Network Working Group   C. Boulton
Internet-Draft   Ubiquity Software Corporation
Intended status: Standards Track   T. Melanchuk
Expires: May 5, 2008   BlankSpace
                                                      S. McGlashan
                                                   Hewlett-Packard
                                                      A. Shiratzky
                                                         Radvision
                                                  November 2, 2007

   A Conference Control Package for the Session Initiation Protocol (SIP)
            draft-boulton-conference-control-package-03

Status of this Memo

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

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

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

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

   This Internet-Draft will expire on May 5, 2008.

Copyright Notice

Abstract

   This document defines a Session Initiation (SIP) Control Package for
   Conference Control. The control of Media Servers and their related
   resources in decomposed network architectures plays an important role
   in various Next Generation Networks. This Control Package aims to
   fulfill Conferencing requirements using the SIP Control Framework.

Table of Contents

1. Introduction

   The SIP Control Framework [I-D.boulton-sip-control-framework]
   provides a generic template for establishment and reporting
   capabilities of remotely initiated commands. The Framework utilizes
   many functions provided by the Session Initiation Protocol [RFC3261]
   (SIP) 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 control of conference instances.

2. **Conventions and Terminology**

   In this document, BCP 14/RFC 2119 [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 a media server, which the AS may 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: A MS Conference provides the media related mixing
      resources and services for conferences. In this document, A MS
      Conference is often referred to simply as a conference.

   MS Connection: A Media Server 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 may be audio or
      video and may be bi-directional or uni-directional.

3. Overview

The SIP Control Framework [I-D.boulton-sip-control-framework]
provides a generic approach for establishment and reporting
capabilities of remotely initiated commands. The Framework utilizes
many functions provided by the Session Initiation Protocol [RFC3261]
(SIP) 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 basic conferencing.

The scope of the package is control of media server functions for
conferencing (e.g. create a conference, add and remove participants
from it, etc) as well as responses and notifications related to these
functions. Although dialog services such as announcements,
recordings, and prompts are generally needed for a complete
conference service, those functions are defined in complementary
packages.

4. **Control Package Definition**

   This section fulfils the mandatory requirement for information that
   MUST be specified during the definition of a Control Framework
   Package, as detailed in Section 9 of
   [I-D.boulton-sip-control-framework].

4.1. **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-conf-audio/1.0" (Media Server Control -
   Conferencing - Audio - version 1.0).

4.2. **Framework Message Usage**

   The intent for the Conference Control package is for the creation,
   manipulation and deletion of conferences as well as joining,
   manipulation and unjoining of participants to conferences. The
   Conference Control package is intended for use in an architecture
   where application logic and media mixing are distributed. For this
   reason the Control Framework will operate in a manner where CONTROL
   messages, as defined in the Conference Framework
   [I-D.boulton-sip-control-framework], MUST only be sent from the
   network entity providing the application logic.

   This package defines XML elements in Section 5 and provides an XML
   Schema in Section 7.

   The XML elements in this package are split into requests, responses
   and event notifications. Requests are carried in CONTROL message
   bodies; <createconference> and <join> are examples of package
   requests. See Section 5.1 for a complete enumeration of requests.
   Responses are carried either in REPORT messages or Control Framework
   200 response bodies; the <response> element is defined as a package
   response. Event notifications are also carried in REPORT message
   bodies; the <event> element is defined for package event
   notifications.

   Note that package responses are different from framework response
   codes. Framework error response codes (see Section 8 of
   [I-D.boulton-sip-control-framework]) are used when the request or
   notification is invalid; for example, a request with invalid XML
   (400), or not understood (500). Package responses are carried in 200
   response or REPORT message bodies. This package's response codes are
   defined in Section 5.13.1.

   The schema uses "connection-id" and "conf-id" attributes which are

imported from schema defined in core Control Framework
[I-D.boulton-sip-control-framework].

4.3. 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 Section 16.1 of
[I-D.boulton-sip-control-framework] MUST be supported for media
dialogs and conferences.

