Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2007

S. McGlashan
Hewlett-Packard
R. Auburn
Voxeo
D. Burke
Voxpilot
E. Candell
Comverse
R. Surapaneni
Tellme Networks
March 2, 2007

Media Server Control Protocol (MSCP)
draft-mcglashan-mscp-03.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies MSCP (Media Server Control Protocol), a protocol to control interactive dialog and conferencing functions on a media server. The protocol messages - requests, responses and notifications - are modeled on dialog and conferencing elements defined in CCXML (Voice Browser Call Control), and interactive dialogs can be specified in VoiceXML (Voice Extensible Markup Language). MSCP messages have self-contained XML representation and transaction models, so the protocol is independent of the underlying transport channel. Messages may be transported using SIP or, preferably, using a dedicated transport channel.

Comments

Comments are solicited and should be addressed to the authors.

Comments already received on the -00/-01 versions will be addressed in the next release of this draft and do not need to be resent.

Table of Contents

1. Conventions

In examples, "AS:" and "MS:" indicate protocol messages sent by the application server and media server respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

This document specifies a protocol for an application server (AS) to control interactive media and conferencing functions on a media server (MS). The AS sends requests for functionality to the MS, and the MS responds. Notifications may also be sent from the AS to the MS, and vice versa.

The protocol messages are directly modeled on W3C CCXML 1.0 [CCXML10] elements for interactive dialog and conferencing, but adapted for over-the-wire transport.

The protocol uses request-response and notification transaction models and the messages themselves are expressed as XML documents.

Protocol transactions are described independent of the transport channel.

[Editors Note: The next version of MSCP will be synchronized with the SIP Control Framework ([SIPCONTROLFRAMEWORK]). MSCP functionality for IVR and conferencing will be characterized in terms of Framework Control Packages. This is expected to have the following impact:

1. The TCP Transport Channel (Section 7.2) will be updated to the dedicated TCP channel in the SIP Control Framework.

2. The Interactive Media Functionality (Section 4) will be updated to the Basic IVR package ([BASIC_IVR_PACKAGE]) and VoiceXML IVR package ([VXML_IVR_PACKAGE]). Functionality relating to VoiceXML transfer will be part of a separate package (yet to be defined).

3. The Conferencing Functionality (Section 5) will be updated to the Basic Conference Package ([BASIC_CONFERENCE_PACKAGE] and an advanced Conferencing package (in development).]

2.1. Scope

The scope of the protocol is control of media server functions for interactive media (e.g. play a prompt, interpret DTMF, etc) and conferencing functions (e.g. create a conference, join participants to conference, etc) as well as notifications related to these functions. The protocol defines request-response and notification messages in XML [XML] for these functions.

The protocol also provides extensibility mechanisms allowing messages which are defined outside this document to be passed using the MSCP protocol. The extensibility mechanisms MAY be used to pass messages describing other functions; for example, functions such as:

- billing and charging
- server management
- resource availability and reservation
- media transcoding (note: this function may be implicit in media server functionality; e.g. the media server may automatically transcode media streams in conference)

While not in the scope of this protocol, these functions may be encoded in the protocol using its extensibility mechanisms. The MS is not required to support functions which are not in scope of this protocol.

If a requested function is not supported by the MS, it MUST respond with the appropriate error message.

ISSUE: A future version of this document may specify how media server capabilities are made available (e.g. whether dialog, conferencing or both are supported, as well as specific conference or dialog capabilities - e.g. VoiceXML 2.1, speech recognition support, video conferencing support, etc).

2.2. Terminology

Many of the basic concepts of the MSCP protocol are modeled on the concepts described in CCXML, including dialog, connection, and conference.

Dialog: A dialog performs media interaction with a user. A dialog is specified by a dialog language, e.g. VoiceXML 2.0 [VXML20], and is identified by a URI. Dialogs typically feature synthesized speech, digitized audio and video, recognition of spoken and DTMF key input, recording of audio and video input, and mixed initiative conversations.

Connection: A connection refers to two or more independent unidirectional media streams and its associated network signaling traffic. For the purposes of this specification, a connection consists of a SIP dialog and its associated multimedia session.

Conference: A conference is an instance of a multi-party conversation.

Application server: A SIP [RFC3261] application server (AS) hosts and executes services such as interactive media and conferencing in an operator's network. An AS influences and impacts the SIP session, in particular by terminating SIP sessions on a media server, which is under its 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. Conferencing mixing [RFC4353] functionality is controlled directly by the AS.

2.3. Architecture

Even though the MSCP protocol is modeled on a subset of CCXML elements, the protocol is agnostic to the implementation of the AS and MS. These may be implemented using CCXML and VoiceXML interpreters, due to ease of authoring applications in XML. For example, the AS is implemented as a CCXML interpreter and the MS is implemented using a VoiceXML interpreter for dialog functions and a conferencing interpreter for conferencing functions.

The MSCP protocol can be used between an AS and MS in a variety of architectures; for example, in SIP architectures including IPCC and 3GPP IMS. In the IMS architecture, an incoming call arriving at the CSCF is handled by a SIP AS. When media processing is required, the AS interacts with a MRF MS to obtain media services.

For conferencing applications, the AS includes the conference focus, conference policy server, and conference notification server [RFC4353]. The AS MAY support the SIP Event Package for Conference State [RFC4575]. The MS exposes mixer functionality under control of the AS via the MSCP protocol.

For a CCXML AS, the conference policy is defined by the executing CCXML application. A conference policy server may be implemented via the CCXML Basic Event I/O Processor. The conference URI, which uniquely identifies the focus of the conference, can be mapped by the AS to a specific CCXML conferencing application. For example, 1234@as1.example.com might be mapped to http://www.example.com/confapp01.ccxml.

2.3.1. Connections, Dialogs and Conferences

Connections, dialogs, and conferences have unique identifiers. Both dialogs and conferences are assigned a unique ID by the MS in response to a <dialogstart> or <createconference> request

respectively.

Connections, on the other hand, are assigned unique identifiers based on the dialog ID of the SIP connection to the MS [RFC3261]. The MSCP protocol represents the dialog ID as an ordered concatenation of the Call-ID value, the From tag, and the To tag, delimited by a semi-colon. For example, the connection id of a84b4c71@10.0.0.2;1923;1134 applies for the following SIP dialog:

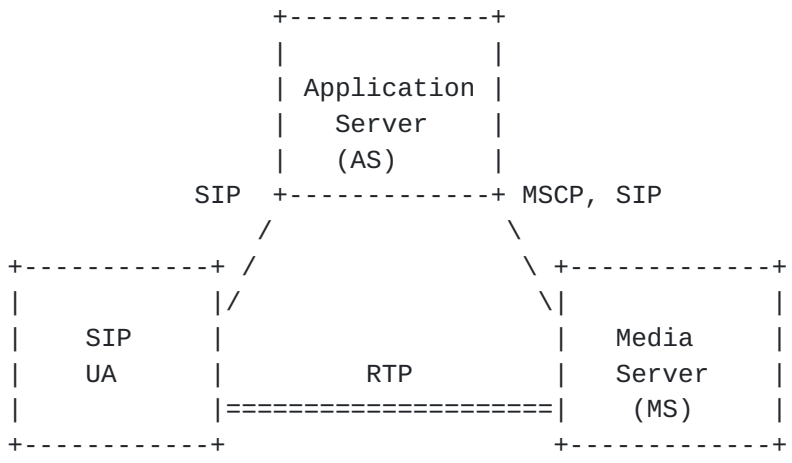
```
ACK sip:ms@example.com SIP/2.0
Via: SIP/2.0/TCP as.example.com:5060;
     branch=z9hG4bK776asdhds
Max-Forwards: 70
To: MS <ms@example.com>;tag=1134
From: AS <sip:as.example.com>;tag=1923
Call-ID: a84b4c71@10.0.0.2
CSeq: 314161 INVITE
Content-Length: 0
```

2.3.2. Interaction Model

The MSCP protocol itself does not require any particular client interaction model except for the assumption that the interactions between the client and the MS are controlled by the AS.

The remainder of this section describes a common client interaction model with the AS and MS and may be considered informative.

The MSCP control channel is established between the AS and MS (see Section 7). SIP clients typically interact with the MS via the AS. For each connection from the UA being served with interactive media or conferencing, a SIP dialog exist between the client and AS, and AS and MS.



The AS employs third party call control (3PCC) procedures [RFC3725] to direct the signaling between a SIP client and MS and to establish media streams between the two.

Example

In this example, a SIP UA sends an INVITE the AS. The AS accepts the connection (e.g. for a CCXML AS, this equates to executing <accept>). The answer contains "black hole" SDP, with its connection address equal to 0.0.0.0. Some time later, the AS then decides to start a dialog and bridge the SIP UA to it. The AS invokes the MS and client and mediates the SDP offer/answer exchange resulting in an RTP stream between the SIP UA and MS. Finally, the AS starts a VoiceXML dialog via the MSCP protocol; the connectionid indicated in the <dialogstart> refers to the dialog ID of the SIP dialog between the AS and MS.

2.4. Transport Channel

The MSCP protocol employs a self-contained XML representation for its control messages, so it is independent of the underlying transport channel. In principle, MSCP protocol messages could be transported over any mechanism which respects the constraints described in this document.

Within SIP-based architecture, SIP is employed for establishing and managing the transport channel for the MSCP protocol. While SIP itself is not an ideal transport channel for a control protocol, MSCP messages MAY be transported over SIP. However, MSCP messages SHOULD be transported over a dedicated transport channel.

The transport channel is described in more detail in Section 7.

3. Protocol Overview

The protocol uses two transaction models: a request-response model and a notification model. Requests, responses and notification messages are encoded as XML documents.

3.1. Transaction Models

In the request-response transaction model, the AS sends a request message to the MS. The MS MUST send a response message as defined in this document.

In the notification transaction model, a notification message can be sent from the AS to the MS or from the MS to the AS. No response message is required.

Note that the AS implicitly subscribes to notifications from a dialog/conference when it creates the dialog/conference or joins a connection to the dialog/conference. The AS may explicitly subscribe to additional notifications from the dialog or conference by using the <subscribe> element mechanism described in Section 4 and Section 5.

3.2. XML Format and Container Elements

All MSCP protocol messages are encoded as XML documents. The content type for MSCP XML documents is "application/mscp+xml". The character encoding MUST be set to UTF-8.

The top-level container element in protocol message is <mscp> which MUST have a version attribute and MUST specify the MSCP namespace. For this version of the protocol, the version attribute is set to the value "1.0". The namespace for the MSCP XML elements is urn:ietf:mscp.

