A Conference Control Package for the Session Initiation Protocol (SIP)
draft-boulton-conference-control-package-00

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on September 10, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

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[7].

Table of Contents

1. Introduction .......................................................... 3
2. Conventions and Terminology ........................................ 3
3. Overview .............................................................. 3
4. Control Package Definition .......................................... 3
   4.1. Control Package Name ........................................... 3
   4.2. Framework Message Usage ...................................... 4
   4.3. Common XML Support ............................................. 4
   4.4. CONTROL Message Body ........................................ 4
   4.5. REPORT Message Body .......................................... 4
5. Conference Schema Description ....................................... 4
   5.1. CONTROL Message Operations .................................. 5
       5.1.1. <create> .................................................. 5
       5.1.2. <modify> ................................................ 5
       5.1.3. <delete> ................................................ 5
       5.1.4. <event> ................................................ 5
   5.2. Configuration Objects .......................................... 5
       5.2.1. Conference Configuration Object .......................... 6
       5.2.2. Members Configuration Object ............................ 6
       5.2.3. <talker-event> ........................................... 8
6. Examples .............................................................. 8
7. Formal Syntax .......................................................... 8
   7.1. Conf Control Schema ........................................... 8
8. Security Considerations .............................................. 13
9. IANA Considerations .................................................. 13
   9.1. Control Package Registration ................................ 13
   9.2. MIME Registration ............................................ 13
   9.3. URN Sub-Namespace Registration ................................ 13
10. Acknowledgments ...................................................... 13
11. References ........................................................... 14
   11.1. Normative References ....................................... 14
   11.2. Informative References ...................................... 14
Authors’ Addresses ....................................................... 15
Intellectual Property and Copyright Statements ....................... 16
1. Introduction

The SIP Control Framework[7] provides a generic template for establishment and reporting capabilities of remotely initiated commands. The Framework utilizes many functions provided by the Session Initiation Protocol [3] (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.

A conference instance in the context of this Control Package can be defined as an identifiable object that represents zero or more associated SIP dialogs. Control commands that explicitly reference a Conference using this Control Package apply to all related SIP dialogs (unless explicitly excluded).

2. Conventions and Terminology

In this document, BCP 14/RFC 2119 [1] 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:

3. Overview

[Editors Note:TODO]

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 8 of [7].

4.1. Control Package Name

The Control Framework requires a Control Package definition to specify and register a unique name. The name of this Control Package is "msc-conf".
4.2. Framework Message Usage

The intent for the Conference Control package is for the creation, manipulation and deletion of conference instances. 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 [7], MUST only be sent from the network entity providing the application logic.

4.3. Common XML Support

This Control package MUST have full support for Control Framework Formal Syntax as defined in [7].

4.4. CONTROL Message Body

This Control Package MUST fully support the XML schema defined in Section 7.1 and described in Section 5. XML commands appearing in CONTROL message bodies will be derived from the <request> element in Section 7.1.

4.5. REPORT Message Body

This Control Package MUST fully support the XML schema defined in Section 7.1 and described in Section 5. XML commands appearing in REPORT (and 200 response) message bodies will be derived from the <response> element in Section 7.1.

5. Conference Schema Description

The XML schema in Section 7.1 is designed to allow flexibility in initiating, modifying and terminating Conferences. The schema can be split into requests and responses. Requests, which are carried in CONTROL message bodies are identified by the <request> element in the schema. Similarly, responses are carried either in REPORT message bodies or 200 responses and are defined by the <response> element in the schema. The schema defines a ‘conf-id’ attribute which is associated with a <request> element. The attribute imports a common conference identifier type from the core Control Framework[7]. Each Control operation, described in Section 5.1, MUST contain a ‘conf-id’ attribute.

The schema defines a number of operations for <request> elements (described in Section 5.1) that are used to define the type of command that is to take place and the objects that are allowed within an operation (defined in Section 5.2).
The following sub-section provides more detail on the operations.

5.1. CONTROL Message Operations

5.1.1. <create>

The <create> command is used for creating initial conference instances. This is achieved by including a 'Conference Configuration Object' (as described in Section 5.2.1) and optionally a 'Member Configuration Object' (as described in Section 5.2.2). The value contained in the <request> attribute 'conf-id' needs MUST be unique.