4.4. CONTROL Message Body

A valid CONTROL body message MUST conform to the schema defined in
Section 7 and described in Section 5. XML messages appearing in
CONTROL messages MUST contain <createconference>, <modifyconference>,
<destroyconference>, <join>, <modifyjoin> or <unjoin> request
elements (Section 5.1).

4.5. REPORT Message Body

A valid REPORT body MUST conform to the schema defined in Section 7
and described in Section 5. XML messages appearing in REPORT
messages MUST contain a <response> (Section 5.13), or a
(notification) <event> element (Section 5.8).

5. Element Definitions

   This section defines the XML messages for this control package.

   [Editors Note: since XML Schema may not be able to express all
   constraints expressed in these definitions, in cases where there is a
   difference in constraints, the definitions in the section take
   priority.]

5.1. Requests

   The following request elements are defined:

   <createconference>: create and configure a new conference - see
      Section 5.2 for detail.

   <modifyconference>: modify the configuration of an existing
      conference - see Section 5.3 for detail.

   <destroyconference>: destroy an existing conference - see
      Section 5.4 for detail.

   <join>: create and configure media streams between connections
      and/or conferences (for example, add a participant to a
      conference) - see Section 5.5 for detail.

   <modifyjoin>: modify the configuration of a joined media stream -
      see Section 5.6 for detail.

   <unjoin>: delete a media stream (for example, remove a participant
      from a conference) - see Section 5.7 for detail.

5.2. <createconference>

   <createconference> is used in a request by the AS to create a new
   conference (multiparty) instance.

   The <createconference> element has the following child elements
   defined:

   <audio-mixing>: an element to configure the audio mixing
      characteristics of a conference (see Section 5.11.1 for more
      detail). The element is mandatory in a <createconference>
      request.

   <subscribe>: an element to request subscription to conference
      events. (see Section 5.10 for more detail). The element optional.

   <reserved-talkers>: element represents the number of guaranteed
      speaker slots to be reserved for the conference. The element is
      optional.

   <reserved-listeners>: element represents the number of guaranteed
      listener slots to be reserved for the conference. The element is
      optional.

   A media server MUST configure the audio mix of a conference as
   specified by the "<audio-mixing>" element or it MUST NOT create the
   conference and MUST report a 405 'Unable to configure audio mix'
   package level error. A media server must create any subscriptions
   requested by the "<subscribe>:" element or it MUST NOT create the
   conference and MUST report a 406 'Unable to create subscription'
   package level error. A media server MUST establish the reservations
   requested by any "<reserved-talkers>:" and "<reserved-listeners>:"
   elements or it MUST NOT create the conference and MUST report a 407
   'Conference reservation failed' package level error.

   The <createconference> element has the following attributes:

   conf-id: string indicating a unique name for the new conference. If
      this attribute is not specified, the MS creates a unique name for
      the conference. The value is used in subsequent references to the
      conference (e.g. as conf-id in a <response>). When present in a
      <createconference> request the new value of this attribute MUST be
      unique or else a 403 'Conference already exists' package level
      error will be reported. The attribute is optional.

   Conferences SHOULD support subscription to the following events:

      active-talkers: the list of zero or more speakers that have been
      active during the previous interval. The input and output
      parameters for event subscription and notification are given in
      the following tables.

   +-----------+-----------+-------------------------+------------+
   | Name | Direction | Description | Definition |
   +-----------+-----------+-------------------------+------------+
interval	input	minimum interval	Table 2
speaker	output	an active talker	Table 3
   +-----------+-----------+-------------------------+------------+

                   Active-Talker Event Parameters

+-------------+--+
| Name | Interval |
+-------------+--+
Description	the minimum amount of time (in seconds) that must
	elapse before further talker-events can be
	generated
Direction	input
Type	A valid TimeDesignation value.
Optional	No
Possible	A valid TimeDesignation value. A value is "0"
Values	suppresses further notifications.
Default	none
+-------------+--+

                         Table 2: Interval

+-------------+--+
| Name | Speaker |
+-------------+--+
Description	The connection identifier of a speaker that has
	been active during the preceding interval.
Direction	output
Type	a connection identifier: see Section 16.1 of
	[I-D.boulton-sip-control-framework]
Optional	No
Possible	Any connection identifier that is currently a
Values	member of the conference
Default	none
+-------------+--+

                         Table 3: Speaker

   Subscribing to events is described in Section 5.10 and the
   notification of events is described in Section 5.8

   For example, a request to create a conference:

```
<createconference conf-id="conference11">
    <audio-mixing mix-type="nbest"/>
</createconference>
```

When a MS has finished processing a <createconference> request, it
MUST reply with an appropriate <response> element (Section 5.13).