For example,

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
   ...
</mscp>
```

The content of the <mscp> element MUST be exactly one of the following elements: <request>, <response> or <notification>.

ISSUE: A future version of this document may specify how multiple requests, responses or notifications are passed in a single <mscp> element.

Notification transactions are specified using the <notification> element. No attributes are defined for this element. The content of <notification> element is a notification element defined within the MSCP namespace, or an element defined in another namespace.

Requests in a request-response transaction are specified using a <request> element and its corresponding response as <response>. The content of a <request> is an element describing a request defined within the MSCP namespace, or an element defined in another namespace. Likewise, the content of a <response> is an element describing a response defined within the MSCP namespace, or an element defined in another namespace.

An optional id attribute (with an ID value) is defined for both elements. This attribute SHOULD be used with transport channels which do not support correlations ids (see Section 7). For example,

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
   <request id="req1">
     ...
   </request>
</mscp>


<mscp version="1.0" xmlns="urn:ietf:mscp">
   <response id="req1">
    ...
   </response>
</mscp>
```

If used, consecutive requests MUST contain monotonically increasing request id values. If a <request> specifies an id, then its <response> MUST specify an id with the same value. If the request does not specify an id value, its response MUST NOT specify an id.

3.3. XML Processing and Extensibility

XML messages MUST be checked for well-formedness and SHOULD be validated against the MSCP schema prior to execution; XML errors must be reported in the transport channel protocol (see Section 7).

The MSCP protocol can be extended using elements and attributes which MUST be defined in a namespace other than the MSCP namespace. Elements not in the MSCP namespace MAY be specified as the contents of <request>, <response> or <notification> elements. For example,

```
<mscp version="1.0" xmlns="urn:ietf:mscp"
        xmlns:vendor="http://www.example.org/vendor">
   <request>
     <vendor:myrequest>
     ...
     </vendor:myrequest>
   </request>
</mscp>
```

Attributes which are not in the MSCP namespace MAY be specified as attributes of any element in the MSCP namespace.

```
<mscp version="1.0" xmlns="urn:ietf:mscp"
         xmlns:vendor="http://www.example.org/vendor">
   <request vendor:tag="mytag">
    ...
   </request>
</mscp>
```

A conformant MSCP AS or MS MUST process XML elements within the MSCP namespace as described in this specification. Elements and attributes defined in other namespaces MAY be processed by a conformant MSCP AS or MS.

If a MSCP AS or MS encounters an XML document with elements or attributes in a non-MSCP namespace which it does not understand, it MUST ignore these elements and attributes and attempt to process the document as if it contains only elements and attributes within the MSCP namespace.

4. Interactive Media Functionality

Interactive media covers functionality such as playing a prompt, recording user input, collecting DTMF, TTS, ASR and other media-based processing. These functions are expressed in a dialog script: i.e. a script which describes the media operations and associated dialog processing. VoiceXML 2.0 [VXML20] is capable of expressing these functions. For example, a VoiceXML document is able to express simple interaction like play a prompt, or prompt and collect, as well as more advanced functionality including speech recognition, mixed initiative interaction, video playback and record, and so forth. While VoiceXML 2.0 may not be able to express all the functions described in this specification, they are either on the W3C roadmap (see http://www.w3.org/Voice/") for VoiceXML 3.0 (such as fax and video), or can be provided by vendor-specific extensions of VoiceXML.

In order to obtain interactive media functionality, the AS requests the MS to execute a dialog script. The dialog script itself describes the specific functionality. There are two key request elements:

<dialogprepare>: prepare a dialog script for later execution

<dialogstart>: execute a dialog script (as defined or previously prepared)

Once a dialog is active, the MS may send dialog notification events to the AS. These include events indicating that the dialog has ended normally or abnormally, intermediate data or that the dialog script requests disconnection or transfer.

The MS MUST support VoiceXML 2.0 dialog scripts and MAY support other dialog script formats.

4.1. <dialogprepare> Request-Response

The <dialogprepare> request is sent from the AS to the MS to request preparation of a dialog. A prepared dialog is executed when the AS sends a <dialogstart> request referencing the prepared dialog (see Section 4.2).

A <dialogprepare> element has the following attributes:

src: string identifying the URI of the dialog document to prepare. The parameter is optional. Exactly one of the src attribute or the <src> element MUST be specified; otherwise, it is an error.

type: string identifying the MIME type of the document. The default value is "application/voicexml+xml". The attribute is optional.

connectionid: string identifying the connection for which this dialog will be prepared. If the connectionid is specified, the conferenceid MUST NOT be specified. If neither the connectionid nor the conferenceid is specified, then the dialog can be started on any connection or conference (see Section 4.2). The parameter is optional.

conferenceid: string identifying the conference bridge for which this dialog will be prepared. If the conferenceid is specified, the connectionid MUST NOT be specified. If neither the connectionid nor the conferenceid is specified, then the dialog can be started on any connection or conference (see Section 4.2).The parameter is optional.

maxage: string defining a time interval according to the max-age parameter in HTTP 1.1 [RFC2616]. The attribute is optional.

maxstale: string defining a time interval according to the max-stale parameter in HTTP 1.1. The attribute is optional.

enctype: string identifying the encoding type of the submitted document. The default value is "application/ x-www-form-url-encoded". The attribute is optional.

method: string indicating the HTTP method to use. Permitted values are "post" or "get". The default value is "get". The attribute is optional.

Note that maxage, maxstale, enctype and method attributes are only relevant when the src attribute is defined with the HTTP protocol. In addition, these attributes only apply to the retrieval and caching of the initial dialog document.

The <dialogprepare> element has the following child elements:

<src>: contains the dialog script itself; e.g. a VoiceXML document. Exactly one of the src attribute or the <src> element MUST be specified; otherwise, it is an error. The element is optional.

<namelist>: contains a list of one or more <item> elements where each item element has mandatory name and value attributes. These parameters are passed into the dialog script. In VoiceXML, they are exposed via the session level object "connection.ccxml.values.*". The element is optional.

<subscribe>: contains a list of one or more <item> elements where each item element has mandatory name and value attributes. The element is optional. The AS uses this element to subscribe to events generated by the MS. Notifications of dialog events are delivered using <dialoguser> notifications (see Section 4.6). If the MS does not support a specific event notification to which the AS subscribes, then the MS MUST ignore the individual <item>. This protocol does not require the MS to support any specific event notifications, but the MS MAY support notification events such as "dtmf" (indicating that a DTMF key has been pressed), or "tone" (indicating that a tone has been detected), "audiostart" (audio playback has started), "bargein" (user has barged in), "mark" (a mark has been encountered in the output stream), "goto" (dialog has transitioned to another location), and so forth.

<stream>: contains a "media" attribute and a "direction" attribute to indicate the type and direction of media flow between dialog and its end point once this dialog is started. The defined values of the "media" attribute are: "audio", "video" and "audiovideo". Defined values of the "direction" attribute are "transmit", "receive" and "both" (transmit and receive media flow). Multiple <stream> elements may be specified so that the media type can be specified for specific directions; for example, audio only for transmission, but video only for reception. If no <stream> elements are specified, then the default is audio in both directions.

For example, a request to prepare a dialog where the dialog script is indicated using the src attribute:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
<dialogprepare src="http://www.example.com/playprompt.vxml">
    <namelist>
     <item name="audio" value="/media/prompt1.wav"/>
    </namelist>
   </dialogprepare>
 </request>
</mscp>
```

Where the namelist parameter "audio" would be available in the VoiceXML script as "connection.ccxml.values.audio" so different prompts can be played using the same dialog script.

In the following example, the VoiceXML dialog script is specified inline.

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <dialogprepare>
    <src>
    <vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
      <form id='main'>
         <block>
            <audio expr="http://www.example.com/media/prompt1.wav"/>
            <exit/>
          </block>
       </form>
     </vxml>
  </src>
 </dialogprepare>
</request>
</mscp>
```

When an MS has received a <dialogprepare> request, it MUST reply with a <dialogprepared>, <errordialognotprepared> or <errordialogwrongstate> response element. These elements have following attributes:

dialogid: string identifying the dialog. The MS assigns a globally unique identifier for this dialog and reuses it in subsequent references to the dialog; for example, as the prepareddialogid in <dialogstart> and in dialog notifications. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. If connectionid was specified in the request, this attribute MUST have the same value. The parameter is optional.

conferenceid: string identifying the conference bridge for this dialog. If conferenceid was specified in the request, this attribute MUST have the same value. The parameter is optional.

The response elements <errordialognotprepared> and <errordialogwrongstate> have the following additional attribute defined:

reason: string specifying the reason why dialog preparation failed. The attribute is mandatory.

For example, a response when the dialog was prepared successfully:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogprepared dialogid="vxi1"/>
 </response>
</mscp>
```

And a response if dialog preparation failed:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <errordialognotprepared dialogid="vxi1"
    reason="HTTP 404 error: http://www.example.com/playprompt.vxml"/>
 </response>
</mscp>
```

4.2. <dialogstart> Request-Response

The <dialogstart> element is sent by the AS to request execution of a dialog. The dialog may be defined in the dialogstart request itself, or reference a previously prepared dialog.

The <dialogstart> element has the following attributes:

src: string identifying the URI of the dialog document to start. The parameter is optional. If the prepareddialogid is specified, the atribute MUST NOT be specified.

type: string identifying the MIME type of the document. The default value is "application/voicexml+xml". The attribute is optional. If the prepareddialogid is specified, the attribute MUST NOT be specified.

prepareddialogid: string identifying a dialog previously prepared using a dialogprepare request. The parameter is optional.

connectionid: string identifying the connection for this dialog. The parameter is optional.

If the prepareddialogid is defined, and the dialogprepare request specified a connnectionid, then if this connectionid is specified, it MUST have the same value; if the connectionid is not specified in this request, but was specified in the dialogprepare request, then the connectionid from the dialogprepare request will be used.

If the connectionid is specified, the conferenceid MUST NOT be specified. It is an error if neither the connectionid nor the conferenceid is specified either directly or through inheritance from the dialogprepare request.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional.

If the prepareddialogid is defined, and the dialogprepare request specified a conferenceid, then if this conferenceid is specified, it MUST have the same value; if the conferenceid is not specified in this request, but was specified in the dialogprepare request, then the conferenceid from the dialogprepare request will be used.

