Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 12, 2008

S. McGlashan
Hewlett-Packard
T. Melanchuk
Rain Willow Communications
C. Boulton
Avaya
June 10, 2008

An Interactive Voice Response (IVR) Control Package for the Media Control Channel Framework
draft-ietf-mediactrl-ivr-control-package-00

Status of this Memo

Abstract

This document defines a Media Control Channel Framework Package for Interactive Voice Response (IVR) dialog interaction on media connections and conferences. The package defines dialog management elements for preparing, starting and terminating dialog interactions, as well as associated responses and notifications. While other dialog types can be used, this package defines a specific IVR dialog type supporting prompt playback, runtime controls, DTMF collect and media recording. The package also defines elements for auditing package capabilities and dialogs.

Table of Contents

1. Introduction

The Media Control Channel Framework ([I-D.ietf-mediactrl-sip-control-framework]) provides a generic approach for establishment and reporting capabilities of remotely initiated commands. The Control Framework utilizes many functions provided by the Session Initiation Protocol [RFC3261] (SIP) for the rendezvous and establishment of a reliable channel for control interactions. The Control Framework also introduces the concept of a Control Package. A Control Package is an explicit usage of the Control Framework for a particular interaction set. This document defines a Control Package for Interactive Voice Response (IVR) dialogs on media connections and conferences. The term 'IVR' is used in its inclusive sense, allowing media other than voice for dialog interaction.

The package defines dialog management elements for preparing, starting and terminating dialog interactions, as well as associated responses and notifications. While other dialog types can be used, this package defines a specific IVR dialog type supporting prompt playback, runtime controls, DTMF collect and media recording. The package also defines elements for auditing package capabilities and dialogs.

This package has been designed to satisfy the IETF MediaCtrl requirements ([RFC5167]) by building upon two major approaches to IVR dialog design. These approaches address a wide range of IVR use cases and are used in many applications which are extensively deployed today.

First, the package is designed to provide the major IVR functionality of SIP Media Server languages such as netann ([RFC4240]), MSCML ([RFC5022]) and MSML ([MSML]) which themselves build upon more traditional non-SIP languages ([H.248.9], [RFC2897]). A key differentiator is that this package provides IVR functionality using the Media Control Channel Framework.

Second, its design is aligned with key concepts of web model as defined in W3C Voice Browser languages. The key dialog management mechanism is closely aligned with CCXML ([CCXML10]). The dialog functionality defined in this package can be largely seen as a subset of VoiceXML ([VXML20], [VXML21]): where possible, basic prompting, DTMF collection and media recording features are incorporated, but not any advanced VoiceXML constructs (such as <form>, its interpretation algorithm, or a dynamic data model). As W3C develops VoiceXML 3.0, we expect to see further alignment, especially in providing a set of basic independent primitive elements (such as prompt, collect, record and runtime controls) which can be re-used in

different dialog languages.

By reusing and building upon design patterns from these approaches to IVR languages, this package is intended to provide a foundation which is familiar to current IVR developers and sufficient for most IVR applications, as well as a path to other languages which address more advanced applications.

This package defines an IVR dialog type. The scope of this dialog type is the following IVR functionality:

- playing one or more media resources as a prompt to the user
- runtime controls (including VCR controls like speed and volume)
- collecting DTMF input from the user according to a grammar
- recording user media input

Out of scope for this dialog type are more advanced functions including ASR (Automatic Speech Recognition), TTS (Text-to-Speech), VoiceXML, fax and media transformation. Such functionality may be

addressed by other dialog types used with this package, extension to this package or other packages.

The functionality of this package is defined by messages, containing XML [XML] elements, transported using the Media Control Channel Framework. The XML elements can be divided into three types: dialog management elements; a dialog element which defines the specific IVR operations carried within dialog management elements; and finally, elements for auditing package capabilities as well as dialogs managed by the package.

Dialog management elements are designed to manage the general lifecycle of a dialog. Elements are provided for preparing a dialog, starting the dialog on a conference or connection, and terminating execution of a dialog. Each of these elements is contained in a Media Control Channel Framework CONTROL message sent to the media server. When the appropriate action has been executed, the media server sends a REPORT message (or a 200 response to the CONTROL if it can execute in time) with a response element indicating whether the operation was successful or not (e.g. if the dialog cannot be started, then the error is reported in this response). Once a dialog has been successfully started, the media server may send further event notifications in a framework CONTROL message. This package defines two event notifications: a DTMF event indicating the DTMF activity; and a dialogexit event indicating that the dialog has exited. If the dialog has executed successful, the dialogexit event

includes information collected during the dialog. If an error occurs during execution (e.g. a media resource failed to play, no recording resource available, etc), then error information is reported in the dialogexit event. Once a dialogexit event is sent, the dialog lifecycle is terminated.