5.1.2. <modify>

The <modify> command is used for manipulating conference instances. This is achieved by optionally including a 'Conference Configuration Object' (as described in Section 5.2.1) and optionally a 'Member Configuration Object' (as described in Section 5.2.2). Either a 'Conference Configuration Object' or a 'Member Configuration Object' MUST exist in a modify command. The value contained in the <request> attribute 'conf-id' MUST reference an existing conference instance.

5.1.3. <delete>

The <delete> command is used for removing conference instances. No configuration objects are required or permitted to delete a conference instance. The value contained in the <request> attribute 'conf-id' needs to reference an existing conference instance.

5.1.4. <event>

The <event> command is used for reporting subsequent events related to an initial command request. Events can only be carried in a REPORT message. Only one event type is defined in the conference package which reports active speaker information for a conference instance. More information is provided later in this document.

5.2. Configuration Objects

Request operations contained in CONTROL messages (create/modify) contain configuration objects. The configuration objects can be split into two distinct areas - configuration and parameters related to the conference instance and the members of the conference instance. The two configuration objects are represented by the <conf-config> and <member-config>. The following subsections provide more information on the two Configuration objects.
5.2.1. Conference Configuration Object

The Conference Configuration Object consists of the following elements:

5.2.1.1. <audio-settings>

The <audio-settings> element then comprises the following features:

5.2.1.1.1. <active-talker>

The <active-talker> element is included if the controlling entity wishes to receive notifications of active speakers. Active speaker notifications are delivered using the <event> (Section 5.1.4) element container and the <talker-event> event type (Section 5.2.3).

attributes:

status: The <interval> attribute represents the minimum amount of time that must lapse before further talker events can be generated. The default value is "0" which suppresses notifications.

5.2.1.1.2. <audio-mixing>

<audio-mixing> element is a string indicating the audio stream mixing policy. Defined values are: "nbest" and "manual" (audio stream is selected manually by the AS via an external floor control event). The default value is "nbest". The attribute is optional.

5.2.1.1.3. <reserved-talkers>

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

5.2.1.1.4. <reserved-listeners>

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

5.2.1.2. <video-settings>

[Editors Note: TODO].

5.2.2. Members Configuration Object

The Members Configuration Object consists of the following elements:
5.2.2.1. <add-member>

This element is included when a member is to be added to a conference instance. The <add-member> element contains a <connection-id> element which supplies the reference to the SIP dialog that is to be added to the Conference Instance (This is imported from the Control Framework[7]). The <add-member> element can appear multiple times.

attributes:

status: The <add-member> element can appear multiple times in a single command and is used to add a member to a conference instance. For this reason, the success of each <add-member> operation is recorded in the status attribute. The default value is false which is then set to true and returned in the 200 response.

5.2.2.2. <modify-member>

This element is included when a member is to be modified in a conference instance. The <modify-member> element contains a <connection-id> element which supplies the reference to the SIP dialog that is to be modified in a Conference Instance (This is imported from the Control Framework[7]). The <add-member> element can appear multiple times.

attributes:

status: The <modify-member> element can appear multiple times in a single command and is used to modify a member of a conference instance. For this reason, the success of each <modify-member> operation is recorded in the status attribute. The default value is false which is then set to true and returned in the 200 response.

5.2.2.3. <remove-member>

This element is included when a member is to be removed from a conference instance. The <remove-member> element contains a <connection-id> element which supplies the reference to the SIP dialog that is to be removed from a Conference Instance (This is imported from the Control Framework[7]). The <add-member> element can appear multiple times.

attributes:

status: The <remove-member> element can appear multiple times in a single command and is used to remove a member from a conference instance. For this reason, the success of each <remove-member> operation is recorded in the status attribute. The default value is
false which is then set to true if successful and returned in the 200
response.