If the conferenceid is specified, the connnectionid MUST NOT be specified. It is an error if neither the connectionid nor the conferenceid is specified either directly or through inheritance from the dialogprepare request.

maxage: string defining a time interval according to the max-age parameter in HTTP 1.1. The attribute is optional. If the prepareddialogid is specified, the attribute MUST NOT be specified.

maxstale: string defining a time interval according to the max-stale parameter in HTTP 1.1. The attribute is optional. If the prepareddialogid is specified, the attribute MUST NOT be specified.

enctype: string identifying the media encoding type of the submitted document. The default value is "application/ x-www-form-url-encoded". The attribute is optional. If the prepareddialogid is specified, the attribute MUST NOT be specified.

method: string indicating the HTTP method to use. Permitted values are "post" or "get". The default value is "get". The attribute is optional. If the prepareddialogid is specified, the attribute MUST NOT be specified.

Note that maxage, maxstale, enctype and method attributes only relevant when the src attribute is defined with the HTTP protocol. In addition, they only apply to the retrieval and caching of the initial dialog document.

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

<src>: contains the dialog script itself; e.g. a VoiceXML document. The element is optional. It is an error to specify both a src attribute and <src> element. If the prepareddialogid is specified, this element MUST NOT be specified.

<namelist>: contains a list of one or more <item> elements where each item element has name and value attributes. These parameters are passed into the dialog script. In VoiceXML, they are exposed via the session level object "connection.ccxml.values.*". The element is optional. If the prepareddialogid is specified, this element MUST NOT be specified.

<subscribe>: contains a list of one or more <item> elements where each item element has mandatory name and value attributes. The element is optional.

The AS uses this element to subscribe to events generated by the MS. Notifications of dialog events are delivered using <dialoguser> notifications (see Section 4.6). If the MS does not support a specific event notification to which the AS subscribes, then the MS MUST ignore the individual <item>. This protocol does not require the MS to support any specific event notifications, but the MS MAY support notification events such as "dtmf" (indicating that a DTMF key has been pressed), or "tone" (indicating that a tone has been detected), "audiostart" (audio playback has started), "bargein" (user has barged in), "mark" (a mark has been encountered in the output stream), "goto" (dialog has transitioned to another location), and so forth.

If the prepareddialogid is specified, this element MUST NOT be specified.

<stream>: contains a "media" attribute and a "direction" attribute to indicate the type and direction of media flow between dialog and its end point once this dialog is started. The defined values of the "media" attribute are: "audio", "video" and "audiovideo". Defined values of the "direction" attribute are "transmit", "receive" and "both" (transmit and receive media flow). Multiple <stream> elements may be specified so that the media type can be specified for specific directions; for example, audio only for transmission, but video only for reception. If no <stream> elements are specified, then the default is audio in both directions.

If the prepareddialogid is specified, this element MUST NOT be specified.

For example, a request to start a dialog where the dialog script is indicated using the src attribute:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
   <dialogstart connectionid="connection1"
              src="http://www.example.com/playprompt.vxml">
     <namelist>
      <item name="media" value="/media/prompt1.3gp"/>
     </namelist>
   </dialogstart>
 </request>
</mscp>
```

Where the namelist parameter "media" would be available in the VoiceXML script as "connection.ccxml.values.media" so different prompts can be played using the same dialog script.

In the following example, the VoiceXML dialog script is specified inline and the media is audiovideo in both directions.

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <dialogstart connectionid="connection1">
    <stream type="audiovideo" direction="both"/>
    <src>
     <vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
      <form id='main'>
         <block>
            <audio expr="http://www.example.com/media/prompt1.3gp"/>
            <exit/>
          </block>
       </form>
     </vxml>
    </src>
  </dialogstart>
 </request>
</mscp>
```

In this example, a previously prepared dialog with the dialogid "vxi1" is started.

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
   <dialogstart prepareddialogid="vxi1" connectionid="connection1"/>
 </request>
</mscp>
```

When an MS has received a <dialogstart> request, it MUST reply with a <dialogstarted>, <errordialognotstarted> or <errordialogwrongstate> response element. These elements have the following attributes:

dialogid: string identifying the dialog. If prepareddialogid is specified in the request, then dialogid MUST have the same value. If prepareddialogid is not specified, then the MS assigns a globally unique identifier for this dialog and reuses it in subsequent references to the dialog; for example, in dialog notifications. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Exactly one of the connectionid or conferenceid MUST be specified.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Exactly one of the connectionid or conferenceid MUST be specified.

The response elements <errordialognotstarted> and <errordialogwrongstate> have the following additional attribute defined.

reason: string specifying the reason why dialog execution failed. The attribute is mandatory.

For example, a response when the dialog was started successfully.

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <dialogstarted dialogid="vxi1" connectionid="connection1"/>
 </response>
</mscp>
```

Response if dialog execution failed:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <error.dialog.notstarted dialogid="vxi1" connectionid="connection1"
    reason="Unhandled VoiceXML error: error.semantic: variable
    xyz not defined"/>
 </response>
</mscp>
```

4.3. <dialogterminate> Notification

A dialog that has been prepared or has been started can be terminated by a <dialogterminate> notification element from the AS.

The <dialogterminate> element has the following attributes:

dialogid: string identifying the dialog. The attribute is mandatory.

immediate: string with the values "true" or "false". The default is "false". The parameter is optional.

For example, assuming a dialog with the dialogid "vxi1" has been started, it can be terminated immediately with the following request:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <dialogterminate dialogid="vxi1" immediate="true"/>
 </notification>
</mscp>
```

The <dialogterminate> notification causes execution of the dialog to be terminated.

If the notification is for immediate termination, then the MS MUST NOT send any further notification events for this dialog.

If the request is for non-immediate termination, then the MS MAY send a <dialogexit> notification event for this dialog as described below.

4.4. <dialogexit> Notification

When an active dialog terminates normally (or after a non-immediate dialogterminate request), the MS MUST send a <dialogexit> notification.

The <dialogexit> element has the following attributes:

dialogid: string identifying the dialog. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

The <dialogexit> element has the following child element:

<namelist>: contains a list of one or more <item> elements where each item element has name and value attributes. The element is optional.

For example, the dialog exits without data being returned:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
  <dialogexit dialogid="vxi1" connectionid="connection1"/>
 </notification>
</mscp>
```

The dialog exits with data being returned:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
  <dialogexit dialogid="vxi1" connectionid="connection1">
   <namelist>
    <item name="callerid" value="12345"/>
    <item name="popidolvote" value="Franz Ferdinand"/>
   </namelist>
  </dialogexit>
 </notification>
</mscp>
```

4.5. <errordialog> Notification

When an active dialog terminates abnormally, the MS MUST send an <errordialog> notification.

The <errordialog> element has the following attributes:

dialogid: string identifying the dialog. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attribute MAY be specified; it is an error to specify both.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attribute MAY be specified; it is an error to specify both.

reason: string specifying the reason why dialog execution failed. The attribute is mandatory.

For example, the dialog execution fails:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
  <errordialog connectionid="connection1" dialogid="vxi1"
                  reason="hardware error"/>
 </notification>
</mscp>
```

4.6. <dialoguser> Notification

During execution of a dialog, <dialoguser> notifications can be sent from the MS to the AS or from the AS to the MS.

The <dialoguser> element has the following attributes:

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

dialogid: string identifying the dialog. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attribute MAY be specified; it is an error to specify both.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attribute MAY be specified; it is an error to specify both.

A <dialoguser> element has the following child element:

<namelist>: contains a list of one or more <item> elements where each item element has name and value attributes. The element is optional.

For example, the MS sends the AS a midcall update on data collected so far:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <dialoguser name="myapp.update" dialogid="vxi1">
    <namelist>
     <item name="city" value="San Francisco"/>
     <item name="state" value="California"/>
    </namelist>
   </dialoguser>
</notification>
</mscp>
```

The AS sends the MS information which may be announced to the user:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <dialoguser name="alert.priority1" dialogid="vxi1">
    <namelist>
   <item name="message" value="John Donne sent you an IM."/>
       </namelist>
   </dialoguser>
</notification>
</mscp>
```

4.7. <dialogdisconnect> Notification

An MS can request disconnection of a call by sending a <dialogdisconnect> notification. The AS subsequently sends a <dialogterminate> notification to the MS (for a VoiceXML dialog, this will result in the connection.disconnect.hangup being thrown and the final processing state being entered).

The <dialogdisconnect> element has the following attributes:

dialogid: string identifying the dialog. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

The <dialogdisconnect> element has the following child element:

<namelist>: contains a list of one or more <item> elements where each item element has name and value attributes. The element is optional.

For example, the MS sends a dialogdisconnect notification to the AS:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <dialogdisconnect dialogid="vxi1"/>
 </notification>
</mscp>
```

and the AS responds by sending a <dialogterminate> notification:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
  <dialogterminate dialogid="vxi1" immediate="false"/>
 </notification>
</mscp>
```

4.8. <dialogtransfer> Notification

An MS can request transfer of the call by sending a <dialogtransfer> notification to the AS.

The <dialogtransfer> element has the following attributes:

dialogid: string identifying the dialog. The attribute is mandatory.

type: a string specifying the transfer type (see below). The parameter is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

uri: a valid uri identifying the destination to which the call is to be transferred. The attribute is mandatory.

maxtime: string in CSS2 time format indicating the maximum amount of time the transfer can be stayed connected. The value "0s" indicates unlimited connection time. The attribute is mandatory.

connecttimeout: string in CSS2 time format indicating the maximum amount of time to elapse attempting to connect the call. The attribute is mandatory.

aai: string of application-to-application information to be passed to the destination when establishing the transfer. The attribute is optional.

The <dialogtransfer> element has the following child element:

<namelist>: contains a list of one or more <item> elements where each item element has name and value attributes. The element is optional.

Two transfer types, blind and bridged, MUST be supported; other types MAY be supported.

For a blind transfer, the AS typically redirects the SIP dialog between the UA and AS by issuing a SIP REFER. The dialog running on the MS is terminated.

For a bridge transfer, the AS creates a new SIP dialog to the callee and a new SIP dialog between the AS and MS on behalf of the callee. The original caller's and the callee's media streams are bridged by the MS. During the bridge transfer, the dialog may listen on the original caller's leg for commands to terminate the transfer. If the callee terminates the call or the dialog terminates the call (e.g. if the maxtime elapses or a command to terminate the transfer is recognized), the dialog is resumed with the original caller.

4.8.1. Blind transfer

The MS sends a notification to the AS requesting a blind transfer:

```
MS -> AS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <dialogtransfer dialogid="vxi1" type="blind"
                     uri="sip:callee@example.com" maxtime="0s"
                     connecttimeout="20s"/>
 </notification>
</mscp>
```

The AS issues a REFER to the UA and sends a notification to the MS to

terminate the dialog (for a VoiceXML dialog, this will result in the event connection.disconnect.transfer being raised):

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
      <dialogterminate dialogid="vxi1"/>
 </notification>
</mscp>
```

4.8.2. Bridge transfer

The MS sends a notification to the AS for a bridge transfer:

```
MS -> AS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
      <dialogtransfer dialogid="vxi1" type="bridge"
                      uri="sip:callee@example.com" maxtime="0s"
                      connecttimeout="20s"/>
 </notification>
</mscp>
```

The AS creates a new SIP dialog with the callee and creates a SIP dialog between the AS and MS on behalf of the callee (identified below by "conn2"). In the event of an error in creating a SIP dialog with the callee, a <dialogterminatecomplete> notification is sent to the MS, otherwise the AS instructs the MS to bridge the two calls via <join>:

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <join id1="conn1" id2="conn2"/>
 </request>
</mscp>

MS -> AS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <conferencejoined id1="conn1" id2="conn2"/>
 </response>
</mscp>
```

As a result, the original caller is bridged full duplex with the callee, and the dialog is "listening" to the caller. The transfer can terminate for several reasons:

1. the caller terminates the call

2. the AS terminates the transfer

3. the callee terminates the transfer

4. the dialog terminates the transfer

For scenario 1, the AS terminates the dialog with <dialogterminate>. The SIP dialogs for the caller and callee between the AS and MS are subsequently terminated.

For scenario 2, the AS terminates the transfer (e.g. because maxtime has elapsed) by sending a <dialogtransfercomplete> notification:

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <dialogtransfercomplete dialogid="vxi1"
                           reason="maxtime_disconnect"/>
</notification>
</mscp>
```

For scenario 3, the AS terminates the SIP dialog for the callee between the AS and MS (for a VoiceXML dialog this will result in the transfer ending with a far_end_disconnect). The AS then rejoins the dialog full duplex with the caller via <join>:

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <join id1="conn1" id2="vxi1"/>
 </request>
</mscp>

MS -> AS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <conferencejoined id1="conn1" id2="vxi1"/>
 </response>
</mscp>
```

For scenario 4, the MS sends the <dialogterminatetransfer> notification to the AS (e.g. because a command to terminate the transfer is recognized). The AS rejoins the dialog full duplex with the caller via <join>:

```
MS -> AS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <dialogterminatetransfer dialogid="vxi1"
                            reason="near_end_disconnect"/>
</notification>
</mscp>

AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <join id1="conn1" id2="vxi1"/>
 </request>
</mscp>

MS -> AS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <conferencejoined id1="conn1" id2="vxi1"/>
 </response>
</mscp>
```

4.9. <dialogterminatetransfer> Notification

An MS can request termination of bridge transfer by sending a <dialogterminatetransfer> notification to the AS.

The <dialogterminatetransfer> element has the following attributes:

dialogid: string identifying the dialog. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

reason: a string indicating the reason transfer is to be terminated. Allowed values are near_end_disconnect. The attribute is mandatory.

4.10. <dialogtransfercomplete> Notification

An AS sends a <dialogtransfercomplete> notification to the MS when a transfer completes due to an error condition or maxtime expiry.

The element has the following attributes:

dialogid: string identifying the dialog. The attribute is mandatory.

connectionid: string identifying the connection for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

conferenceid: string identifying the conference bridge for this dialog. The parameter is optional. Only one of the connectionid or conferenceid attributes MAY be specified; it is an error to specify both.

reason: a string indicating the reason the transfer completed. Allowed values are maxtime_disconnect (if the maxtime elapsed), busy (if the callee returned a 486 response to the SIP INVITE), network_busy (if the callee returned a 6xx response to the SIP INVITE), or noanswer (if connecttimeout elapsed before a final response was received from the callee to the SIP INVITE). The attribute is mandatory.

5. Conferencing Functionality

MSCP supports many conferencing models including a distributed, centralized controller model where signaling and media are directed to a central location, the conference focus, controlled, or implemented, by the AS (as defined by the IETF XCON Working Group).

ISSUE: A future version of this document will further clarify the conference models for MSCP.

In order obtain conferencing functionality, the AS makes a request to the MS. There are 4 key requests:

<createconference>: create a conference

<destroyconference>: destroys a previously created conference

<join>: add a participant to a conference

<unjoin>: remove a participant from a conference

Note that the join and unjoin requests can also be used to (un)join connections and dialogs.

Once a conference is active, the MS may send conference notification events to the AS; for example, active talkers. The AS may also send notification events to the MS; for example, explicit floor control notifications.

5.1. <createconference> Request-Response

An AS can request an MS to create a conference by sending a <createconference> request.

The <createconference> element has the following attributes:

reservedtalkers: number of guaranteed speaker slots to be reserved for the conference. The attribute is optional.

reservedlisteners: number of guaranteed listeners slots to be reserved for the conference. The attribute is optional.

reservedmedia: string specifying the type of media resources which are to be reserved. Defined values are: "audio", "video", "audiovideo". The default value is "audio". The attribute is optional.

audiomixingpolicy: a string indicating the audio stream mixing policy. Defined values are: "nbest" (see audiomixingnbest below) 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.

If the conference media is "video" or "audiovideo" and the audiomixingpolicy is specified as "manual", then the videomixingpolicy MUST also be set to "manual", otherwise it is an error.

audiomixingnbest: non-negative integer specifying the maximum number of participants generating output to be included in the audio mix based on greatest energy. A value of 0 indicates that all participants generating output are to be included in the audio mix. The default value is 0. The attribute is optional.

videomixingpolicy: a string indicating the video stream mixing policy. Defined values are: "vas" (video stream of loudest active speaker is selected), "manual" (video stream is selected manually by the AS via an external floor control event) or "userdefined" (see ISSUES note below). The attribute is optional.

If the conference media is "video" or "audiovideo", then a value for audiomixingpolicy MUST also be specified, otherwise it is an error.

If the conference media is "video" or "audiovideo" and the videomixingpolicy is specified as "manual", then the audiomixingpolicy MUST also be set to "manual", otherwise it is an error.

videomixingvas: a positive integer specifying the minimum amount time in seconds which MUST elapse before a change in video stream selection. The default value is 3. The attribute is optional.

The <createconference> element has the following child element:

<subscribe>: contains a list of one or more <item> elements where each item element has mandatory name and value attributes. The element is optional.

The AS uses this element to subscribe to events generated by the MS. Notifications of conference events are delivered using <conferenceuser> notifications (see Section 5.5). If the MS does not support a specific event notification to which the AS subscribes, then the MS MUST ignore the individual <item>.

This protocol does not require the MS to support any specific event notifications, but it does define an active speaker pattern to notify the AS of active speakers. If an MS supports active speaker notifications, then it MUST use this pattern. The AS subscribes using an <item> where the name is "activespeaker" and the value is a positive integer specifying the minimum amount time in seconds which MUST elapse before a change in active speaker is notified to the AS. A value of '0' indicates that active speaker notification is disabled. The MS notifies the AS of active speakers using a <conferenceuser> notification where the event is "activespeaker", and its namelist includes one or more <item>s with the name set to "speaker" and the value set to the connectionid of the active speaker's connection.

ISSUES:

1. A future version of this specification may address how conference resourcing, mixing policies, etc can be changed once a conference has been created. One potential solution is for the AS to send conference updates as notifications to the MS; e.g. a notification to change the audio mixing policy. Another solution is to introduce an explicit <modifyconference> request element.

2. More investigation is required for "userdefined" video mixing. For example, making use of (a subset of) SMIL to specify video layout, regions, titles, switching within regions, etc.

3. Further investigation is also required on the addition of an automatic conference termination parameter with values: nomedia (last participant gone), nocontrol (SIP session on which conference is created gone).

For example, to create a new audio conference:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <createconference reservedtalkers="5" reservedlisteners="3"/>
 </request>
</mscp>
```

When an MS has received a <createconference> request, it MUST respond with a <conferencecreated> or <errorconferencecreate> response element. These elements have the following attribute:

conferenceid: string specifying the id of the conference. The MS assigns a globally unique identifier for this conference and reuses it in subsequent references to the conference; for example, in conference notifications. The attribute is mandatory.

In addition, the <errorconferencecreate> element has the following attribute defined.

reason: string specifying the reason why conference was not created. The attribute is mandatory.

For example, the conference has been created successfully.

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferencecreated conferenceid="conf1"/>
 </response>
</mscp>
```

An error occurred preventing conference creation.

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <errorconferencecreate conferenceid="conf1"
       reason="no conference mixer available"/>
 </response>
</mscp>
```

5.2. <destroyconference> Request-Response

A conference is terminated by the AS sending a <destroyconference> request.

The <destroyconference> element has the following attribute:

conferenceid: string indicating the conference id. The attribute is mandatory.