Specific dialog types are referenced or contained within dialog management elements for preparing and starting dialogs. While other dialog types defined elsewhere can be used, this package defines a IVR dialog type which contains child elements for playing prompts to the user, specifying runtime controls, collecting DTMF input from the user and recording media input from the user. The child elements can co-occur in this dialog type so as to provide 'play announcement', 'prompt and collect' as well as 'prompt and record' functionality.

The document is organized as follows. Section 3 descibes how this control package fulfills the requirements for a Media Control Channel Framework control package. Section 4 describes the syntax and semantics of defined elements, including dialog management (Section 4.2), the IVR dialog element (Section 4.3) and audit elements (Section 4.4). Section 5 describes an XML schema for these elements and provides extensibility by allowing attributes and elements from other namespaces. Section 6 provides examples of package usage.

2. Conventions and Terminology

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

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

Dialog: A dialog performs media interaction with a user. A dialog is specified as inline XML, or via a URI reference to an external XML document type. Traditional IVR dialogs typically feature capabilities such as playing audio prompts, collecting DTMF input and recording audio input from the user. More inclusive definitions may include support for other media types, runtime controls, synthesized speech, recording and playback of video, recognition of spoken input, and mixed initiative conversations.

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 initiated by the application server.

3. Control Package Definition

This section fulfills the mandatory requirement for information that MUST be specified during the definition of a Control Framework Package, as detailed in Section 8 of [I-D.ietf-mediactrl-sip-control-framework].

3.1. Control Package Name

The Control Framework requires a Control Package to specify and register a unique name and version.

The name and version of this Control Package is "msc-ivr/1.0" (Media Server Control - Interactive Voice Response - version 1.0). Its IANA registration is specified in Section 8.1.

3.2. Framework Message Usage

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

This package specifies CONTROL and reponse messages in terms of XML elements defined in Section 4. These elements describe requests, response and notifications and all are contained within a root <mscivr> element (Section 4.1).

In this package, the MS operates as a Control Framework Server in receiving requests from, and sending responses to, the AS (operating as Control Framework Client). Dialog management requests and responses are defined in Section 4.2. Audit requests and responses are defined in Section 4.4. dialog management and audit responses are carried in a framework 200 response or REPORT message bodies. This package's response codes are defined in Section 4.5.

Note that package responses are different from framework response codes. Framework error response codes (see Section 8 of [I-D.ietf-mediactrl-sip-control-framework]) are used when the request or event notification is invalid; for example, a request is invalid XML (400), or not understood (500).

The MS also operates as a Control Framework Client in sending event notification to the AS (Control Framework Server). Event notifications (Section 4.2.5) are carried in CONTROL message bodies. The AS MUST respond with a Control Framework 200 response.

3.3. Common XML Support

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

This package requires that the XML Schema in Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework] MUST be supported for media dialogs and conferences.

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

3.4. CONTROL Message Body

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

When operating as Control Framework Server, the MS receives CONTROL messages body containing an <mscivr> element with either a dialog management or audit request child element.

The following dialog management request elements are carried in CONTROL message bodies to MS: <dialogprepare> (Section 4.2.1), <dialogstart> (Section 4.2.2) and <dialogterminate> (Section 4.2.3)elements.

The <audit> request element (Section 4.4.1) is also carried in CONTROL message bodies.

When operating as Control Framework Client, the MS sends CONTROL messages with a body containing a notification <event> element (Section 4.2.5).

3.5. REPORT Message Body

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

When operating as Control Framework Server, the MS sends REPORT bodies containing a <mscivr> element with a response child element.

The response element for dialog management requests is a <response> element (Section 4.2.4). The response element for an audit request is a <auditresponse> element (Section 4.4.2).

3.6. Audit

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

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

The syntax and semantics of audit request and response elements is defined in Section 4.4.

4. Element Definitions

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

The root element is <mscivr> (Section 4.1). All other XML elements (requests, responses and notification elements) are contained within it. Child elements describe dialog management (Section 4.2) and audit (Section 4.4) functionality. The IVR dialog element (contained within dialog management elements) is defined in Section 4.3. Response status codes are defined in Section 4.5 and type definitions in Section 4.6.

