configuration.	
424	Unable to configure video switch	request to create or modify a conference failed due to unsupported video switch configuration.	
425	Unable to configure codecs	request to create or modify a conference failed due to unsupported codec.	

426	Unable to join - mixing connections not supported	request to join connection entities (Section 4.2.2.2) failed due to lack of support for mixing connections.	
427	Unable to join - mixing conferences not supported	request to join conference entities (Section 4.2.2.2) failed due to lack of support for mixing conferences.	
428	Unsupported foreign namespace attribute or element	the request contains attributes or elements from another namespace that the MS does not support.	
435	Other unsupported capability	request requires another capability not supported by the MS.	

Table 1: Status Codes

4.7. Type Definitions

This section defines types referenced in attribute definitions.

4.7.1. Boolean

The value space of boolean is the set {true, false, 1, 0} as defined in Section 3.2.2 of [XMLSchema:Part2]. In accordance with this definition, the concept of false can be lexically represented by the strings "0" and "false" and the concept of true by the strings "1" and "true"; implementations MUST support both styles of lexical representation.

4.7.2. Non-Negative Integer

The value space of non-negative integer is the infinite set {0,1,2,...} as defined in Section 3.3.20 of [XMLSchema:Part2].

4.7.3. Positive Integer

The value space of positive integer is the infinite set {1,2,...} as defined in Section 3.3.25 of [XMLSchema:Part2].

4.7.4. String

A string in the character encoding associated with the XML element as defined in Section 3.2.1 of [XMLSchema:Part2].

4.7.5. Time Designation

A time designation consists of a non-negative real number followed by a time unit identifier.

The time unit identifiers are: "ms" (milliseconds) and "s" (seconds).

Examples include: "3s", "850ms", "0.7s", ".5s" and "+1.5s".

4.7.6. MIME Media Type

A string formatted as an IANA MIME media type [MIME.mediatypes]. The ABNF ([RFC5234]) production for the string is:

media-type = type-name "/" subtype-name *(";" parameter)

parameter = parameter-name "=" value

where "type-name" and "subtype-name" are defined in Section 4.2 of [RFC4288], "parameter-name" is defined in Section 4.3 of [RFC4288], and "value" is defined in Section 5.1 of [RFC2045].

4.7.7. Language Identifier

A language identifier labels information content as being of a particular human language variant. Following the XML specification for language identification [XML], a legal language identifier is identified by a [RFC5646] code and matched according to [RFC4647].

5. Formal Syntax

This section defines the XML schema for the Mixer Control Package. The schema is normative.

The schema defines datatypes, attributes, and mixer elements in the urn:ietf:params:xml:ns:msc-mixer namespace. In most elements, the order of child elements is significant. The schema is extensible: elements allow attributes and child elements from other namespaces.

Elements from outside this package's namespace can occur after elements defined in this package.

The schema is dependent upon the schema (framework.xsd) defined in Appendix A.1 of the Control Framework [RFC6230].

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:ietf:params:xml:ns:msc-mixer"
 xmlns:fw="urn:ietf:params:xml:ns:control:framework-attributes"
 elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="urn:ietf:params:xml:ns:msc-mixer"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:annotation>
  <xsd:documentation>
   IETF MediaCtrl Mixer 1.0 (20110104)

   This is the schema of the Mixer Control Package.  It
   defines request, response, and notification elements for
   mixing.

   The schema namespace is urn:ietf:params:xml:ns:msc-mixer

  </xsd:documentation>
 </xsd:annotation>

 <!--
  ################################################################

  SCHEMA IMPORTS

  ################################################################
 -->


 <xsd:import
  namespace="urn:ietf:params:xml:ns:control:framework-attributes"
  schemaLocation="framework.xsd">
  <xsd:annotation>
   <xsd:documentation>
    This import brings in the framework attributes for
    conferenceid and connectionid.
   </xsd:documentation>
  </xsd:annotation>
 </xsd:import>

 <!--
```

```
     ######################################################

     Extensible core type

     ######################################################
    -->


    <xsd:complexType name="Tcore">
     <xsd:annotation>
      <xsd:documentation>
       This type is extended by other (non-mixed) component types to
       allow attributes from other namespaces.
      </xsd:documentation>
     </xsd:annotation>
     <xsd:sequence/>
     <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>

    <!--
     ######################################################

     TOP-LEVEL ELEMENT: mscmixer

     ######################################################
    -->

    <xsd:complexType name="mscmixerType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:choice>
         <xsd:element ref="createconference" />
         <xsd:element ref="modifyconference" />
         <xsd:element ref="destroyconference" />
         <xsd:element ref="join" />
         <xsd:element ref="unjoin" />
         <xsd:element ref="modifyjoin" />
         <xsd:element ref="response" />
         <xsd:element ref="event" />
         <xsd:element ref="audit" />
         <xsd:element ref="auditresponse" />
         <xsd:any namespace="##other" minOccurs="0"
          maxOccurs="unbounded" processContents="lax" />
        </xsd:choice>
       </xsd:sequence>
       <xsd:attribute name="version" type="version.datatype"
        use="required" />
```

```
      <xsd:attribute name="desclang" type="xsd:language"
       default="i-default" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="mscmixer" type="mscmixerType" />