For example:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
   <destroyconference conferenceid="conf1"/>
 </request>
</mscp>
```

When an MS has received a <destroyconference> request, it MUST reply with a <conferencedestroyed> or <errorconferencedestroy> response element. These elements have the following attribute:

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

The <errorconferencedestroy> also has the following attribute:

reason: string specifying the reason why conference was not destroyed. The attribute is mandatory.

For example, the conference "conf1" has been successfully destroyed:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferencedestroyed conferenceid="conf1"/>
 </response>
</mscp>
```

The conference "conf2" has not been destoyed:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <errorconferencedestroy conferenceid="conf2"
      reason="conference identified by conf2 does not exist"/>
 </response>
</mscp>
```

5.3. <join> Request-Response

The AS can join a participant to a conference using the join request. The participant can be a dialog, a connection or another conference.

This request can also be used to modify how a previously connected participant is joined to the conference.

The <join> element has the following attributes:

id1: string indicating a connection, dialog or conference. The parameter is mandatory.

id2: string indicating the connection, dialog or conference to join to id1. The parameter is mandatory.

mediapreferred: string indicating whether the participant is always allowed to contribute to the audio mix event if they are not eligible in terms of loudness. Defined values are "true" (always mixed) and "false" (must be eligible in terms of loudness). The default value is "false". The attribute is optional. The attribute is ignored if the conference audio mixing policy is not "nbest".

entertone: string with the values 'true','false' or URI (custom audio) indicating whether a tone is to be played when another participant joins the conference. The default value is 'true'. The parameter is optional.

exittone: string with the values 'true','false' or URI (custom audio) indicating whether a tone is to be played when another participant leaves the conference. The default value is 'true'. The parameter is optional.

autoinputgain: string with values 'true' or 'false' indicating whether AGC is to be used on the participant's input to the conference. The default is 'true'. The parameter is optional. The parameter is ignored if AGC is not supported.

autooutputgain: string with values 'true' or 'false' indicating whether AGC is to be used on the participant's output from the conference. The default is 'true'. The parameter is optional. The parameter is ignored if AGC is not supported.

dtmfclamp: string with values 'true' or 'false' indicating whether DTMF is to be removed from the participant's input to the conference. The default is 'true'. The parameter is optional. The parameter is ignored if this behavior is not supported.

toneclamp: string with values 'true' or 'false' indicating whether loud single frequency tones from are to be removed from the participant's input to the conference. The default is 'true'. The parameter is optional. The parameter is ignored if this behavior is not supported.

The <join> element has the following child element:

<stream>: contains a "media" attribute and a "direction" attribute to indicate the type and direction of media flow between id1 and id2. The defined values of the "media" attribute are: "audio", "video" and "audiovideo". Defined values of the "direction" attribute are "transmit" (media flow from id2 to id1 only), "receive" (media flow from id1 to id2 only) and "both" (media flow in both directions). Multiple <stream> elements may be specified so that the media type can be specified for different directions; for example, audio only for transmission, but video only for reception. If no <stream> elements are specified, then the default is audio in both directions.

For example, to join a participant's connection "connection1" with full duplex audio to the conference "conf1":

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <join id1="conf1" id2="connection1"/>
 </request>
</mscp>
```

The semantics of <join> follow the bridging model in CCXML 1.0, Section 10.4. In particular <join> follows three invariant topology rules:

1. The media stream relationship specified is established between two connections/conferences/dialogs referenced by id1 and id2 attributes in the <join> element. Any existing stream relationship between these connections/conferences/dialogs is torn down automatically if it conflicts with the specified relationship.

2. If the relationship specified in the <join> element requires a connection/dialog to listen and the connection is listening to a different source, this existing stream relationship is torn down automatically.

3. Any existing stream relationship that does not present a conflict according to invariant #1 or #2 is preserved.

When an MS has received a join request, it MUST reply with a <conferencejoined> or <errorconferencejoin> response elements. These elements have following attributes:

id1: string indicating a connection, dialog or conference. This MUST have the same value as id1 in the <join> request. The parameter is mandatory.

id2: string indicating a connection, dialog or conference. This MUST have the same value as id2 in the <join> request. The parameter is mandatory.

The <errorconferencejoin> element also has the following attribute:

reason: string specifying the reason why the join did not occur. The attribute is mandatory.

For example, the connection "connection1" is successfully joined to conference "conf1":

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <conferencejoined id1="conf1" id2="connection1"/>
 </response>
</mscp>
```

or if this join was unsuccessful:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <errorconferencejoin id1="conf1" id2="connection1"
     reason="no such connection"/>
 </response>
</mscp>
```

5.4. <unjoin> Request-Response

A participant can be removed from a conference by sending an <unjoin> request. The participant can be a dialog, connection or another conference.

The <unjoin> element has the following attributes:

id1: string indicating a connection, dialog or conference. The parameter is mandatory.

id2: string indicating the connection, dialog or conference to unjoin from id1. The parameter is mandatory.

For example, to unjoin a connection "connection1" from the conference "conf1":

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <unjoin id1="conf1" id2="connection1"/>
 </request>
</mscp>
```

When an MS has received an <unjoin> request, it MUST reply a <conferenceunjoined> or <errorconferenceunjoin> response elements. These elements have the following attributes:

id1: string indicating a connection, dialog or conference. This MUST have the same value as id1 in the <unjoin> request. The parameter is mandatory.

id2: string indicating a connection, dialog or conference. This MUST have the same value as id2 in the <unjoin> request. The parameter is mandatory.

The <errorconferenceunjoin> element also has the following attribute:

reason: string specifying the reason why the unjoin did not occur. The attribute is mandatory.

For example, connection "connection1" has successfully been unjoined from the conference "conf1":

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferenceunjoined id1="conf1" id2="connection1"/>
 </response>
</mscp>
```

or the unjoin was unsuccessful:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <errorconferenceunjoin id1="conf1" id2="connection1"
       reason="internal error"/>
 </response>
</mscp>
```

5.5. <conferenceuser> Notification

During a conference, <conferenceuser> notifications can be sent from the MS to the AS or from the AS to the MS. No response is required.

Notifications from the MS to the AS may include active talker notifications and resource assignment notifications.

Notifications from the AS to the MS may include floor control.

The <conferenceuser> element has the following attributes:

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

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

A <conferenceuser> element has the following child element defined:

<namelist>: contains a list of one or more <item> elements where each item element has name and value attributes. The element is optional.

For example, the MS sends the AS a midconference update on conference attendance data so far:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <conferenceuser name="attendance.update" conferenceid="conf1">
    <namelist>
     <item name="total" value="45"/>
    </namelist>
   </conferenceuser>
 </notification>
</mscp>
```

The AS sends the MS floor control information:

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
   <conferenceuser name="floorcontrol.setactive" conferenceid="vxi1">
    <namelist>
   <item name="connectionid" value="connection1"/>
       </namelist>
   </conferenceuser>
</notification>
</mscp>
```

6. Examples

The following examples show how the protocol is used to create interactive media and conferencing sessions.

For the sake of brevity, examples assume that the AS uses SIP 3PCC to setup media connection between incoming call from UE and MS, and that the ids of connections are based on the SIP dialog ID (see Section 2.3.1).

6.1. Interactive Media Dialogs

This example shows how the AS requests the MS to play a simple audio prompt to the user.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
   <dialogstart type="audio/wav"
     src="http://www.example.com/welcome.wav"
     connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
 </request>
</mscp>


AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogstarted dialogid="dialog1"
           connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
 </response>
</mscp>

Comment: the audio dialog has started

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogexit dialogid="dialog1"
           connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
 </response>
</mscp>

Comment: the dialog has completed.
```

The second example shows how the AS requests the MS to prepare and play an inline VoiceXML dialog which collects 4 digits from the user.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <dialogprepare>
   <src>
    <vxml version="2.0" xml:lang="en"
           xmlns="http://www.w3.org/2001/vxml">
     <form>
      <field name="digits" type="digits?length=4">
       <prompt>Please type in your pin number.</prompt>
       <filled>
        <exit namelist="digits"/>
       </filled>
      </field>
     </form>
    </vxml>
  </src>
</dialogprepare>
 </request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogprepared dialogid="dialog2"/>
 </response>
</mscp>

Comment: the dialog has been prepared for any connection but not
started.

AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
<dialogstart prepareddialogid="dialog2"
             connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
 </request>
</mscp>

Comment: the prepared dialog is started on the connection
identified as a1e8ad50029@15.124.40.73.

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogstarted dialogid="dialog2"
           connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
 </response>
```

```
</mscp>
```

Comment: the prompt and collect digits dialog has started.

```
AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogexit dialogid="dialog2"
           connectionid="a1e8ad50029@15.124.40.73;2134;1423">
      <namelist>
       <item name="digits" value="1234"/>
      </namelist>
    </dialogexit>
 </response>
</mscp>
```

Comment: the dialog has completed and returns the digit string ('1234') which the user entered.

In this final example, a VoiceXML dialog is started which plays a video prompt, records a video from the user and uploads to a server.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <dialogstart connectionid="a1e8ad50029@15.124.40.73;2134;1423">
   <stream type="audiovideo" direction="both"/>
    <src>
      <vxml version="2.0" xml:lang="en"
            xmlns="http://www.w3.org/2001/vxml">
       <form>
        <record name="msg" type="video/3gpp">
         <audio src="http://www.example.com/recordprompt.3gpp"/>
         <filled>
          <submit namelist="msg" enctype="multipart/form-data"
            method="post"
            next="http://www.example.com/recordingStore"/>
         </filled>
        </record>
      </form>
    </vxml>
   </src>
  </dialogstart>
 </request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogstarted dialogid="dialog3"
           connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
 </response>
</mscp>

Comment: audiovideo dialog started successfully.

AS <- MS

<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogexit dialogid="dialog3"
           connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
 </response>
</mscp>

Comment: dialog complete and recording uploaded to record store.
```

6.2. Basic Audio Conference

This example create a conferences, participant dials in, prompts to record their name, joins them to conference and announces their name.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <createconference/>
 </request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
  <conferencecreated conferenceid="conf1"/>
 </response>
</mscp>
```

Comment: AS requests conference to be created. The MS replies that this request has been successful and returns the conference id "conf1".

[AS receives INVITE from UE]

```
AS -> MS
SIP: INVITE (UE SDP)

AS <- MS
SIP: 200 (MS SDP)

AS -> MS
ACK
```

[AS sends 200 to UE; receives ACK]

Comment: AS uses SIP 3PCC to setup media connection between incoming call from UE and MS. The dialog ID of the UE SIP call is "a1e8ad50029@15.124.40.73;2134;1423".

```
AS -> MS

<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
  <dialogstart connectionid="a1e8ad50029@15.124.40.73;2134;1423"
     src="http://www.example.com/scripts/promptAndRecord.vxml"/>
 </request>
</mscp>
```

```
   Comment: The AS requests that a dialog is played on the connection
   identified by the SIP dialog ID. The dialog prompts the user for the
   conference code and password, and then records their name.

   AS <- MS

   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <response>
       <dialogstarted dialogid="dialog1"
              connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
    </response>
   </mscp>

   Comment: MS responses that the dialog started successfully.

   AS <- MS

   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <notification>
      <dialogexit dialog="dialog1"
          connectionid="a1e8ad50029@15.124.40.73;2134;1423">
        <namelist>
         <item name="username"
            value=http://www.example.com/recordings/asswrw456grfg.wav"/>
        </namelist>
      </dialogexit>
      </notification>
   </mscp>

   Comment: The MS notifies the AS that dialog has completed and also
   provides the uri of the username recording.

   AS -> MS

   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <request>
        <join id1="conf1" id2="a1e8ad50029@15.124.40.73;2134;1423"/>
    </request>
   </mscp>

   AS <- MS

   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <response>
      <conferencejoined id1="conf1"
             id2="a1e8ad50029@15.124.40.73;2134;1423"/>
    </response>
   </mscp>
```

```
   Comment: The AS requests that MS to join the participant to the
   conference created earlier using the default bidirectional audio
   stream setup. The MS responds that the join was successful.