Implementation of this control package MUST adhere to the syntax and semantics of XML elements defined in this section and the schema (Section 5). The XML schema supports extensibility by allowing attributes and elements from other namespaces. Implementations MAY support attributes and elements from other namespaces. If an implementation encounters attributes or elements from another namespace which it does not understand, it MUST ignore them and continue processing.

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

Usage examples are provided in Section 6.

4.1. <mscivr>

The <mscivr> element has the following attributes (in addition to

standard XML namspace attributes such as xmlns):

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

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

1. dialog management elements defined in Section 4.2:

    <dialogprepare> prepare a dialog. See Section 4.2.1

    <dialogstart> start a dialog. See Section 4.2.2

    <dialogterminate> terminate a dialog. See Section 4.2.3

    <response> response to a dialog request. See Section 4.2.4

    <event> dialog or subscription notification. See Section 4.2.5

2. audit elements defined in Section 4.4:

    <audit> audit package capabilities and managed dialogs. See Section 4.4.1

    <auditresponse> response to an audit request. See Section 4.4.2

For example, a request to the MS to start an IVR dialog playing a prompt:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart>
  <dialog>
   <prompt>
    <media src="http://www.example.com/welcome.wav"/>
   </prompt>
  </dialog>
```

```
</mscivr>
```

and a response from the MS that the dialog started sucessfully:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="200" dialogid="d1"/>
</mscivr>
```

and finally a notification from the MS indicating that the dialog exited upon completion of playing the prompt:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="d1">
  <dialogexit status="1">
   <promptinfo termmode="completed"/>
  </dialogexit>
 </event>
</mscivr>
```


4.2. Dialog Management Elements

This section defines the dialog management XML elements for this control package. These elements are divided into requests, responses and notifications.

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

<dialogprepare>: prepare a dialog for later execution

<dialogstart>: start a dialog on a connection or conference

<dialogterminate>: terminate an IVR dialog

The MS MUST support the IVR dialog type defined in Section 4.3. The MS MAY support other dialog types.

Responses from the MS describe the status of the requested operation. Responses are specified in a <response> element (Section 4.2.4). The MS MUST respond to a request message with a response message. If the MS is not able to carry out the requested dialog operation, it is an error and the MS MUST indicate the error in the status code of the response.

Notifications are sent from the MS to provide updates on the status of a dialog or operations defined within the dialog. Notifications are specified in an <event> element (Section 4.2.5). The MS MUST always send a dialog exit notification (Section 4.2.5.1) when the dialog exits, including when runtime errors occur. The MS MUST send DTMF subscription notifications (Section 4.2.5.2) when they occur during the dialog.

The MS MUST adhere to the following life cycle for dialog. Each dialog has the following state machine:

IDLE: the dialog is uninitialized.

PREPARING: the dialog is being prepared. If an error occurs the dialog transitions to the TERMINATED state and the MS MUST send a response indicating the error type.

PREPARED: the dialog has been successfully prepared and has a valid dialog identifier. The MS MUST send a response indicating the prepare operation was successful. If the duration the dialog remains in the PREPARED state exceeds the maximum preparation duration, the dialog transitions to the TERMINATED state and the

MS MUST send a dialogexit notification with an error status code (Section 4.5). A maximum preparation duration of 30s is RECOMMENDED.

STARTING: the dialog is being started. If an error occurs the dialog transitions to the TERMINATED state and the MS MUST send a response indicating the error type.

STARTED: the dialog has been successfully started and is now active. The MS MUST send a response indicating the start operation was successful. If any dialog events occurs which were subscribed to,

the MS MUST send a notifications when the dialog event occurs. When the dialog exits, the MS MUST send a dialogexit notification event and the dialog transitions to the TERMINATED state.

TERMINATED: the dialog is terminated and its dialogid is no longer valid. Dialog notifications MUST NOT be sent for this dialog.

It is an error to prepare or start a dialog with the same dialogid as that of a dialog on the MS which is in the state PREPARING, PREPARED, STARTING or STARTED: the MS MUST send a response with a status code 401. Once a dialog is in a TERMINATED state, its dialogid can be reused.

[Editors note: IVR-200. Would this description benefit from a state machine diagram?]

4.2.1. <dialogprepare>

The <dialogprepare> request is sent to the MS to request preparation of a dialog. Dialog preparation consists of (a) retrieving external dialog document and resources (if required), and (b) validating the dialog document syntactically and semantically.

A prepared dialog is executed when the MS receives a <dialogstart> request referencing the prepared dialog identifier (see Section 4.2.2).

The <dialogprepare> element has the following attributes:

src: specifies the location of an external dialog document to prepare. A valid value is a URI (see Section 4.6.9). It is an error if the document cannot be retrieved or processed (e.g. URI protocol not supported). The attribute is optional. There is no default value.

type: specifies the type of the external dialog document indicated in the 'src' attribute. The MS MAY use the value to assist the remote source in selecting the appropriate resource type (e.g. with HTTP 'accept' header) and to determine how the document is to

be processed. A valid value is a MIME type (see Section 4.6.10). The attribute is optional. The default value is the mime type "application/msc-ivr+xml" (Section 8.3) associated with the IVR dialog type (Section 4.3).

fetchtimeout: the maximum interval to wait when fetching an external dialog document. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 30s.

dialogid: string indicating a unique name for the dialog. It is an error if a dialog with the same name already exists on the MS. If this attribute is not specified, the MS MUST create a unique name for the dialog. The value is used in subsequent references to the dialog (e.g. as dialogid in a <response>). The attribute is optional. There is no default value.

The <dialogprepare> element has one optional child element:

<dialog> an IVR dialog (Section 4.3.1) to prepare. The element is optional.

The dialog to prepare can either be specified inline as a child element or externally using the src attribute (but not both). It is an error if both an inline <dialog> element and a src attribute are specified. The type and fetchtimeout attributes are only relevant when a dialog is specified as an external document.

For example, a <dialogprepare> request to prepare an inline IVR dialog with a single prompt:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogprepare>
  <dialog>
   <prompt>
    <media src="http://www.example.com/welcome.wav"/>
   </prompt>
  </dialog>
 </dialogprepare>
</mscivr>
```