   <!--
    ######################################################

    CONFERENCE MANAGEMENT TYPES

    ######################################################
   -->

   <!--  createconference -->

   <xsd:complexType name="createconferenceType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:element ref="codecs" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="audio-mixing" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="video-layouts" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="video-switch" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="subscribe" minOccurs="0"
        maxOccurs="1" />
       <xsd:any namespace="##other"
        processContents="lax" minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
      <xsd:attribute name="conferenceid" type="xsd:string" />
      <xsd:attribute name="reserved-talkers"
       type="xsd:nonNegativeInteger" default="0" />
      <xsd:attribute name="reserved-listeners"
       type="xsd:nonNegativeInteger" default="0" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>


   <xsd:element name="createconference" type="createconferenceType" />

   <!--  modifyconference -->
```

```
    <xsd:complexType name="modifyconferenceType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="codecs" minOccurs="0"
         maxOccurs="1" />
        <xsd:element ref="audio-mixing" minOccurs="0"
         maxOccurs="1" />
        <xsd:element ref="video-layouts" minOccurs="0"
         maxOccurs="1" />
        <xsd:element ref="video-switch" minOccurs="0"
         maxOccurs="1" />
        <xsd:element ref="subscribe" />
        <xsd:any namespace="##other"
         processContents="lax" minOccurs="0" maxOccurs="unbounded" />
       </xsd:sequence>
       <xsd:attribute name="conferenceid" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="modifyconference" type="modifyconferenceType" />


    <!--  destroyconference -->

    <xsd:complexType name="destroyconferenceType">
    <xsd:complexContent>
      <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
      <xsd:attribute name="conferenceid" type="xsd:string"
      use="required" />
      </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="destroyconference"
     type="destroyconferenceType" />

    <!--
     ######################################################

     JOIN TYPES
```

```
     ######################################################
    -->

    <xsd:complexType name="joinType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="stream" minOccurs="0"
         maxOccurs="unbounded" />
        <xsd:any namespace="##other"
         processContents="lax" minOccurs="0" maxOccurs="unbounded" />
       </xsd:sequence>
       <xsd:attribute name="id1" type="xsd:string"
        use="required" />
       <xsd:attribute name="id2" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="join" type="joinType" />

    <xsd:complexType name="modifyjoinType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="stream" minOccurs="0"
         maxOccurs="unbounded" />
        <xsd:any namespace="##other"
         processContents="lax" minOccurs="0" maxOccurs="unbounded" />
       </xsd:sequence>
       <xsd:attribute name="id1" type="xsd:string"
        use="required" />
       <xsd:attribute name="id2" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="modifyjoin" type="modifyjoinType" />


    <xsd:complexType name="unjoinType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="stream" minOccurs="0"
         maxOccurs="unbounded" />
```

```
      <xsd:any namespace="##other"
       processContents="lax" minOccurs="0" maxOccurs="unbounded" />
     </xsd:sequence>
     <xsd:attribute name="id1" type="xsd:string"
      use="required" />
     <xsd:attribute name="id2" type="xsd:string"
      use="required" />
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="unjoin" type="unjoinType" />

  <!--
   ########################################################

   OTHER TYPES

   ########################################################
  -->

  <xsd:complexType name="eventType">
   <xsd:complexContent>
    <xsd:extension base="Tcore">
     <xsd:sequence>
      <xsd:choice>
       <xsd:element ref="active-talkers-notify"
        minOccurs="0" maxOccurs="1" />
       <xsd:element ref="unjoin-notify"
        minOccurs="0" maxOccurs="1" />
       <xsd:element ref="conferenceexit"
        minOccurs="0" maxOccurs="1" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:choice>
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="event" type="eventType" />

  <xsd:complexType name="activetalkersnotifyType">
   <xsd:complexContent>
    <xsd:extension base="Tcore">
     <xsd:sequence>
      <xsd:element ref="active-talker" minOccurs="0"
       maxOccurs="unbounded" />
```

```
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="conferenceid" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="active-talkers-notify"
     type="activetalkersnotifyType" />


    <xsd:complexType name="activetalkerType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attributeGroup ref="fw:framework-attributes" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="active-talker" type="activetalkerType" />


    <xsd:complexType name="unjoinnotifyType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="status" type="xsd:nonNegativeInteger"
         use="required" />
       <xsd:attribute name="reason" type="xsd:string" />
         <xsd:attribute name="desclang" type="xsd:language"/>
       <xsd:attribute name="id1" type="xsd:string"
        use="required" />
       <xsd:attribute name="id2" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="unjoin-notify" type="unjoinnotifyType" />
```

```
    <!--  conferenceexit-->

    <xsd:complexType name="conferenceexitType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="conferenceid" type="xsd:string"
        use="required" />
       <xsd:attribute name="status"
        type="xsd:nonNegativeInteger" use="required" />
       <xsd:attribute name="reason" type="xsd:string" />
         <xsd:attribute name="desclang" type="xsd:language"/>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="conferenceexit" type="conferenceexitType" />


    <xsd:complexType name="responseType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="status" type="status.datatype"
        use="required" />
       <xsd:attribute name="reason" type="xsd:string" />
         <xsd:attribute name="desclang" type="xsd:language"/>
       <xsd:attributeGroup ref="fw:framework-attributes" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>


    <xsd:element name="response" type="responseType" />

    <xsd:complexType name="subscribeType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="active-talkers-sub"
         minOccurs="0" maxOccurs="1" />
        <xsd:any namespace="##other" minOccurs="0"
```