5.3. <modifyconference>

<modifyconference> is used by the controlling client to modify
properties of an existing conference.

The <modifyconference> element has the following child elements
defined:

<audio-mixing>: an element to configure the conference (see
   Section 5.11.1 for more detail). The element is optional in a
   <modifyconference> request.

<subscribe>: an element to request subscription to conference
   events. (see Section 5.10 for more detail). The element optional.

One or both of the elements <audio-mixing> and <subscribe> SHOULD be
present in a <modifyconference> request.

A media server MUST configure the audio mix of a conference as
specified by any "<audio-mixing>" element or it MUST NOT modify
anything and MUST report a 405 'Unable to configure audio mix'
package level error. A media server must create any subscriptions
requested by the "<subscribe>:" element or it MUST NOT modify
anything and MUST report a 406 'Unable to create subscription'
package level error.

The <modifyconference> element has the following attributes:

conf-id: string indicating the name of the conference to modify.
   The conference MUST be known by the receiving entity or else a 404
   'Conference does not exist' package level error will be generated.
   This attribute is mandatory.

When a MS has finished processing a <modifyconference> request, it
MUST reply with an appropriate <response> element (Section 5.13).

[Editors Note: TODO - need to cover who to respond if modify is not
successful]

5.4. <destroyconference>

<destroyconference> is used by the controlling client to destroy an
existing conference.

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

The <destroyconference> element has the following attributes:

conf-id: string indicating the name of the conference to destroy.
   The conference MUST be known by the receiving entity or else a 404
   'Conference does not exist' package level error will be generated.
   This attribute is mandatory.

When a MS has finished processing a <destroyconference> request, it
MUST reply with an appropriate <response> element (Section 5.13).
Successfully destroying the conference will result in a 200 package
level response to the request. Package level error responses do not
result in the conference being destroyed.

5.5. <join>

<join> is used by the controlling AS to create one or more media
streams either between a connection and a conference, between
connections, or between conferences. Streams may be audio or video
and may be bi-directional or uni-directional. A bi-directional
stream is implicitly composed of two uni-directional streams that can
be manipulated independently. The streams to be established are
specified by child <stream> elements (see Section 5.12).

A MS MUST support <join>ing a participant connection to a conference.
It MAY support <join>ing one connection to another connection, or one
conference to another conference.

Connections are created and destroyed on a media server using the
Session Initiation Protocol [RFC3261] (SIP).

The <join> element has the following child elements defined:

<stream>:   an element that both identifies the media streams to join
   and defines the way that they are to be joined (see Section 5.12).
   The element is optional. If no <stream> elements are specified,
   then the default is the media configuration of the connection or
   conference.

One or more <stream> elements may be specified so that individual
media streams can be controlled independently; for example, audio
only for transmission, but video only for reception. It is an error

if a <stream> element is in conflict with (a) another <stream>
element, (b) with dialog, connection or conference media
capabilities, or (c) with a SDP label value as part of the
connection-id (see Section 16.1 of
[I-D.boulton-sip-control-framework]).

The two entities to join are specified by the attributes of <join>.
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 Section 16.1 of
   [I-D.boulton-sip-control-framework] The attribute is mandatory.

id2: an identifier for either a connection or a conference. The
   identifier MUST conform to the syntax defined in Section 16.1 of
   [I-D.boulton-sip-control-framework] The attribute is mandatory.

Note: Section 16.1 of [I-D.boulton-sip-control-framework] 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.

When an controlling client has finished processing a <join> request,
it MUST reply with an <response> element (Section 5.13). If the
participant is already a member of the conference instance, a xxx
package level response should be generated.