   AS -> MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <request>
      <dialogstart conferenceid="conf1"
        src="http://www.example.com/scripts/announce.vxml">
        <namelist>
         <item name="username"
          value="http://www.example.com/recordings/asswrw456grfg.wav"/>
        </namelist>
      </dialogstart>
    </request>
   </mscp>

   Comment: The AS requests the MS to play a dialog to the conference
   announcing the new participant.

   AS <- MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <response>
      <dialogstarted dialogid="dialog2" conferenceid="conf1"/>
    </response>
   </mscp>

   Comment: The MS responds that the dialog has started.

   AS <- MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <notification>
      <dialogexit dialogid="dialog2" conferenceid="conf1"/>
        </notification>
   </mscp>

   Comment: The MS notifies that the dialog has finished. Above steps
   are repeated as each participant joins (the dialog and connections
   ids would be different).

   AS -> MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <request>
      <unjoin id1="conf1" id2="a1e8ad50029@15.124.40.73;2134;1423" />
    </request>
   </mscp>

   AS <- MS
```

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferenceunjoined id1="conf1"
       id2="a1e8ad50029@15.124.40.73;2134;1423"/>
 </response>
</mscp>
```

Comment: The participant leaves the conference. This step is repeated as each participant leaves the conference.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
   <destroyconference conferenceid="conf1"/>
 </request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferencedestroyed conferenceid="conf1"/>
 </response>
</mscp>
```

Comment: Once all participants have left the conference, the conference is destroyed.

6.3. Manipulating Conference Participants

The following example shows how a conference participant's mute/unmute setting can be controlled by attaching a dialog to a participant's connection. The example assumes that the conference has already been created.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
   <dialogstart connectionid="a1e8ad50029@15.124.40.73;2134;1423"
     src="http://www.example.com/scripts/digitsLoop.vxml">
    <stream type="audio" direction="receive"/>
    <subscribe>
      <item name="dtmf" value="single"/>
    </subscribe>
   </dialogstart>
 </request>
</mscp>
```

```
   Comment: The AS requests the MS starts a dialog on the connection.
   The dialog is a VoiceXML script which listens for digits; when it
   receives a digit, it will notify the AS and continue listening for
   digits.

   AS <- MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <response>
      <dialogstarted dialogid="dialog5"
          connectionid="a1e8ad50029@15.124.40.73;2134;1423"/>
    </response>
   </mscp>

   Comment: The MS responds that the dialog has started.

   AS -> MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <request>
        <join id1="conf1" id2="a1e8ad50029@15.124.40.73;2134;1423"/>
    </request>
   </mscp>

   AS <- MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <response>
      <conferencejoined id1="conf1"
                           id2="a1e8ad50029@15.124.40.73;2134;1423"/>
    </response>
   </mscp>

   Comment: The participant is now joined to the conference in
   bi-directional audio mode. Note that audio output from the
   participant is streamed to both the dialog and conference.

   AS <- MS
   <mscp version="1.0" xmlns="urn:ietf:mscp">
    <notification>
      <dialoguser event="dtmf">
        <namelist>
         <item name="digit" value="1"/>
        </namelist>
      </dialoguser>
    </notification>
   </mscp>

   Comment: MS notifies the AS that the user has pressed 1.

   AS -> MS
```

```
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
     <join id1="conf1" id2="a1e8ad50029@15.124.40.73;2134;1423">
      <stream type="audio" direction="receive"/>
     </join>
 </request>
</mscp>
```

Comment: AS instructs the MS to (re)join the participant to the conference without the participant's input: i.e. they are on mute. Note that the participant's input is still streamed to the dialog, so their DTMF commands can be interpreted.

```
AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferencejoined id1="conf1"
       id2="a1e8ad50029@15.124.40.73;2134;1423"/>
 </response>
</mscp>
```

6.4. Sidebar Conference

In this example, a sidebar conference is setup and some participants are moved into the sidebar conference so they can chat privately. They are still able to hear the main conference. The example assumes that the participants are currently joined to the main conference and that a sidebar conference with the name 'conf2' has already been created.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
     <join id1="conf1" id2="conf2">
      <stream type="audio" direction="receive"/>
     </join>
 </request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferencejoined id1="conf1" id2="conf2"/>
 </response>
</mscp>
```

Comment: AS instructs MS to join the sidebar conference to the main conference so the sidebar conference can hear the main conference but

not vice versa.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
     <unjoin id1="conf1"
                id2="a1e8ad50029@15.124.40.73;2139;1429"/>
</request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferenceunjoined id1="conf1"
                id2="a1e8ad50029@15.124.40.73;2139;1429"/>
 </response>
</mscp>
```

Comment: participant on connection a1e8ad50029@15.124.40.73;2139;1429 is unjoined from the main conference.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
     <join id1="conf2" id2="a1e8ad50029@15.124.40.73;2139;1429"/>
</request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferencejoined id1="conf2"
           id2="a1e8ad50029@15.124.40.73;2139;1429"/>
 </response>
</mscp>
```

Comment: participant on connection a1e8ad50029@15.124.40.73;2139;1429 is joined to the sidebar conference.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
     <unjoin id1="conf1" id2="a1e8999999@15.124.40.73;2134;1423"/>
</request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
```

```
 <response>
   <conferenceunjoined id1="conf1"
         id2="a1e8999999@15.124.40.73;2134;1423"/>
 </response>
</mscp>
```

Comment: participant on connection a1e8999999@15.124.40.73;2134;1423 is unjoined from the main conference.

```
AS -> MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
     <join id1="conf2" id2="a1e8999999@15.124.40.73;2134;1423"/>
</request>
</mscp>

AS <- MS
<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
   <conferencejoined id1="conf2"
            id2="a1e8999999@15.124.40.73;2134;1423"/>
 </response>
</mscp>
```

Comment: participant on connection a1e8999999@15.124.40.73;2134;1423 is joined to the sidebar conference. The participants on connections a1e8ad50029@15.124.40.73;2139;1429 and a1e8999999@15.124.40.73;2134;1423 can now chat to each other privately while still hearing the main conference.

7. Transport Channel

By employing a self-contained XML representation for the MSCP messages, the protocol is independent of the underlying transport channel.

While SIP itself is not an ideal transport channel for a control protocol, the MSCP protocol MAY be transported in SIP INFO messages. However, protocol messages SHOULD be transported over a dedicated transport channel as described in Section 7.2. This has several advantages: it fits neatly within the architecture and goals of SIP, it avoids problems with MTU limits and sequencing when using SIP over UDP, it circumvents timing issues, and it avoids putting unnecessary burdens on SIP network elements such as proxy servers.

ISSUES:

1. A future version of this document will further clarify the relationship between the SIP channel and individual connections for conferencing.

2. A future version may also specify how SOAP can be used as a dedicated transport channel.

7.1. SIP Transport Channel

MSCP messages can be transported with SIP INFO messages. Requests, responses, and notification messages are carried in the body of separate SIP INFO messages.

To use this mechanism, the AS SHOULD initiate a media-less SIP dialog with the MS. Alternatively, protocol messages MAY be transported on the SIP dialogs managing individual participants sessions between the AS and MS. The SIP dialog used to send a response MUST be the same SIP session on which the request was sent. Note that in this case, there is no connection between the target of request messages / notifications and the particular SIP dialog.

Support for multipart mime payloads in the body of SIP messages may be necessary, but is not required by this protocol.

7.2. TCP Transport Channel

This transport channel requires the use of a connection-oriented transport layer such as TCP or SCTP to provide guaranteed delivery and sequencing of messages between the AS and MS. If security is required, TLS may be employed.

A simple transaction model is employed. The AS can send requests to the MS. A response message must be sent from the MS to the AS in reply to each request message. Notification messages may be sent at any time from the AS to the MS or from the MS to the AS.

7.2.1. Establishing the Transport Channel

The AS can locate the MS in the SIP network using standard SIP mechanisms [RFC3263]. The transport channel is established via the SDP offer/answer model [RFC3264] and the mechanism described in [RFC4145]. The transport channel is identified in the SDP offer and answer with an m-line of type "application". The port number in the offer is fixed at 9. The corresponding m-line in the answer specifies the MS port number for which the AS must connect to. The a=setup attribute MUST be "active" for the offer from the AS and MUST be "passive" for the answer from the MS.

The transport channel persists for the duration of the associated SIP dialog. If an endpoint determines that the TCP connection has been closed and it should be reestablished, it SHOULD perform a new offer/answer exchange using a connection value of 'new' for this m-line.

Example:

```
AS->MS:
      INVITE sip:ms@example.com SIP/2.0
      Via: SIP/2.0/TCP as.example.com:5060;
           branch=z9hG4bK776asdhds
      Max-Forwards: 70
      To: MS <msf@example.com>
      From: AS <sip:as.example.com>;tag=1928301774
      Call-ID: a84b4c76e66710
      CSeq: 314161 INVITE
      Contact: <sip:as@example.com>
      Content-Type: application/sdp
      Content-Length: ...

      v=0
      o=as 53655765 53655765 IN IP4 192.168.1.1
      s=Example
      c=IN 192.168.1.1
      m=application 9 TCP/MS
      a=setup:active
      a=connection:new

MS->AS:
      SIP/2.0 200 OK
      Via: SIP/2.0/TCP as.example.com:5060;
```

```
            branch=z9hG4bK776asdhds
          To: MS <ms@example.com>;tag=19
          From: AS <sip:as.example.com>;tag=1928301774
          Call-ID: a84b4c76e66710
          CSeq: 314161 INVITE
          Contact: <sip:ms1@example.com>
          Content-Type: application/sdp
          Content-Length: ...

          v=0
          o=ms 1435262 1435262 IN IP4 192.168.1.2
          s=Example
          c=IN IP4 192.168.1.2
          m=application 32416 TCP/MS
          a=setup:passive
          a=connection:new

    AS->MS:
          ACK sip:ms@example.com SIP/2.0
          Via: SIP/2.0/TCP as.example.com:5060;
               branch=z9hG4bK776asdhds
          Max-Forwards: 70
          To: MS <ms@example.com>;tag=19
          From: AS <sip:as.example.com>;tag=1928301774
          Call-ID: a84b4c76e66710
          CSeq: 314161 INVITE
          Content-Length: 0
```

7.2.2. Transport Channel PDU

The TCP transport channel PDU contains a textual header part in the ASCII character subset of UTF-8 and a binary body part.