```
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>


   <xsd:element name="subscribe" type="subscribeType" />

   <xsd:complexType name="activetalkerssubType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="interval"
       type="xsd:nonNegativeInteger" default="3" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="active-talkers-sub"
    type="activetalkerssubType" />


   <xsd:complexType name="streamType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:element ref="volume" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="clamp" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="region" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="priority" minOccurs="0"
        maxOccurs="1" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="media" type="media.datatype"
       use="required" />
      <xsd:attribute name="label" type="label.datatype" />
      <xsd:attribute name="direction"
       type="direction.datatype" default="sendrecv" />
     </xsd:extension>
    </xsd:complexContent>
```

```
    </xsd:complexType>

    <xsd:element name="stream" type="streamType" />


    <xsd:complexType name="volumeType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="controltype"
        type="volumecontroltype.datatype" use="required" />
       <xsd:attribute name="value" type="xsd:string" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="volume" type="volumeType" />


    <xsd:complexType name="clampType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="tones" type="xsd:string"
        default="1 2 3 4 5 6 7 8 9 0 * # A B C D"/>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="clamp" type="clampType" />


    <!--  region  -->
    <xsd:simpleType name="regionType">
     <xsd:restriction base="xsd:NMTOKEN" />
    </xsd:simpleType>

    <xsd:element name="region" type="regionType" />


    <!--  priority  -->
    <xsd:simpleType name="priorityType">
```

```
     <xsd:restriction base="xsd:positiveInteger" />
    </xsd:simpleType>

    <xsd:element name="priority" type="priorityType" />

    <xsd:complexType name="audiomixingType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
      <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="type" type="audiomix.datatype"
        default="nbest" />
       <xsd:attribute name="n" type="xsd:nonNegativeInteger"
        default="0" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="audio-mixing" type="audiomixingType" />

    <!-- video-switch -->

    <xsd:complexType name="videoswitchType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
         <xsd:choice>
          <xsd:element name="vas" type="Tcore"/>
          <xsd:element name="controller" type="Tcore"/>
          <xsd:any namespace="##other" processContents="lax" />
        </xsd:choice>
       </xsd:sequence>
       <xsd:attribute name="interval"
        type="xsd:nonNegativeInteger" default="3" />
       <xsd:attribute name="activespeakermix"
        type="xsd:boolean" default="false" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="video-switch" type="videoswitchType" />

    <!-- video-layouts -->

    <xsd:complexType name="videolayoutsType">
     <xsd:complexContent>
```

```
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="video-layout" minOccurs="0"
         maxOccurs="unbounded" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="video-layouts" type="videolayoutsType" />


    <!-- video-layout -->
    <xsd:complexType name="videolayoutType">
    <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:choice>
          <xsd:element name="single-view" type="Tcore"/>
          <xsd:element name="dual-view" type="Tcore"/>
          <xsd:element name="dual-view-crop" type="Tcore"/>
          <xsd:element name="dual-view-2x1" type="Tcore"/>
          <xsd:element name="dual-view-2x1-crop" type="Tcore"/>
          <xsd:element name="quad-view" type="Tcore"/>
          <xsd:element name="multiple-3x3" type="Tcore"/>
          <xsd:element name="multiple-4x4" type="Tcore"/>
          <xsd:element name="multiple-5x1" type="Tcore"/>
          <xsd:any namespace="##other" processContents="lax" />
        </xsd:choice>
       </xsd:sequence>
      <xsd:attribute name="min-participants"
        type="xsd:positiveInteger" default="1" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="video-layout" type="videolayoutType" />

    <xsd:complexType name="auditType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
      <xsd:sequence>
      <xsd:any namespace="##other" minOccurs="0"
       maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
       <xsd:attribute name="capabilities"
```

```
        type="xsd:boolean" default="true" />
      <xsd:attribute name="mixers" type="xsd:boolean"
       default="true" />
      <xsd:attribute name="conferenceid" type="xsd:string" />
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="audit" type="auditType" />


  <xsd:complexType name="auditresponseType">
   <xsd:complexContent>
    <xsd:extension base="Tcore">
     <xsd:sequence>
      <xsd:element ref="capabilities" minOccurs="0"
       maxOccurs="1" />
      <xsd:element ref="mixers" minOccurs="0"
       maxOccurs="1" />
      <xsd:any namespace="##other" minOccurs="0"
       maxOccurs="unbounded" processContents="lax" />
     </xsd:sequence>
     <xsd:attribute name="status" type="status.datatype"
      use="required" />
     <xsd:attribute name="reason" type="xsd:string" />
       <xsd:attribute name="desclang" type="xsd:language"/>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="auditresponse" type="auditresponseType" />

  <!-- mixers -->

  <xsd:complexType name="mixersType">
   <xsd:complexContent>
    <xsd:extension base="Tcore">
     <xsd:sequence>
      <xsd:element ref="conferenceaudit" minOccurs="0"
       maxOccurs="unbounded" />
      <xsd:element ref="joinaudit" minOccurs="0"
       maxOccurs="unbounded" />
      <xsd:any namespace="##other" minOccurs="0"
       maxOccurs="unbounded" processContents="lax" />
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
```

```
    <xsd:element name="mixers" type="mixersType" />

    <!--  joinaudit -->

    <xsd:complexType name="joinauditType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:any namespace="##other"
         processContents="lax" minOccurs="0" maxOccurs="unbounded" />
       </xsd:sequence>
       <xsd:attribute name="id1" type="xsd:string"
        use="required" />
       <xsd:attribute name="id2" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="joinaudit" type="joinauditType" />