In this example, a request to prepare a dialog type located externally with a specified dialogid;

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogprepare dialogid="d2" type="application/voicexml+xml"
                src="http://www.example.com/mydialog.vxml"
                fetchtimeout="15s"/>
</mscivr>
```

Since MS support for dialog types other than the IVR dialog type is optional, if the MS does not support the dialog type, it MUST send a response with the status code 409 (dialog type not supported).

4.2.2. <dialogstart>

The <dialogstart> element is sent to the MS to start a dialog. If the dialog has not been prepared, the dialog is prepared (retrieving an external document and resources if necessary, and the dialog document validated syntactically and semantically). Media processors (e.g. DTMF and prompt queue) are activated and associated with the specified connection or conference.

The <dialogstart> element has the following attributes:

src: specifies the location of an external dialog document to start. A valid value is a URI (see Section 4.6.9). It is an error if the document cannot be retrieved or processed (e.g. URI protocol not supported). The attribute is optional. There is no default value.

type: specifies the type of the external dialog document indicated in the 'src' attribute. The MS MAY use the value to assist the remote source in selecting the appropriate resource type (e.g. with HTTP 'accept' header) and to determine how the document is to be processed. A valid value is a MIME type (see Section 4.6.10). The attribute is optional. The default value is the mime type "application/msc-ivr+xml" (Section 8.3) associated with the IVR dialog type (Section 4.3).

fetchtimeout: the maximum interval to wait when fetching an external dialog document. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 30s.

dialogid: string indicating a unique name for the dialog. It is an error if a dialog with the same name already exists on the MS. If neither the dialogid attribute nor the prepareddialogid attribute is specified, the MS MUST create a unique name for the dialog. The value is used in subsequent references to the dialog (e.g. as dialogid in a <response>). The attribute is optional. There is

no default value.

prepareddialogid: string identifying a dialog previously prepared using a dialogprepare request. The attribute is optional. There is no default value.

connectionid: string identifying the SIP dialog connection on which this dialog is to be started (see Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework]). The attribute is optional. There is no default value.

conferenceid: string identifying the conference on which this dialog is to be started (see Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework]). The attribute is optional. There is no default value.

Exactly one of the connectionid or conferenceid attributes MUST be specified. It is an error to specify both connectionid and conferenceid attributes or neither.

It is an error if the connection or conference referenced by a specific connectionid or conferenceid attribute is not available on the MS at the time the <dialogstart> request is executed.

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

<dialog>: specifies the IVR dialog (Section 4.3.1) to execute. The element is optional.

<subscribe>: specifies subscriptions to dialog events (Section 4.2.2.1). The element is optional.

<params>: specifies input parameters (Section 4.2.6) for a dialog types defined outside this specification. The element is optional.

<stream>: determines the media stream(s) associated with the connection or conference on which the dialog is executed (Section 4.2.2.2). The <stream> element is optional. Multiple <stream> elements may be specified.

The dialog to start on a connection or conference can be specified either inline, or externally, or reference a previously prepared dialog. Exactly one of the src attribute, the prepareddialogid or a <dialog> child element MUST be specified. If the prepareddialogid is specified, it is an error to specify the src attribute, the dialogid attribute or a dialog type child element. If the src attribute is specified, it is an error to specify the prepareddialogid attribute, or a dialog type child element. If a <dialog> child element is