The PDU header consists of a start-line, message-header fields (also known as "headers") delimited by CRLF, an empty line (i.e. a line with nothing preceding the CRLF) indicating the end of the header fields and a body containing a MIME entity.

The body is used for carrying the XML message payload but MAY contain additional messages by exploiting multipart MIME. All PDUs include the Content-Type and Content-Length headers and a body.

```
generic-message  =  start-line
                    message-header
                    CRLF
                    message-body

start-line       =  (request-line | response-line |
                    notification-line) CRLF

message-header   =  content-length
                    content-type

message-body     =  *OCTET

content-length   =  "Content-Length" ":" 1*DIGIT CRLF

content-type     =  "Content-Type" ":" media-type CRLF
```

7.2.3. Request

The request message is from the AS to the MS. The transport channel request consists of the request-line followed by message-header fields.

```
request-line     =  "REQ" SP version SP
                    SP request-id CRLF
```

The version field contains the protocol version, is included in request, response, notification messages, and takes the format:

```
version          =  "MSCP" "/" 1*DIGIT "." 1*DIGIT
```

This document specifies MSCP/1.0.

The request-id is used to correlate responses to specific requests. The initial value of the request-id is arbitrary. Consecutive requests MUST contain monotonically increasing request-id values. The MS MUST include the same request-id in its response.

```
request-id       =  1*DIGIT
```

7.2.4. Response

The response message is from the MS to the AS. Each request to the MS MUST be followed by a response with a matching request-id.

The transport channel response consists of the response-line followed by message-header fields.

```
response-line   =   version SP status-code SP reason-phrase
                    SP request-id CRLF
```

The status-code element is a 3-digit integer result code of the attempt to understand the request and is explained below.

```
status-code  =     "200" ; OK
               | "400" ; Bad Request
                       | "500" ; Internal Error

reason-phrase     =   "OK"
            | "Bad Request"
            | "Internal Error"
```

A status-code / reason-phrase of 200 OK implies the request was received and processed. The response is contained in the body.

A status-code / reason-phrase of 400 Bad Request implies the request had an error in it and could not be processed (for example, malformed XML). The error response is contained in the body.

A status-code / reason-phrase of 500 Internal Error implies the MS encountered an error when receiving and attempting to dispatch the request for processing. The error response is contained in the body.

7.2.5. Notification

The notification message is from the AS to the MS or from the MS to the AS. The transport channel event consists of the notification-line followed by message-header fields.

```
notification-line     =   "NOTIF" SP version CRLF
```

The notification is contained in the body.

7.2.6. Example Exchange

The following illustrates an example of the TCP transport channel for an AS that starts a dialog which plays a prompt to the user and collects DTMF digits. When the dialog terminates, a notification is sent to the AS to indicate termination of the dialog and includes the collected digits for processing by the AS.

```
AS->MS:

REQ MSCP/1.0 21091
Content-Type: application/mscp+xml
Content-Length: nnnn

<mscp version="1.0" xmlns="urn:ietf:mscp">
 <request>
<dialogstart connectionid="23142"
  src="http://www.example.com/scripts/promptAndCollect.vxml"/>
 </request>
</mscp>

MS->AS:

MSCP/1.0 200 OK 21091
Content-Type: application/mscp+xml
Content-Length: nnnn

<mscp version="1.0" xmlns="urn:ietf:mscp">
 <response>
    <dialogstarted dialogid="dialog1" connectionid="23142"/>
 </response>
</mscp>

MS->AS:

NOTIF MSCP/1.0
Content-Type: application/mscp+xml
Content-Length: nnnn

<mscp version="1.0" xmlns="urn:ietf:mscp">
 <notification>
  <dialogexit dialogid="dialog1" connectionid="23142">
   <namelist>
  <item name="accountnum" value="12345"/>
   </namelist>
  </dialogexit>
 </notification>
</mscp>
```

8. Formal Syntax

The XML schema for MSCP messages is specified below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:ietf:mscp"
elementFormDefault="qualified" xmlns="urn:ietf:mscp"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:annotation>
  <xsd:documentation> MSCP 1.0 schema (20050708) </xsd:documentation>
 </xsd:annotation>
 <xsd:element name="mscp">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element ref="request"/>
    <xsd:element ref="response"/>
    <xsd:element ref="notification"/>
    <xsd:any namespace="##other" processContents="strict"/>
   </xsd:choice>
   <xsd:attribute name="version" type="xsd:string" use="required"/>
   <xsd:anyAttribute namespace="##other" processContents="strict"/>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="request">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element ref="dialogstart"/>
    <xsd:element ref="dialogprepare"/>
    <xsd:element ref="createconference"/>
    <xsd:element ref="destroyconference"/>
    <xsd:element ref="join"/>
    <xsd:element ref="unjoin"/>
    <xsd:any namespace="##other" processContents="strict"/>
   </xsd:choice>
   <xsd:attribute name="id" type="xsd:ID"/>
   <xsd:anyAttribute namespace="##other" processContents="strict"/>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="response">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element ref="dialogprepared"/>
    <xsd:element ref="errordialognotprepared"/>
    <xsd:element ref="errordialogwrongstate"/>
    <xsd:element ref="dialogstarted"/>
    <xsd:element ref="errordialognotstarted"/>
```

```
      <xsd:element ref="conferencecreated"/>
      <xsd:element ref="errorconferencecreate"/>
      <xsd:element ref="conferencedestroyed"/>
      <xsd:element ref="errorconferencedestroy"/>
      <xsd:element ref="conferencejoined"/>
      <xsd:element ref="errorconferencejoin"/>
      <xsd:element ref="conferenceunjoined"/>
      <xsd:element ref="errorconferenceunjoin"/>
      <xsd:any namespace="##other" processContents="strict"/>
     </xsd:choice>
     <xsd:attribute name="id" type="xsd:ID"/>
     <xsd:anyAttribute namespace="##other" processContents="strict"/>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="notification">
    <xsd:complexType>
     <xsd:choice>
      <xsd:element ref="dialogterminate"/>
      <xsd:element ref="dialogexit"/>
      <xsd:element ref="errordialog"/>
      <xsd:element ref="dialoguser"/>
      <xsd:element ref="dialogdisconnect"/>
      <xsd:element ref="dialogtransfer"/>
      <xsd:element ref="dialogterminatetransfer"/>
      <xsd:element ref="dialogtransfercomplete"/>
      <xsd:element ref="conferenceuser"/>
      <xsd:any namespace="##other" processContents="strict"/>
     </xsd:choice>
     <xsd:anyAttribute namespace="##other" processContents="strict"/>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="dialogterminate">
    <xsd:complexType>
     <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:any namespace="##other" processContents="strict"/>
     </xsd:choice>
     <xsd:attributeGroup ref="DialogNotification.attribs"/>
     <xsd:attribute name="immediate" type="boolean.datatype"
     default="false"/>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="dialogexit">
    <xsd:complexType>
     <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="namelist"/>
      <xsd:any namespace="##other" processContents="strict"/>
     </xsd:choice>
     <xsd:attributeGroup ref="DialogNotification.attribs"/>
```

```
      <xsd:attributeGroup ref="DialogExtendedNotification.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errordialog">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogNotification.attribs"/>
      <xsd:attributeGroup ref="DialogExtendedNotification.attribs"/>
      <xsd:attribute name="reason" type="xsd:string" use="required"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialoguser">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="namelist"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="name" type="usereventname.datatype"
      use="required"/>
      <xsd:attributeGroup ref="DialogNotification.attribs"/>
      <xsd:attributeGroup ref="DialogExtendedNotification.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogdisconnect">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="namelist"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogNotification.attribs"/>
      <xsd:attributeGroup ref="DialogExtendedNotification.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogtransfer">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="namelist"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogNotification.attribs"/>
      <xsd:attributeGroup ref="DialogExtendedNotification.attribs"/>
      <xsd:attribute name="type" type="xsd:string" use="required"/>
      <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="maxtime" type="duration.datatype"
      use="required"/>
      <xsd:attribute name="connecttimeout" type="duration.datatype"
```

```
      use="required"/>
      <xsd:attribute name="aai" type="xsd:string"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogterminatetransfer">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogNotification.attribs"/>
      <xsd:attributeGroup ref="DialogExtendedNotification.attribs"/>
      <xsd:attribute name="reason" type="xsd:string" use="required"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogtransfercomplete">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogNotification.attribs"/>
      <xsd:attributeGroup ref="DialogExtendedNotification.attribs"/>
      <xsd:attribute name="reason" type="xsd:string" use="required"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="conferenceuser">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="namelist"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="name" type="usereventname.datatype"
      use="required"/>
      <xsd:attributeGroup ref="ConferenceNotification.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogprepare">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="namelist"/>
       <xsd:element ref="subscribe"/>
       <xsd:element ref="stream"/>
       <xsd:element ref="src"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="src" type="xsd:anyURI"/>
      <xsd:attribute name="type" type="xsd:string"
      default="application/voicexml+xml"/>
      <xsd:attribute name="connectionid" type="xsd:string"/>
```

```
      <xsd:attribute name="conferenceid" type="xsd:string"/>
      <xsd:attribute name="maxage" type="xsd:string"/>
      <xsd:attribute name="maxstale" type="xsd:string"/>
      <xsd:attribute name="enctype" type="xsd:string"
      default="application/x-www-form-urlencoded"/>
      <xsd:attribute name="method" type="method.datatype"
      default="get"/>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogprepared">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogResponse.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errordialognotprepared">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogResponse.attribs"/>
      <xsd:attributeGroup ref="ResponseError.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errordialogwrongstate">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogResponse.attribs"/>
      <xsd:attributeGroup ref="ResponseError.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogstart">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="namelist"/>
       <xsd:element ref="subscribe"/>
       <xsd:element ref="src"/>
       <xsd:element ref="stream"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="src" type="xsd:anyURI"/>
      <xsd:attribute name="type" type="xsd:string"
      default="application/voicexml+xml"/>
```