    <!-- conferenceaudit -->

    <xsd:complexType name="conferenceauditType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="codecs" minOccurs="0"
         maxOccurs="1" />
        <xsd:element ref="participants" minOccurs="0"
         maxOccurs="1" />
        <xsd:element ref="video-layout" minOccurs="0"
         maxOccurs="1" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="conferenceid" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="conferenceaudit" type="conferenceauditType" />


    <!-- participants -->

    <xsd:complexType name="participantsType">
```

```
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="participant" minOccurs="0"
         maxOccurs="unbounded" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="participants" type="participantsType" />

    <!-- participant -->

    <xsd:complexType name="participantType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
       <xsd:attribute name="id" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="participant" type="participantType" />

    <!-- capabilities -->

    <xsd:complexType name="capabilitiesType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="codecs" minOccurs="1"
         maxOccurs="1" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="capabilities" type="capabilitiesType" />
```

```
    <!-- codecs -->

    <xsd:complexType name="codecsType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="codec" minOccurs="0"
         maxOccurs="unbounded" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="codecs" type="codecsType" />

    <!-- codec -->

    <xsd:complexType name="codecType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="subtype" minOccurs="1"
         maxOccurs="1" />
        <xsd:element ref="params" minOccurs="0"
         maxOccurs="1" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string"
        use="required" />
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="codec" type="codecType" />

    <!-- subtype -->

    <xsd:simpleType name="subtypeType">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>

    <xsd:element name="subtype" type="subtypeType" />


    <!-- params -->
```

```
    <xsd:complexType name="paramsType">
     <xsd:complexContent>
      <xsd:extension base="Tcore">
       <xsd:sequence>
        <xsd:element ref="param" minOccurs="0"
         maxOccurs="unbounded" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:sequence>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>

    <xsd:element name="params" type="paramsType" />


    <!--  param -->
       <!--  doesn't extend tCore since its content model is mixed -->
    <xsd:complexType name="paramType" mixed="true">
     <xsd:sequence/>
     <xsd:attribute name="name" type="xsd:string" use="required" />
     <xsd:attribute name="type" type="mime.datatype"
     default="text/plain" />
        <xsd:attribute name="encoding" type="xsd:string"/>
    </xsd:complexType>

    <xsd:element name="param" type="paramType" />


    <!--
     ########################################################

     DATATYPES

     ########################################################
    -->

   <xsd:simpleType name="version.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="1.0" />
     </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="eventname.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9\.]+" />
     </xsd:restriction>
    </xsd:simpleType>
```

```
    <xsd:simpleType name="audiomix.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="nbest" />
      <xsd:enumeration value="controller" />
     </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="media.datatype">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>

    <xsd:simpleType name="label.datatype">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>

    <xsd:simpleType name="status.datatype">
     <xsd:restriction base="xsd:positiveInteger">
      <xsd:pattern value="[0-9][0-9][0-9]" />
     </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="direction.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="sendonly" />
      <xsd:enumeration value="recvonly" />
      <xsd:enumeration value="sendrecv" />
      <xsd:enumeration value="inactive" />
     </xsd:restriction>
    </xsd:simpleType>


    <xsd:simpleType name="mime.datatype">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>


    <xsd:simpleType name="volumecontroltype.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="automatic" />
      <xsd:enumeration value="setgain" />
      <xsd:enumeration value="setstate" />
     </xsd:restriction>
    </xsd:simpleType>