specified, it is an error to specify the src attribute or the prepareddialogid attribute. The type and fetchtimeout attributes are only relevant when a dialog is specified as an external document.

The <stream> element provides explicit control over which media streams on the connection or conference are used during dialog execution. For example, if a connection supports both audio and video streams, a <stream> element could be used to indicate that only the audio stream is used in receive mode. In cases where there are multiple media streams of the same type for a dialog, it is RECOMMENDED that the configuration is explicitly specified using <stream> elements. If no <stream> elements are specified, then the default media configuration is that defined for the connection or conference.

It is an error if a <stream> element is in conflict with (a) another <stream> element, (b) with specified connection or conference media capabilities, (c) with a SDP label value as part of the connectionid (see Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework]) or (d) if the media stream configuration is not supported by the MS. If the MS does not support a requested <stream> configuration, it MUST response with a <response> with a 413 status code.

This specification allows multiple, simultaneous dialogs to be started on the same connection or conference. It is RECOMMENDED the MS support the following cases:

1. different media streams used in different dialogs; e.g. audio only on one dialog and video only on another dialog

2. the same media stream received by different dialogs

3. use of implicit mixing (where appropriate) when the same type of

media stream is sent from different dialogs

If the MS does not support starting another dialog on the same connection or conference it MUST report an error when starting that dialog.

[Editors Note: IVR-201. If the conference mixer package supports video layout regions, then we need a mechanism to specify that the dialog is to be started on a specific region of the conference.]

For example, a request to start an ivr dialog on a connection subscribing to DTMF notificatons:


```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="connection1">
  <dialog>
   <prompt>
    <media src="http://www.example.com/getpin.wav"/>
   </prompt>
   <collect maxdigits="2"/>
  </dialog>
 <subscribe>
  <dmtfnotify matchmode="all"/>
 </subscribe>
 </dialogstart>
</mscivr>
```

In this example, the dialog is started on a conference where only audio media stream is received:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart conferenceid="conference1">
  <dialog>
   <record maxtime="384000s"/>
  </dialog>
  <stream media="audio" direction="recvonly"/>
</mscivr>
```

4.2.2.1. <subscribe>

The <subscribe> element allows the AS to subscribe to, and be notified of, specific events which occur during execution of the dialog. Notifications of dialog events are delivered using the <event> element (see Section 4.2.5).

The <subscribe> element has no attributes.

The <subscribe> element has the following sequence of child elements (0 or more occurences):

<dtmfsub>: Subscription to DTMF input during the dialog (Section 4.2.2.1.1). The element is optional.

The MS MUST support a <dtmfsub> subscription. It MAY support other dialog subscriptions. If the MS does not support a requested subscription, it MUST send a <response> with a 412 status code.

4.2.2.1.1. <dtmfsub>

The <dtmfsub> element has the following attributes:

matchmode: controls which DTMF input are subscribed to. Valid values are: "all" - notify all DTMF keypresses received during the dialog; "collect" - notify only DTMF input matched by the collect operation (Section 4.3.1.3); and "control" - notify only DTMF input matched by the runtime control operation (Section 4.3.1.2). The attribute is optional. The default value is "all".

The <dtmfsub> element has no child elements.

DTMF notifications are delivered in the <dtmfnotify> element (Section 4.2.5.2).