```
      <xsd:attribute name="prepareddialogid" type="xsd:string"/>
      <xsd:attribute name="connectionid" type="xsd:string"/>
      <xsd:attribute name="conferenceid" type="xsd:string"/>
      <xsd:attribute name="maxage" type="xsd:string"/>
      <xsd:attribute name="maxstale" type="xsd:string"/>
      <xsd:attribute name="enctype" type="xsd:string"
      default="application/x-www-form-urlencoded"/>
      <xsd:attribute name="method" type="method.datatype"
      default="get"/>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="dialogstarted">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogResponse.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errordialognotstarted">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="DialogResponse.attribs"/>
      <xsd:attributeGroup ref="ResponseError.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="createconference">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="subscribe"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="reservedtalkers"
      type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="reservedlisteners"
      type="xsd:nonNegativeInteger"/>
      <xsd:attribute name="reservedmedia" type="mediatype.datatype"
      default="audio"/>
      <xsd:attribute name="audiomixingpolicy"
      type="audiomixingpolicy.datatype" default="nbest"/>
      <xsd:attribute name="audiomixingnbest"
      type="xsd:nonNegativeInteger" default="0"/>
      <xsd:attribute name="activespeakernotification"
      type="xsd:nonNegativeInteger" default="0"/>
      <xsd:attribute name="videomixingpolicy"
```

```
      type="videomixingpolicy.datatype"/>
      <xsd:attribute name="videomixingvas"
      type="xsd:nonNegativeInteger"/>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="conferencecreated">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="ConferenceResponse.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errorconferencecreate">
     <xsd:complexType>
      <xsd:attributeGroup ref="ConferenceResponse.attribs"/>
      <xsd:attributeGroup ref="ResponseError.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="destroyconference">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="conferenceid" type="xsd:string"
      use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="conferencedestroyed">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="ConferenceResponse.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errorconferencedestroy">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="ConferenceResponse.attribs"/>
      <xsd:attributeGroup ref="ResponseError.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="join">
```

```
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="stream"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="id1" type="xsd:string" use="required"/>
      <xsd:attribute name="id2" type="xsd:string" use="required"/>
      <xsd:attribute name="mediapreferred" type="boolean.datatype"
      default="false"/>
      <xsd:attribute name="entertone" type="tone.datatype"
      default="true"/>
      <xsd:attribute name="exittone" type="tone.datatype"
      default="true"/>
      <xsd:attribute name="autoinputgain" type="boolean.datatype"
      default="true"/>
      <xsd:attribute name="autooutputgain" type="boolean.datatype"
      default="true"/>
      <xsd:attribute name="dtmfclamp" type="boolean.datatype"
      default="true"/>
      <xsd:attribute name="toneclamp" type="boolean.datatype"
      default="true"/>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="conferencejoined">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="JoinResponse.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errorconferencejoin">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="JoinResponse.attribs"/>
      <xsd:attributeGroup ref="ResponseError.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="unjoin">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attribute name="id1" type="xsd:string" use="required"/>
      <xsd:attribute name="id2" type="xsd:string" use="required"/>
```

```
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="conferenceunjoined">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="JoinResponse.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="errorconferenceunjoin">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:attributeGroup ref="JoinResponse.attribs"/>
      <xsd:attributeGroup ref="ResponseError.attribs"/>
     </xsd:complexType>
    </xsd:element>
    <!-- SHARED -->
    <xsd:element name="subscribe">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="item"/>
      </xsd:choice>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="namelist">
     <xsd:complexType>
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
       <xsd:element ref="item"/>
       <xsd:any namespace="##other" processContents="strict"/>
      </xsd:choice>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="item">
     <xsd:complexType>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="value" type="xsd:string" use="required"/>
      <xsd:anyAttribute namespace="##other" processContents="strict"/>
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="src">
     <xsd:complexType>
      <xsd:choice>
```

```
      <xsd:any namespace="##other" processContents="strict"/>
     </xsd:choice>
     <xsd:anyAttribute namespace="##other" processContents="strict"/>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="stream">
    <xsd:complexType>
     <xsd:attribute name="type" type="mediatype.datatype"
     use="required"/>
     <xsd:attribute name="direction" type="mediadirection.datatype"
     use="required"/>
     <xsd:anyAttribute namespace="##other" processContents="strict"/>
    </xsd:complexType>
   </xsd:element>
   <!-- ATTRIBUTE GROUPS -->
   <xsd:attributeGroup name="DialogResponse.attribs">
    <xsd:attribute name="dialogid" type="xsd:string" use="required"/>
    <xsd:attribute name="connectionid" type="xsd:string"/>
    <xsd:attribute name="conferenceid" type="xsd:string"/>
    <xsd:anyAttribute namespace="##other" processContents="strict"/>
   </xsd:attributeGroup>
   <xsd:attributeGroup name="ConferenceResponse.attribs">
    <xsd:attribute name="conferenceid" type="xsd:string"
    use="required"/>
    <xsd:anyAttribute namespace="##other" processContents="strict"/>
   </xsd:attributeGroup>
   <xsd:attributeGroup name="JoinResponse.attribs">
    <xsd:attribute name="id1" type="xsd:string" use="required"/>
    <xsd:attribute name="id2" type="xsd:string" use="required"/>
    <xsd:anyAttribute namespace="##other" processContents="strict"/>
   </xsd:attributeGroup>
   <xsd:attributeGroup name="ResponseError.attribs">
    <xsd:attribute name="reason" type="xsd:string"/>
   </xsd:attributeGroup>
   <xsd:attributeGroup name="DialogNotification.attribs">
    <xsd:attribute name="dialogid" type="xsd:string" use="required"/>
    <xsd:anyAttribute namespace="##other" processContents="strict"/>
   </xsd:attributeGroup>
   <xsd:attributeGroup name="DialogExtendedNotification.attribs">
    <xsd:attribute name="connectionid" type="xsd:string"/>
    <xsd:attribute name="conferenceid" type="xsd:string"/>
   </xsd:attributeGroup>
   <xsd:attributeGroup name="ConferenceNotification.attribs">
    <xsd:attribute name="conferenceid" type="xsd:string"
    use="required"/>
    <xsd:anyAttribute namespace="##other" processContents="strict"/>
   </xsd:attributeGroup>
   <!-- DATATYPES -->
```

```
    <xsd:simpleType name="mediadirection.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="both"/>
      <xsd:enumeration value="transmit"/>
      <xsd:enumeration value="receive"/>
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="mediatype.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="audio"/>
      <xsd:enumeration value="video"/>
      <xsd:enumeration value="audiovideo"/>
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="method.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="get"/>
      <xsd:enumeration value="post"/>
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="usereventname.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:pattern value="[a-zA-Z0-9\.]+"/>
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="boolean.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="true"/>
      <xsd:enumeration value="false"/>
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="tone.datatype">
     <xsd:union memberTypes="boolean.datatype xsd:anyURI"/>
    </xsd:simpleType>
    <xsd:simpleType name="audiomixingpolicy.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="nbest"/>
      <xsd:enumeration value="manual"/>
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="videomixingpolicy.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="vas"/>
      <xsd:enumeration value="manual"/>
      <xsd:enumeration value="userdefined"/>
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="duration.datatype">
```

```
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="(\+)?([0-9]*\.)?[0-9]+(ms|s)"/>
      </xsd:restriction>
   </xsd:simpleType>
   </xsd:schema>
```

9. Security Considerations

The MSCP protocol may carry sensitive application information such as usernames, passwords, confidential or private information, etc. For this reason, the AS and MS endpoints must have the option of secure communication for MSCP messages they send and receive. This can be achieved by using a transport channel which can be properly secured as described in Section 7.

10. IANA Considerations

The MSCP media type "application/mscp+xml" and the MSCP XML namespace "urn:ietf:mscp" may need to be registered with IANA.

11. Change Summary

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

- Corrected <join> parameter "wtoneclamp" to "toneclamp"
- Add Editor's Note in Section 2 describing how the next version of MSCP will be aligned with SIP Control Framework as well as its related IVR and conferencing packages.

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

- Miscellaneous typos corrected
- Changed <dialogterminatecomplete/> to <dialogtransfercomplete/> in 4.8.2 "Bridge transfer" in scenario 2 example code

12. Contributors

The editors gratefully acknowledge the following individuals and their companies who contributed to MSCP:

R. J. Auburn (Voxeo)

Hans Bjurstrom (Hewlett-Packard)

Dave Burke (Voxpilot)

Emily Candell (Comverse)

Jeff Haynie (Vocalocity)

Scott McGlashan (Hewlett-Packard)

Ken Rehor (Vocalocity)

Dave Renshaw (IBM)

Rao Surapaneni (Tellme Networks)

13. References

13.1. Normative References

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

13.2. Informative References

[BASIC_CONFERENCE_PACKAGE]
Boulton, C., Melanchuk, T., McGlashan, S., and A. Shiratzky, "A Conference Control Package for the Session Initiation Protocol (SIP)", draft-boulton-conference-control-package-01 (work in progress), November 2006.

[BASIC_IVR_PACKAGE]
Boulton, C., Melanchuk, T., McGlashan, S., and A. Shiratzky, "A Basic Interactive Voice Response (IVR) Control Package for the Session Initiation Protocol (SIP)", draft-boulton-ivr-control-package-03 (work in progress), March 2007.

[CCXML10] Auburn, R J., "Voice Browser Call Control: CCXML Version 1.0", W3C Working Draft (work in progress), January 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.

[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.

[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.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, February 2006.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

[SIPCONTROLFRAMEWORK] Boulton, C., Melanchuk, T., McGlashan, S., and A. Shiratzky, "A Control Framework for the Session Initiation Protocol (SIP)", draft-boulton-sip-control-framework-05 (work in progress), February 2007.

[VXML20] McGlashan, S., Burnett, D., Carter, J., Danielsen, P., Ferrans, J., Hunt, A., Lucas, B., Porter, B., Rehor, K., and S. Tryphonas, "Voice Extensible Markup Language (VoiceXML) Version 2.0", W3C Recommendation, March 2004.

[VXML_IVR_PACKAGE] Boulton, C., Melanchuk, T., McGlashan, S., and A. Shiratzky, "A VoiceXML Interactive Voice Response (IVR) Control Package for the Session Initiation Protocol (SIP)", draft-boulton-ivr-vxml-control-package-02 (work in progress), March 2007.

[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

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

Email: Scott.McGlashan@hp.com

R. J. Auburn
Voxeo
100 East Pine Street #600
Orlando, FL 32801
USA

Email: rj@voxeo.com

Dave Burke
Voxpilot
6 - 9 Trinity Street
Dublin 2
Ireland

Email: david.burke@voxpilot.com

Emily Candell
Comverse
100 Quannapowitt Parkway
Wakefield, MA 01880
USA

Email: Emily.Candell@comverse.com

Rao Surapaneni
Tellme Networks
1310 Villa Street
Mountain View, CA 94041
USA

Email: rao@tellme.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).