   </xsd:schema>
```

Figure 10: Mixer Package XML Schema

6. Examples

This section provides examples of the Mixer Control Package.

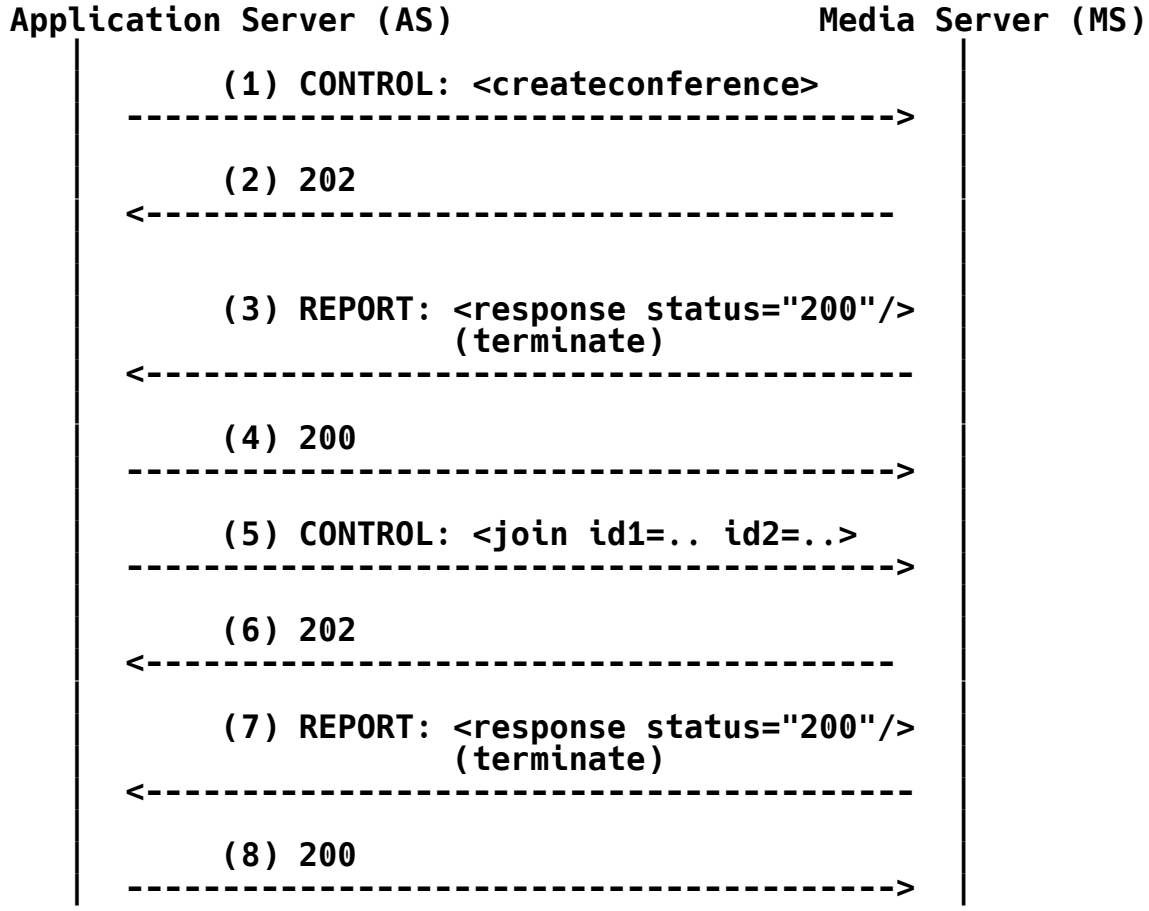
6.1. AS-MS Framework Interaction Examples

The following example assumes a Control Channel has been established and synced as described in the Media Control Channel Framework ([RFC6230]).

The XML messages are in angled brackets (with the root <mscmixer> and other details omitted for clarity); the REPORT status is in parentheses. Other aspects of the protocol are omitted for readability.

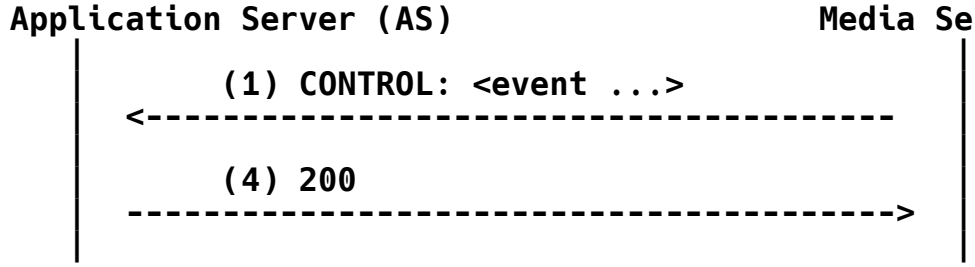
6.1.1. Creating a Conference Mixer and Joining a Participant

A conference mixer is created successfully and a participant is joined.



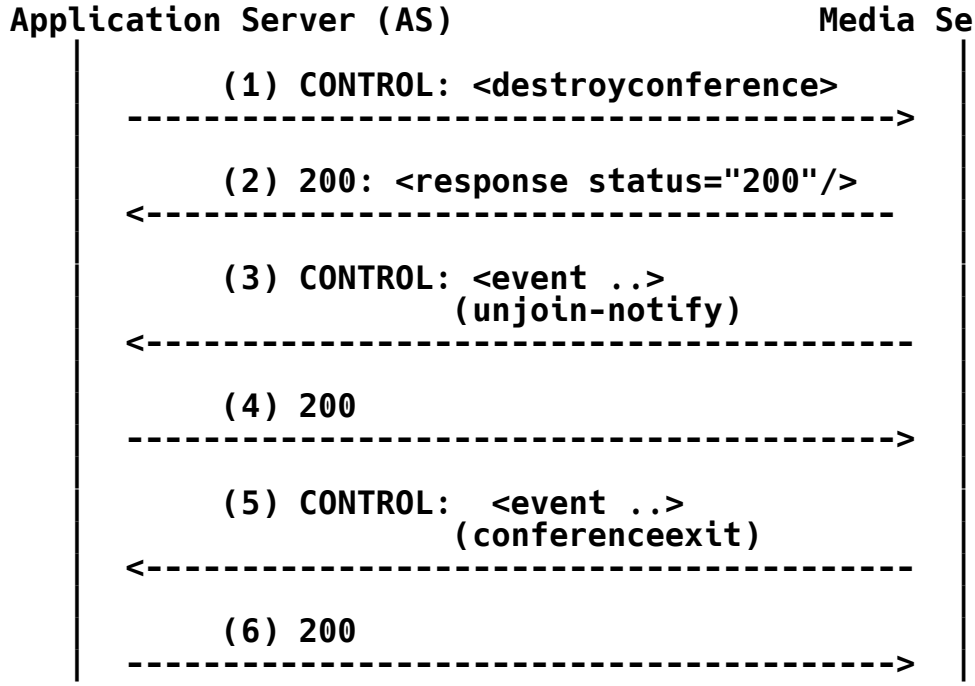
6.1.2. Receiving Active Talker Notifications

An active talker notification event is sent by the MS.



6.1.3. Conference Termination

The MS receives a request to terminate the conference, resulting in conferenceexit and participant unjoined notifications.



6.2. Mixing Examples

The following examples show how the mixing package can be used to create audio conferences, bridge connections, and video conferences.

The examples do not specify all messages between the AS and MS.

6.2.1. Audio Conferencing

The AS sends a request to create a conference mixer:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <createconference conferenceid="conf1"
  reserved-talkers="2" reserved-listeners="3">
  <audio-mixing type="nbest"/>
  <subscribe>
   <active-talkers-sub interval="5"/>
  </subscribe>
 </createconference>
</mscmixer>
```

The request specifies that the conference is assigned the conference id "conf1" and is configured with 2 reserved talkers, 3 reserved listener slots, audio-mixing policy set to nbest, and with active talkers notifications set to 5 seconds.