For example, the AS wishes to subscribe to DTMF keypress matching a runtime control:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart dialogid="d3" connectionid="connection1">
  <dialog>
   <prompt>
```

```
      <media src="http://www.example.com/getpin.wav"/>
     </prompt>
     <control ffkey="2" rewkey="3"/>
    </dialog>
   <subscribe>
    <dmtfnotify matchmode="control"/>
   </subscribe>
   </dialogstart>
  </mscivr>
```

Each time a '2' or '3' DTMF input is received, the MS sends a notification event:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="d3">
   <dtmfnotify matchmode="collect" dtmf="2"
       timestamp="2008-05-12T12:13:14Z"/>
 /event>
</mscivr>
```

4.2.2.2. <stream>

the <stream> element has the following attributes:

media: a string indicating the type of media associated with the stream. It is strongly RECOMMENDED that the following values are used for common types of media: "audio" for audio media, and "video" for video media. The attribute is mandatory.

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

direction: a string indicating the direction of the media flow between a dialog and its end point conference or connection. Defined values are: "sendrecv" (media can be sent and received), "sendonly" (media can only be sent), "recvonly" (media can only be received) and "inactive" (stream is not to be used). The default value is "sendrecv". The attribute is optional.

the <stream> element has no child elements.

For example, assume a user agent connection with multiple audio and video streams associated with the user and a separate web camera. In this case, the dialog could be started to record only the audio and video streams associated with the user:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="connection1">
  <dialog>
   <record maxtime="384000s"/>
  </dialog>
  <stream media="audio" label="camaudio" direction="inactive"/>
  <stream media="video" label="camvideo" direction="inactive"/>
  <stream media="audio" label="useraudio" direction="recvonly"/>
  <stream media="video" label="uservideo" direction="recvonly"/>
</mscivr>
```

4.2.3. <dialogterminate>

A dialog that has been successfully prepared or started can be terminated by sending a <dialogterminate> request element to the MS.

The <dialogterminate> element has the following attributes:

dialogid: string identifying the dialog to terminate. It is an error if the dialog identifier is not valid. The attribute is mandatory.

immediate: indicates whether the dialog is to be terminated immediately or not. A valid value is a boolean (see Section 4.6.1). A value of true indicates that the dialog is terminated immediately and a dialogexit notification without report information MUST be sent. A value of false indicates that the dialog terminates after the current iteration and the MS MUST send a dialogexit notification with report information. The attribute is optional. The default value is false.

When an MS receives a <dialogterminate> request, the dialog MUST be in a PREPARED, STARTING or STARTED states; otherwise, the MS MUST send a response with the status code 402. If it is in a PREPARED state, then it transitions immediately to the TERMINATED state. If it is in STARTING state, then any further starting (or preparation)

of the dialog is canceled, the dialog transitions to the TERMINATED state and the MS MUST send a response with the status code 414 for the dialogstart request. If the dialog is in a STARTED state, then the dialog is terminated according to the value of the immediate attribute.

The MS MUST reply to <dialogterminate> request with a <response> element (Section 4.2.4), reporting whether the dialog was stopped successful or not.

For example, immediately terminating a dialog with dialogid "d4":

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogterminate dialogid="d4" immediate="true"/>
</mscivr>
```

If the dialog is terminated successfully, then the response to the dialogterminate request would be:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="200" dialogid="d4"/>
 <dialogterminate dialogid="d4" immediate="true"/>
</mscivr>
```

4.2.4. <response>

Responses to dialog management requests are specified with a <response> element.

The <response> element has following attributes:

status: numeric code indicating the response status. Valid values are defined in Section 4.5. The attribute is mandatory.

reason: string specifying a reason for the response status. The attribute is optional. There is no default value.

dialogid: string identifying the dialog. If the request specifies a dialogid, then that value is used. Otherwise, with <dialogprepare> and <dialogstart> requests, the dialogid generated by the MS is used. If there is no available dialogid (e.g. a <dialogterminate> request with no dialogid attribute specified), then the value is the empty string. The attribute is mandatory.

connectionid: string identifying the SIP dialog connection associated with the dialog (see Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework]). The attribute is optional. There is no default value.

conferenceid: string identifying the conference associated with the dialog (see Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework]). The attribute is optional. There is no default value.

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

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="200" dialogid="d5"/>
</mscivr>
```

The response if dialog preparation failed due to an unsupported dialog type:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="409" dialogid="d5"
    reason="Unsupported dialog type: application/voicexml+xml"/>
</mscivr>
```

In this example a <dialogterminate> request does not specify a dialogid:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogterminate/>
</mscivr>
```