5.5.1. Joining Model

The <join> operation creates a media stream between a connection and
a conference, between connections, or between conferences. This sub-
section describes the model of conferences and connections and
specifies the behaviour 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 mix 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
conferences available inputs and/or outputs, to the maximum number of
supported participants.

A connection is the termination of a RTP session(s) 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 may feed
several different inputs such as both a conference mix and a
recording of that participants audio.

Joining two connections which are 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 may 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 pre-empt the
conference media.

Another common case is the call centre coaching scenario where a
supervisor can listen to the conversation between an agent and a
customer, and provide hints to the agent, which 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 to 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 straight forward 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. A
controlling AS MUST use the <createconference> request for generic
conferences where the complex mixing algorithm is required.

Specifications which extend this package to handle additional media
types such as text or video, 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.

5.6. <modifyjoin>

An AS uses <modifyjoin> to change the configuration of media
stream(s) that were previously established between a connection and a
conference, between two connections, or between two conferences.

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

The <modifyjoin> element has the following child elements defined:

<stream>:   an element that both identifies the media streams to
   modify and defines the way that each stream should now be
   configured (see Section 5.12). The element is mandatory.

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 Section 16.1 of
   [I-D.boulton-sip-control-framework] The attribute is mandatory.

id2: an identifier for either a connection or a conference. The
   identifier MUST conform to the syntax defined in Section 16.1 of
   [I-D.boulton-sip-control-framework] The attribute is mandatory.

The media server MUST configure the streams that are included within
<modifyjoin> to that stated by the child elements. It MUST NOT
change the configuration of any streams not included as child
elements.

When an MS has finished processing a <modifyjoin> request, it MUST
reply with an appropriate <response> element (Section 5.13).

5.7. <unjoin>

An AS uses <unjoin> to remove previously established media stream(s)
from between a connection and a conference, between two connections,
or between two conferences.

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

The <unjoin> element has the following child elements defined:

<stream>:   an element that identifies the media stream(s) to remove
    (see Section 5.12). The element is optional. When not present,
    all streams between "id1" and "id2" are removed.

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 Section 16.1 of
    [I-D.boulton-sip-control-framework] The attribute is mandatory.

id2: an identifier for either a connection or a conference. The
    identifier MUST conform to the syntax defined in Section 16.1 of
    [I-D.boulton-sip-control-framework] The attribute is mandatory.

When an MS has successfully received a <unjoin> request, it MUST
reply with a successful <response> element (Section 5.13). If the
participant is not identified as a member of the conference instance,
a xxx package level response should be generated.

5.8. Notifications

Event notifications are specified in an <event> element.

5.8.1. <event>

Conference event notifications are either manual or automatic.

Manual event notifications are defined when an controlling client
subscribes to notifications for conference events using the
<subscribe> element of <createconference> or <modifyconference>. For
information on the <subscribe> element, see Section 5.10.

Automatic event notifications are defined as part of a Control
Package. The controlling client SHOULD NOT subscribe to these event
notifications. The MS MUST support these event notifications. This
package defines one automatic notification event: "conferenceexit"
which indicates that a conference has terminated. Note that this
notification is not sent if the conference has been destroyed by the
AS using a <destroyconference/> request.

The <event> element has the following attributes:

name: string indicating the name of conference event. The string is
    restricted to a sequence of alphanumeric or "." characters. The
    attribute is mandatory.

conferenceid: string identifying the conference. The attribute is
      mandatory.

   The <event> element has the following child element:

   <data>: an XML data structure (see Section 5.9) to pass additional
      information about the conference event. The element is optional.

5.9. <data>

   The <data> element is a general container for parameterized data.

   The <data> element has no attributes, but has the following child
   elements defined:

   <item>: contains the following attributes:

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

      value: a string indicating the value of the parameter. Multiple
         values of a parameters can be specified using space separation.
         The attribute is mandatory.

   Multiple <item> elements may be specified.

5.10. <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 5.8.1).

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

   <notify>: contains the following attributes:

      name: a string indicating the name of the event to be notified.
         The attribute is mandatory.

   The <notify> element may have a <data> child element.