If the MS is able to create this conference mixer, it sends a 200 response:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <response status="200" reason="conference created"
           conferenceid="conf1"/>
</mscmixer>
```

The AS is now able to join connections to the conference as participants. A participant able to contribute to the audio mix would be joined as follows:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="1536067209:913cd14c" id2="conf1">
  <stream media="audio" direction="sendrecv"/>
 </join>
</mscmixer>
```

If the MS can join the participant 1536067209:913cd14c to the conference conf1 with audio in both directions, then it sends a successful response:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <response status="200" reason="join successful"/>
</mscmixer>
```

The AS could also join listener-only participants to the conference by setting the stream direction to receive only:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="9936067209:914cd14c" id2="conf1">
  <stream media="audio" direction="recvonly"/>
 </join>
</mscmixer>
```

If the MS can join the participant 9936067209:914cd14c to the conference conf1, then it would send a successful response (not shown).

As the active talker changes, the MS sends an active talker notification to the AS:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <event>
  <active-talkers-notify conferenceid="conf1">
   <active-talker connectionid="1536067209:913cd14c"/>
  </active-talkers-notify>
 </event>
</mscmixer>
```

The AS could decide to change the status of a talker connection so that they can only listen:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <modifyjoin id1="1536067209:913cd14c" id2="conf1">
  <stream media="audio" direction="recvonly"/>
 </modifyjoin>
</mscmixer>
```

Where the participant 1536067209:913cd14c is no longer able to contribute to the audio mix on the conference. If the MS is able to execute this request, it would send a 200 response.

The AS could decide to remove this participant from the conference:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <unjoin id1="1536067209:913cd14c" id2="conf1"/>
</mscmixer>
```

Again, if the MS can execute this request, a 200 response would be sent.

Finally, the AS terminates the conference:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <destroyconference conferenceid="conf1"/>
</mscmixer>
```

If the MS is able to destroy the conference conf1, it sends a 200 response:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <response status="200" conferenceid="conf1"/>
</mscmixer>
```

For each participant attached to the conference when it is destroyed, the MS sends an unjoin notification event:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <event>
  <unjoin-notify status="2" id1="9936067209:914cd14c"
      id2="conf1"/>
 </event>
</mscmixer>
```

And the MS sends a conferenceexit notification event when the conference finally exits:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <event>
  <conferenceexit status="0" conferenceid="conf1"/>
 </event>
</mscmixer>
```

6.2.2. Bridging Connections

The mixer package can be used to join connections to one another. In a call-center scenario, for example, this package can be used to set up and modify connections between a caller, agent, and supervisor.

A caller is joined to an agent with bidirectional audio:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="caller:001" id2="agent:002">
  <stream media="audio" direction="sendrecv"/>
 </join>
</mscmixer>
```

If the MS is able to establish this connection, then it would send a 200 response:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <response status="200"/>
</mscmixer>
```

Now assume that the AS wants a supervisor to listen into the agent conversation with the caller and provide whispered guidance to the agent. First, the AS would send a request to join the supervisor and the caller connections:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join  id1="supervisor:003" id2="caller:001">
  <stream media="audio" direction="recvonly"/>
 </join>
</mscmixer>
```

If this request was successful, audio output from the caller connection would now be sent to both the agent and the supervisor.

Second, the AS would send a request to join the supervisor and the agent connections:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <join id1="supervisor:001" id2="agent:002">
  <stream media="audio" direction="sendrecv"/>
 </join>
</mscmixer>
```

If this request was successful, the audio mixing would occur on both the agent and supervisor connections: the agent would hear the caller and supervisor, and the supervisor would hear the agent and caller. The caller would only hear the agent. If the MS is unable to join and mix connections in this way, it would send a 426 response.

6.2.3. Video Conferencing

In this example, an audio-video conference is created where the loudest participant has the most prominent region in the video layout.

The AS sends a request to create an audio-video conference:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <createconference conferenceid="conf2">
  <audio-mixing type="nbest"/>
  <video-layouts>
   <video-layout min-participants="1"><single-view/></video-layout>
   <video-layout min-participants="2"><dual-view/></video-layout>
   <video-layout min-participants="3"><quad-view/></video-layout>
   <video-layout min-participants="5"><multiple-5x1/></video-layout>
  </video-layouts>
  <video-switch><vas/></video-switch>
 </createconference>
</mscmixer>
```

In this configuration, the conference uses a nbest audio mixing policy and a <vas/> video-switching policy, so that the loudest speaker receives the most prominent region in the layout. Multiple video layouts are specified and the active one depends on the number of participants.

Assume that 4 participants are already joined to the conference. In that case, the video layout will be quad-view (Figure 6) with the most active speaker displayed in region 1. When a fifth participant joins, the video layout automatically switches to a multiple-5x1 layout (Figure 9), again with the most active speaker in region 1.