5.2.3. <talker-event>

The XML schema also defines the generation of responses to the
operations carried out in <request> elements. It should be noted
that responses have nothing to do with the status of the Control
Framework message transaction (CONTROL, REPORT) which has its own set
of responses defined in [7]. The result of the operation is carried
in the body of either a Control Framework 200 response to a CONTROL
message or in a REPORT message (depending on timing of operation).
The following codes are defined to appear in the <response> element:

Success
200 OK

Client Error
400 Bad Request
401 Conf already exists
402 Conf does not exist
401 Unknown or Unsupported Element
402 Element Required
403 Unknown or Unsupported Attribute
404 Attribute Required

Server Error
500 Internal Server Error
503 Service Unavailable
520 No Resource

[Editors Note: TODO - response section needs to be completed
properly.

6. Examples

7. Formal Syntax

[Editors note: XML schema goes here.]

7.1. Conf Control Schema

Text

<?xml version="1.0" encoding="UTF-8"?>
<xs:element name="msc-conf">
  <xs:complexType>
    <xs:choice>
      <xs:element name="request" type="msc-conf-request"/>
      <xs:element name="response" type="msc-conf-response"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:complexType name="msc-conf-request">
  <xs:choice>
    <xs:element name="create" type="createType"/>
    <xs:element name="modify" type="modifyType"/>
    <xs:element name="delete" type="xs:empty"/>
    <xs:element name="event" type="eventType"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:choice>
  <xs:attribute name="conf-id" type="fw:conf-id" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<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:element name="member-operations" type="memberConfigType"/>
  </xs:sequence>
  <xs:attribute name="response">
    <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="conf-config" type="confConfigType" minOccurs="1" maxOccurs="1"/>
    <xs:element name="member-config" type="memberConfigType" minOccurs="0" maxOccurs="1"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="modifyType">
  <xs:sequence>
    <xs:element name="conf-config" type="confConfigType" minOccurs="0" maxOccurs="1"/>
    <xs:element name="member-config" type="memberConfigType" minOccurs="0" maxOccurs="1"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="confConfigType">
  <xs:sequence>
    <xs:element name="audio-settings" minOccurs="0" maxOccurs="1" default="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="active-talker" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:attribute name="interval" type="xs:positiveInteger" default="0"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="audio-mixing" type="audioMixingType" minOccurs="0" maxOccurs="1" default="nbest"/>
          <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:anyAttribute namespace="##any" processContents="lax"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="video-settings" minOccurs="0" maxOccurs="1" default="0">
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="memberConfig" type="memberConfigType"/>

<xs:complexType name="memberConfigType">
  <xs:sequence>
    <xs:element name="add-member" type="addMemberType"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="modify-member" type="modifyMemberType"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="remove-member" type="removeMemberType"
      minOccurs="0" maxOccurs="1"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="addMemberType">
  <xs:sequence>
    <xs:element name="connection-id" type="connection-idType"
      minOccurs="1" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="status" type="xs:boolean"
    default="false" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="modifyMemberType">
  <xs:sequence>
    <xs:element name="connection-id" type="connection-idType"
      minOccurs="1" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="status" type="xs:boolean"
    default="false" use="required"/>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="removeMemberType">
</xs:complexType>
<xs:sequence>
  <xs:element name="connection-id" type="connection-idType"
    minOccurs="1" maxOccurs="unbounded"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="status" type="xs:boolean"
  default="false" use="required"/>
<xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="connection-idType">
  <xs:sequence>
    <xs:element name="id" type="fw:connection-id"
      minOccurs="1" maxOccurs="1"/>
    <xs:element name="direction" type="directionType"
      minOccurs="0" maxOccurs="unbounded" default="both"/>
    <xs:element name="label" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="eventType">
  <xs:choice>
    <xs:element name="talker-event" type="talkerEventType"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:choice>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="talkerEventType">
  <xs:sequence>
    <xs:element name="id" type="fw:connection-id"
      minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="audioMixingType">
  <xs:restriction base="xsd:NMTOKEN">
    <xs:enumeration value="nbest"/>
    <xs:enumeration value="manual"/>
  </xs:restriction>
</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>

Figure 1: 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

9.2. MIME Registration

TODO: application/msc-conf+xml

9.3. URN Sub-Namespace Registration


10. Acknowledgments

The authors would like to thank Ian Evans from Ubiquity Software for help in the design of concepts contained in this document.

11. References
11.1. Normative References


11.2. Informative References


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
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

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

Disclaimer of Validity

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

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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