   Multiple <notify> elements may be specified.

   For example, an AS that wants to subscribe to "active-talker" events
   but does not want to be notified more frequently than every 3 seconds
   could include the following as a child of either a <createconference>

or <modifyconference> element:

```
<subscribe>
   <notify name="active-talker">
      <data>
         <item name="interval" value="3s"/>
      </data>
   </notify>
</subscribe>
```

The MS would use the <event> element to send notifications to the AS.

5.11. Conference Configuration

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

5.11.1. <audio-mixing>

The <audio-mixing> element defines the configuration of the conference audio mix. It has no child elements and has the following attributes:

mix-type: is a string indicating the audio stream mixing policy.
   Defined values are: "nbest" (where the N best 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 default value is "nbest". The attribute
   is optional.

5.12. Media Streams

<join>, <modifyjoin> and <unjoin> require the identification and manipulations 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. It is strongly recommended that the following values are
   used for common types of media: "audio" for audio media, and
   "video" for video media. 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 direction of the media flow
   between a dialog and its end point conference or connection.
   Defined values are: "sendrecv" (media can be sent and received),
   "sendonly" (media can only be sent), and "recvonly" (media can
   only be received). The default value is "sendrecv". The
   attribute is optional.

When the "media" attribute has a value of "audio", the <stream>
element has the following child elements defined:

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

<clamp>: an element to configure filtering and removal of tones from
   a media stream. The element is optional.

5.12.1. Configuring Volume

The <volume> element is used to configure the volume of an audio
media stream. It may 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), "setstate" (set the state
   of the stream to "mute" or "unmute" as specified by the value of
   the "value" attribute). The attribute is optional.

value: a string specifying the amount or state for the volume
   control defined by the value of the "controltype" attribute.

5.12.2. Configuring Tone Removal

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

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

tones: A list of the tones to remove.

5.13. Responses

Responses are specified in a <response> element.

5.13.1. <response>

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

The <response> element has following attributes:

status: numeric code indicating the response status. The attribute
   is mandatory.

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

conf-id: string identifying the conference (see Section 16.1 of
   [I-D.boulton-sip-control-framework]). The attribute is optional.

connection-id: string identifying the SIP dialog connection (see
   Section 16.1 of [I-D.boulton-sip-control-framework]). The
   attribute is optional.

The following status codes are defined:

```
+-----------+-------------------------------------------------+
| code      | description                                     |
+-----------+-------------------------------------------------+
| 200       | OK                                              |
|           |                                                 |
| 401       | Dialog already exists                           |
|           |                                                 |
| 402       | Dialog does not exist                           |
|           |                                                 |
| 403       | Conference already exists                       |
|           |                                                 |
| 404       | Conference does not exist                       |
|           |                                                 |
| 405       | Unable to configure audio mix                   |
|           |                                                 |
| 406       | Unable to create subscription                   |
|           |                                                 |
| 407       | Conference reservation failed                   |
|           |                                                 |
| 420       | Unable to join requested entities               |
|           |                                                 |
| 421       | Unable to join - conference full                |
|           |                                                 |
| 422       | Unable to join - mixing connection inputs not   |
|           | supported                                       |
|           |                                                 |
| 450       | Unknown or unsupported element                  |
|           |                                                 |
| 451       | Element required                                |
|           |                                                 |
| 452       | Unknown or unsupported attribute                |
|           |                                                 |
| 453       | Attribute required                              |
+-----------+-------------------------------------------------+
```

                   Table 4: <response> Status codes

   [Editors Note: more codes probably need to be defined.]

   For example, a response when a conference was created successfully:

   <response code="200">
    <reason>Success</reason>
   </response>

   The response if conference creation failed due to the requested
   conference id already existing:

```
<response code="403">
 <reason>Conf already exists</reason>
</response>
```

6. Examples

[Editors Note: TODO]

7. **Formal Syntax**

   [Editors Note: XML schema to be updated.]