The response status indicates a 409 error (attribute required) and dialogid attribute has an empty string value:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="408" dialogid=""
    reason="Attribute required: dialogid"/>
</mscivr>
```

4.2.5. <event>

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

The <event> element has the following attributes:

dialogid: string identifying the dialog which generated the event. The attribute is mandatory.

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

<dialogexit>: indicates that the dialog has exited (Section 4.2.5.1).

<dtmfnotify>: indicates that a DTMF key press occured (Section 4.2.5.2).

4.2.5.1. <dialogexit>

The <dialogexit> event indicates that a prepared or active dialog has exited because it is complete, has been terminated, or because an error occurred during execution (for example, a media resource cannot be played). This event MUST be sent by the MS when the dialog exits.

The <dialogexit> element has the following attributes:

status: a status code indicating success or failure of the dialog. A valid value is a non-negative integer (see Section 4.6.4). A value of 0 indicates that the dialog has been terminated by a <dialogterminate> request. A value of 1 indicates success. A value of 2 indicates that the dialog terminated because the connection or conference associated with the dialog has terminated. A value of 3 indicates the dialog terminated due to exceeding its maximum duration. A value of 4 indicates the dialog terminated due to an execution error. Any other value indicates an error defined by the MS. The attribute is mandatory.

reason: a textual description providing a reason for the status code; e.g. details about an error. A valid value is a string (see Section 4.6.6). The attribute is optional. There is no default value.

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

<promptinfo>: report information (Section 4.3.2.1) about the prompt execution in an IVR <dialog>. The element is optional.

<controlinfo>: reports information (Section 4.3.2.2) about the control execution in an IVR <dialog>. The element is optional.

<collectinfo>: reports information (Section 4.3.2.3) about the collect execution in an IVR <dialog>. The element is optional.

<recordinfo>: reports information (Section 4.3.2.4) about the record execution in an IVR <dialog>. The element is optional.

<params>: reports exit parameters (Section 4.2.6) for a dialog type defined outside this specification. The element is optional.

For example, an active <dialog> exits normally the MS sends a dialogexit <event> reporting information:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="d6"/>
  <dialogexit status="1">
   <collectinfo dtmf="1234"  termmode="match"/>
  </dialogexit>
 </event>
```

4.2.5.2. <dtmfnotify>

The <dtmfnotify> element provide a notification of DTMF input received during the active dialog as requested by a <dtmfsub> subscription (Section 4.2.2.1).

The <dtmfnotify> element has the following attributes:

matchmode: indicates the matching mode specified in the subscription request. Valid values are: "all" - all DTMF keypresses notified; "collect" - only DTMF input matched by the collect operation notified; and "control" - only DTMF input matched by the control operation notified. The attribute is optional. The default value is "all".

dtmf: DTMF keypress received. A valid value is a DTMF string (see

Section 4.6.3) with no space between characters. The attribute is mandatory.

timestamp: indicates the time (on the MS) at which the key press occured. A valid value is a dateTime expression (Section 4.6.12). The attribute is mandatory.

For example, a notification of a DTMF matched during the collect operation:


```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="d3">
   <dtmfnotify matchmode="collect" dtmf="3"
     timestamp="2008-05-12T12:13:14Z"/>
 /event>
</mscivr>
```

4.2.6. <params>

The <params> element is a container for <param> elements (Section 4.2.6.1).

The <params> element has no attributes, but the following child elements are defined (0 or more):

<param>: specifies a parameter name and value (Section 4.2.6.1).

For example, usage with a dialog type defined outside this specification to send additional parameters into the dialog:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
<dialogstart type="application/x-dialog"
             src="nfs://nas01/dialog4" connectionid="c1">
  <params>
   <param name="mode">playannouncement</param>
   <param name="prompt1">nfs://nas01/media1.3gp"</param>
   <param name="prompt2">nfs://nas01/media2.3gp"</param>
  </params>
```

```
 </dialogstart>
</mscivr>
```

4.2.6.1. <param>

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

The <param> element has the following attributes:

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

type: specifies a mimetype of the parameter value. A valid value is a MIME type (see Section 4.6.10). The attribute is optional. There is no default value.

valuetype: a string indicating the type of the parameter value. The attribute is optional. The default value is a string type.

If the MS supports parameters, it MUST support the string valuetype

and MAY support other parameter valuetypes (e.g. base64Binary for inline binary data, such as a media recording).

The <param> element content model is the value of the parameter.

For example, usage with a dialog type defined outside this specification to receive parameters from the dialog when it exits:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="d6"/>
  <dialogexit status="1">
  <params>
   <param name="mode">recording</param>
   <param name="recording1" mimetype="audio/x-wav"
          valuetype="base64Binary">
   R0lGODlhZABqALMAAFrMYr/BvlKOVJKOg2xZUKmenMfDw8tgWJpV
   </param>
  </params>
  </dialogexit>
 </event>
</mscivr>
```

4.3. IVR Dialog Elements

This section describes the IVR dialog type. The MS MUST support this dialog type.