The AS can manipulate which participants are displayed in the remaining regions. For example, it could force an existing conference participant to be displayed in region 2:

```
<mscmixer version="1.0" xmlns="urn:ietf:params:xml:ns:msc-mixer">
 <modifyjoin id1="1536067209:913cd14c" id2="conf2">
  <stream media="video">
   <region>2</region>
  </stream>
 </modifyjoin>
</mscmixer>
```

7. Security Considerations

As this Control Package processes XML markup, implementations MUST address the security considerations of [RFC3023].

As a Control Package of the Media Control Channel Framework, security, confidentiality, and integrity of messages transported over the Control Channel MUST be addressed as described in Section 12 of the Media Control Channel Framework ([RFC6230]), including transport-level protection, Control Channel policy management, and session establishment. In addition, implementations MUST address security,

confidentiality, and integrity of User Agent sessions with the MS, both in terms of SIP signaling and the associated RTP media flow; see [RFC6230] for further details on this topic.

Adequate transport protection and authentication are critical, especially when the implementation is deployed in open networks. If the implementation fails to correctly address these issues, it risks exposure to malicious attacks, including (but not limited to):

Denial of Service: An attacker could insert a request message into the transport stream causing specific conferences or join mixers on the MS to be destroyed. For example, <destroyconference conferenceid="XXXX">, where the value of "XXXX" could be guessed or discovered by auditing active mixers on the MS using an <audit> request. Likewise, an attacker could impersonate the MS and insert error responses into the transport stream thereby denying the AS access to package capabilities.

Resource Exhaustion: An attacker could insert into the Control Channel new request messages (or modify existing ones) with, for instance, <createconference> elements causing large numbers of conference mixer resources to be allocated. At some point, this will exhaust the number of conference mixers that the MS is able to allocate.

The Media Control Channel Framework permits additional policy management (beyond that specified for the Media Control Channel Framework), including resource access and Control Channel usage, to be specified at the Control Package level. (See Section 12.3 of [RFC6230].)

Since creation of conference and join mixers is associated with media-mixing resources on the MS, the security policy for this Control Package needs to address how such mixers are securely managed across more than one Control Channel. Such a security policy is only useful for secure, confidential, and integrity-protected channels. The identity of Control Channels is determined by the channel identifier, i.e., the value of the 'cfw-id' attribute in the SDP and Dialog-ID header in the channel protocol (see [RFC6230]). Channels are the same if they have the same identifier; otherwise, they are different. This Control Package imposes the following additional security policies:

Responses: The MS MUST only send a response to a mixer management or audit request using the same Control Channel as the one used to send the request.

Notifications: The MS MUST only send notification events for conference and join mixers using the same Control Channel as it received the request creating the mixer.

Auditing: The MS MUST only provide audit information about conference and join mixers that have been created on the same Control Channel as the one upon which the <audit> request is sent. For example, if a join between two connections has been created on one channel, then a request on another channel to audit all mixers -- <audit mixers="true"/> -- would not report on this join mixer.

Rejection: The MS SHOULD reject requests to audit or manipulate an existing conference or join mixer on the MS if the channel is not the same as the one used when the mixer was created. The MS rejects a request by sending a Control Framework 403 response (see Sections 7.4 and 12.3 of [RFC6230]). For example, if a channel with identifier 'cfw1234' has been used to send a request to create a particular conference and the MS receives on channel 'cfw98969' a request to audit or destroy this particular conference, then the MS sends a Control Framework 403 response.

There can be valid reasons why an implementation does not reject an audit or mixer manipulation request on a different channel from the one that created the mixer. For example, a system administrator might require a separate channel to audit mixer resources created by system users and to terminate mixers consuming excessive system resources. Alternatively, a system monitor or resource broker might require a separate channel to audit mixers managed by this package on a MS. However, the full implications need to be understood by the implementation and carefully weighed before accepting these reasons as valid. If the reasons are not valid in their particular circumstances, the MS rejects such requests.

There can also be valid reasons for 'channel handover' including high availability support or when one AS needs to take over management of mixers after the AS that created them has failed. This could be achieved by the Control Channels using the same channel identifier, one after another. For example, assume a channel is created with the identifier 'cfw1234', and the channel is used to create mixers on the MS. This channel (and associated SIP dialog) then terminates due to a failure on the AS. As permitted by the Control Framework, the channel identifier 'cfw1234' could then be reused so that another channel is created with the same identifier 'cfw1234', allowing it to 'take over' management of the mixers on the MS. Again, the implementation needs to understand the full implications and carefully weigh them before accepting these reasons as valid. If the reasons are not valid for their particular circumstances, the MS uses

the appropriate SIP mechanisms to prevent session establishment when the same channel identifier is used in setting up another Control Channel (see Section 4 of [RFC6230]).

8. IANA Considerations

Per this specification, IANA has registered a new Media Control Channel Framework Package, a new XML namespace, a new XML schema, and a new MIME type.

IANA has further created a new registry for the response codes for the MEDIACTRL Mixer Control Package, RFC 6505.