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

    <xs:element name="createconference" type="createType"/>
    <xs:element name="modifyconference" type="modifyConfType"/>
    <xs:element name="destroyconference" type="destroyConferenceType"/>
    <xs:element name="join" type="joinType"/>
    <xs:element name="modifyjoin" type="modifyJoinType"/>
    <xs:element name="unjoin" type="unjoinType"/>
    <xs:element name="event" type="eventType"/>
    <xs:element name="response" type="msc-conf-response"/>
    <xs:any namespace="##other" processContents="lax"
      minOccurs="0" maxOccurs="unbounded"/>

  <xs:complexType name="msc-conf-response">
   <xs:sequence>
    <xs:element name="reason" minOccurs="0">
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:pattern value="[a-zA-Z0-9.-_]+"/>
      </xs:restriction>
     </xs:simpleType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="code">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:pattern value="\d{3}"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
  </xs:complexType>

  <xs:complexType name="createType">
   <xs:sequence>
    <xs:element name="audio-mixing" type="audioMixingType"
       minOccurs="1" maxOccurs="1" default="nbest"/>
    <xs:element ref="subscribe"/>
    <xs:element name="reserved-talkers" type="xs:nonNegativeInteger"
```

```
        minOccurs="0" maxOccurs="1" default="0"/>
     <xs:element name="reserved-listeners" type="xs:nonNegativeInteger"
        minOccurs="0" maxOccurs="1" default="0"/>
     <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="conf-id" type="fw:conf-id" use="required"/>
   <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <xs:complexType name="modifyConfType">
   <xs:sequence>
     <xs:element name="audio-mixing" type="audioMixingType"
        minOccurs="1" maxOccurs="1" default="nbest"/>
     <xs:element ref="subscribe"/>
     <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
   <xs:attribute name="conf-id" type="fw:conf-id" use="required"/>
   <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <xs:complexType name="destroyConferenceType">
   <xs:sequence>
     <xs:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
   <xs:attribute name="conf-id" type="fw:conf-id" use="required"/>
   <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <xs:element name="subscribe">
   <xs:complexType>
     <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="notify"/>
     </xs:choice>
     <xs:anyAttribute namespace="##other" processContents="strict"/>
   </xs:complexType>
  </xs:element>

  <xs:element name="notify">
   <xs:complexType>
     <xs:choice minOccurs="0" maxOccurs="1">
      <xs:element ref="data"/>
     </xs:choice>
     <xs:attribute name="name" type="xs:string" use="required"/>
     <xs:anyAttribute namespace="##other" processContents="strict"/>
   </xs:complexType>
```

```
    </xs:element>

    <xs:complexType name="joinType">
     <xs:sequence>
      <xs:element ref="stream"/>
      <xs:any namespace="##other" processContents="lax"
         minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="id-1" type="xs:string" use="required"/>
     <xs:attribute name="id-2" type="xs:string" use="required"/>
     <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="modifyJoinType">
     <xs:sequence>
      <xs:element ref="stream"/>
      <xs:any namespace="##other" processContents="lax"
         minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="id-1" type="xs:string" use="required"/>
     <xs:attribute name="id-2" type="xs:string" use="required"/>
     <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="unjoinType">
      <xs:sequence>
      <xs:element ref="stream"/>
      <xs:any namespace="##other" processContents="lax"
         minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:attribute name="id-1" type="xs:string" use="required"/>
     <xs:attribute name="id-2" type="xs:string" use="required"/>
     <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>

    <xs:element name="stream">
     <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
       <xs:any namespace="##other" processContents="strict"/>
      </xs:choice>
      <xs:attribute name="media" type="media.datatype"
          use="required"/>
      <xs:attribute name="label" type="label.datatype"/>
      <xs:attribute name="direction" type="direction.datatype"
         default="sendrecv" />
      <xs:anyAttribute namespace="##other" processContents="strict"/>
     </xs:complexType>
    </xs:element>
```

```
     <xs:complexType name="eventType">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
       <xs:element ref="data"/>
       <xs:any namespace="##other" processContents="lax"
           minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
      <xs:attribute name="name" type="eventname.datatype"
          use="required"/>
      <xs:attribute name="conf-id" type="fw:conf-id" use="required"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
     </xs:complexType>

      <xs:element name="data">
      <xs:complexType>
       <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="item"/>
        <xs:any namespace="##other" processContents="strict"/>
       </xs:choice>
       <xs:anyAttribute namespace="##other" processContents="strict"/>
      </xs:complexType>
     </xs:element>

     <xs:complexType name="audioMixingType">
      <xs:attribute name="mix-type" type="mix-typeType"
          use="optional"/>
       <xs:anyAttribute namespace="##other" processContents="strict"/>
     </xs:complexType>


     <xs:element name="item">
      <xs:complexType>
       <xs:attribute name="name" type="xs:string"
          use="required"/>
       <xs:attribute name="value" type="xs:string"
          use="required"/>
       <xs:anyAttribute namespace="##other" processContents="strict"/>
      </xs:complexType>
     </xs:element>

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

     <xs:simpleType name="mix-typeType">
      <xs:restriction base="xs:NMTOKEN">
       <xs:enumeration value="nbest"/>
```

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

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

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

 <xs:simpleType name="directionType">
  <xs:restriction base="xs:NMTOKEN">
   <xs:enumeration value="both"/>
   <xs:enumeration value="transmit"/>
   <xs:enumeration value="receive"/>
  </xs:restriction>
 </xs:simpleType>

</xs:schema>
```

Figure 5: Conference Package XML Schema

8. Security Considerations

Security Considerations to be included in later versions of this document.

9. IANA Considerations

   This document registers a new SIP Control Framework Package, a new
   MIME type, and a new XML namespace.

9.1. Control Package Registration

   Control Package name: msc-conf-audio/1.0

9.2. MIME Registration

   TODO: application/msc-conf+xml

9.3. URN Sub-Namespace Registration

   TODO: urn:ietf:params:xml:ns:msc-conf

10. Change Summary

The following are the major changes between the -03 of the draft and the -02 version.

o   None

The following are the major changes between the -02 of the draft and the -01 version.

o   clarified the model for join operations and introduced several new package error codes

o   added definition for MS connection

The following are the major changes between the -01 of the draft and the -00 version.

o   restructured into single request response model for non-trival operations

o   aligned with XML structure of other control framework packages

11. Acknowledgments

The authors would like to thank Ian Evans from Ubiquity Software for help in the design of concepts contained in this document. Dave Burke has contributed valuable review comments.

12. References

12.1. Normative References

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

12.2. Informative References

[I-D.boulton-sip-control-framework]
            Boulton, C., "A Control Framework for the Session
            Initiation Protocol (SIP)",
            draft-boulton-sip-control-framework-05 (work in progress),
            February 2007.

[RFC2616]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
            Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
            Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[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.

[RFC3262]   Rosenberg, J. and H. Schulzrinne, "Reliability of
            Provisional Responses in Session Initiation Protocol
            (SIP)", RFC 3262, June 2002.

[RFC3263]   Rosenberg, J. and H. Schulzrinne, "Session Initiation
            Protocol (SIP): Locating SIP Servers", RFC 3263,
            June 2002.

[RFC3264]   Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model
            with Session Description Protocol (SDP)", RFC 3264,
            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.

[RFC3725]   Rosenberg, J., Peterson, J., Schulzrinne, H., and G.
            Camarillo, "Best Current Practices for Third Party Call
            Control (3pcc) in the Session Initiation Protocol (SIP)",
            BCP 85, RFC 3725, April 2004.

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

   [XML]   Bray, T., Paoli, J., Sperberg-McQueen, C M., Maler, E.,
              and F. Yergeau, "Extensible Markup Language (XML) 1.0
              (Third Edition)", W3C Recommendation, February 2004.

Authors' Addresses

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

    Email: cboulton@ubiquitysoftware.com

    Tim Melanchuk
    BlankSpace

    Email: tim.melanchuk@gmail.com

    Scott McGlashan
    Hewlett-Packard
    Gustav III:s boulevard 36
    SE-16985 Stockholm, Sweden

    Email: scott.mcglashan@hp.com

    Asher Shiratzky
    Radvision
    24 Raoul Wallenberg st
    Tel-Aviv, Israel

    Email: ashers@radvision.com