8.1. Control Package Registration

This section registers a new Media Control Channel Framework package, per the instructions in Section 13.1 of [RFC6230].

```
To: ietf-sip-control@iana.org
Subject: Registration of new Channel Framework package
Package Name: msc-mixer/1.0
Published Specification(s): RFC 6505
Person & email address to contact for further information:
   IETF MEDIACTRL working group (mediactrl@ietf.org),
   Scott McGlashan (smcg.stds01@mcglashan.org).
```

8.2. URN Sub-Namespace Registration

This section registers a new XML namespace, "urn:ietf:params:xml:ns:msc-mixer", per the guidelines in RFC 3688 [RFC3688].

```
URI: urn:ietf:params:xml:ns:msc-mixer
Registrant Contact:
   IETF MEDIACTRL working group (mediactrl@ietf.org),
   Scott McGlashan (smcg.stds01@mcglashan.org).
```

```
         XML:
            BEGIN
            <?xml version="1.0"?>
            <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
             <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
              <head>
               <title>Media Control Channel Framework Mixer
                      Package attributes</title>
              </head>
              <body>
               <h1>Namespace for Media Control Channel
                   Framework Mixer Package attributes</h1>
               <h2>urn:ietf:params:xml:ns:msc-mixer</h2>
            <p>See <a href="http://www.rfc-editor.org/rfc/rfc6505.txt">
            RFC 6505</a>.</p>
              </body>
             </html>
            END
```

8.3. XML Schema Registration

This section registers an XML schema as per the guidelines in RFC 3688 [RFC3688].

```
   URI:  urn:ietf:params:xml:schema:msc-mixer
   Registrant Contact:
      IETF MEDIACTRL working group (mediactrl@ietf.org),
      Scott McGlashan (smcg.stds01@mcglashan.org).
   Schema:  The XML for this schema can be found in
      Section 5 of this document.
```

8.4. MIME Media Type Registration for 'application/msc-mixer+xml'

This section registers the "application/msc-mixer+xml" MIME type.

```
   To:  ietf-types@iana.org
   Subject:  Registration of MIME media type
             application/msc-mixer+xml
   MIME media type name:  application
   MIME subtype name:  msc-mixer+xml
   Required parameters:  (none)
   Optional parameters:  charset
      Indicates the character encoding of enclosed XML.  Default is
      UTF-8.
   Encoding considerations:  Uses XML, which can employ 8-bit
      characters, depending on the character encoding used.  See RFC
      3023 [RFC3023], Section 3.2.
```

Security considerations: No known security considerations outside of those provided by the Media Control Channel Framework Mixer Package.
Interoperability considerations: This content type provides constructs for the Media Control Channel Framework Mixer Package.
Published specification: RFC 6505
Applications that use this media type: Implementations of the Media Control Channel Framework Mixer package.
Additional Information:
Magic Number(s): (none)
File extension(s): (none)
Macintosh File Type Code(s): (none)
Person & email address to contact for further information:
Scott McGlashan <smcg.stds01@mcglashan.org>
Intended usage: LIMITED USE
Author/Change controller: The IETF
Other information: None.

8.5. Mixer Control Package Status Code Registration

This section creates an IANA registry for the response codes for the MEDIACTRL Mixer Control Package. New status codes are assigned using the Standards Action process defined in RFC 5226 [RFC5226]. The initial population of the registry is defined in Section 4.6.

The format of this registry is as follows:

Code	Summary	Description	Reference
status code number	brief summary of the status code	full description of the status code	reference document defining the status code

Table 2: Fields for Mixer Control Package Status Code Registry

9. Contributors

Asher Shiratzky provided valuable support and contributions to early draft versions of this document.

The authors would like to thank the Mixer design team consisting of Roni Even, Lorenzo Miniero, Adnan Saleem, Diego Besprosvan, and Mary Barnes who provided valuable feedback, input, diagrams, and text to this document.

10. Acknowledgments

The authors would like to thank Steve Buko and Stephane Bastien for expert reviews of this work.

Shawn Emery carried out a thorough security review.

11. References

11.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3023] Murata, M., St. Laurent, S., and D. Kohn, "XML Media Types", RFC 3023, January 2001.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC4574] Levin, O. and G. Camarillo, "The Session Description Protocol (SDP) Label Attribute", RFC 4574, August 2006.

[RFC4647] Phillips, A. and M. Davis, "Matching of Language Tags", BCP 47, RFC 4647, September 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

[RFC6230] Boulton, C., Melanchuk, T., and S. McGlashan, "Media Control Channel Framework", RFC 6230, May 2011.

[RFC6501] Novo, O., Camarillo, G., Morgan, D., and J. Urpalainen, "Conference Information Data Model for Centralized Conferencing (XCON)", RFC 6501, March 2012.

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

[XMLSchema:Part2] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation, October 2004.

11.2. Informative References

[IANA] IANA, "RTP Payload Types", <http://www.iana.org/assignments/rtp-parameters>.

[MIME.mediatypes] IANA, "MIME Media Types", <http://www.iana.org/assignments/media-types>.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC5022] Van Dyke, J., Burger, E., and A. Spitzer, "Media Server Control Markup Language (MSCML) and Protocol", RFC 5022, September 2007.

[RFC5167] Dolly, M. and R. Even, "Media Server Control Protocol Requirements", RFC 5167, March 2008.

[RFC5707] Saleem, A., Xin, Y., and G. Sharratt, "Media Server Markup Language (MSML)", RFC 5707, February 2010.

Authors' Addresses

Scott McGlashan
Hewlett-Packard

EMail: smcg.stds01@mcglashan.org

Tim Melanchuk
Rainwillow

EMail: timm@rainwillow.com

Chris Boulton
NS-Technologies

EMail: chris@ns-technologies.com