The <dialog> element is an execution container for operations of playing prompts (Section 4.3.1.1), runtime controls (Section 4.3.1.2), collecting DTMF (Section 4.3.1.3),and recording user input (Section 4.3.1.4. Results of the dialog execution (Section 4.3.2) are reported in a dialogexit notification event.

Using these elements, three common dialog models are supported:

playannouncements: only a <prompt> element is specified in the container. The prompt media resources are played in sequence.

promptandcollect: a <collect> element is specified and, optionally, a <prompt> element. If a <prompt> element is specified and bargein is enabled, playing of the prompt is terminated when bargein occurs, and DTMF collection is initiated; otherwise, the prompt is played to completion before DTMF collection is initiated. If no prompt element is specified, DTMF collection is initiated immediately.

promptandrecord: a <record> element is specified and, optionally, a <prompt> element. If a <prompt> element is specified and bargein is enabled, playing of the prompt is terminated when bargein occurs, and recording is initiated; otherwise, the prompt is played to completion before recording is initiated. If no prompt element is specified, recording is initiated immediately.

In addition, this dialog type supports runtime ('VCR') controls enabling a user to control prompt playback using DTMF.

Each of the core elements - <prompt>, <control>, <collect> and <record> - are specified so that their execution and reporting is largely self-contained. This facilitates their re-use in other dialog container elements. Note that DTMF and bargein behavior

affects multiple elements and is addressed in the relevant element definitions.

Execution results are reported in the <dialogexit> notification event with child elements defined in Section 4.3.2. If the dialog terminated normally (i.e. not due to an error or to a <dialogterminate> request), then the MS MUST report the results for the operations specified in the dialog:

<prompt>: <promptinfo> (see Section 4.3.2.1) with at least the termmode attribute specified.

<control>: <controlinfo> (see Section 4.3.2.2) if any runtime controls are matched.

<collect>: <collectinfo> (see Section 4.3.2.3) with the dtmf and termmode attributes specified.

<record>: <recordinfo> (see Section 4.3.2.4) with at least the recording, type and termmode attributes specified.

The media format requirements for IVR dialogs are undefined. This package is agnostic to the media types and codecs for media resources and recording which need to be supported by an implementation. For example, a MS implementation may choose to support only audio and in particular the 'audio/basic' codec for media playback and recording. However, when executing a dialog, if an MS encounters a media type or codec which it cannot process, the MS MUST stop further processing and report the error using the dialogexit notification.

4.3.1. <dialog>

An IVR dialog to play prompts to the user, allow runtime controls, collect DTMF or record input. The dialog is specified using a

<dialog> element.

A <dialog> element has the following attributes:

repeatCount: number of times the dialog is to be executed. A valid value is a non-negative integer (see Section 4.6.4). A value of 0 indicates that the dialog is repeated until halted by other means.

The attribute is optional. The default value is 1.

repeatDur: maximum duration for dialog execution. A valid value is a Time Designation (see Section 4.6.7). If no value is specified, then there is no limit on the duration of the dialog. The attribute is optional. There is no default value.

The repeatDur attribute takes priority over the repeatCount attribute in determining maximum duration of the dialog. See 'repeatCount' and 'repeatDur' in SMIL ([W3C.REC-SMIL2-20051213]) for further information.

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

<prompt>: defines media resources to play in sequence (see Section 4.3.1.1). The element is optional.

<control>: defines how DTMF is used for runtime controls (see Section 4.3.1.2). The element is optional.

<collect>: defines how DTMF is collected (see Section 4.3.1.3). The element is optional.

<record>: defines how recording takes place (see Section 4.3.1.4). The element is optional.

It is an error if no child element is specified. The behavior is not defined if both <collect> and <record> are specified.

The IVR dialog has the following execution model after initialization (initialization errors are reported in the response):

1. If an error occurs during execution, then the dialog terminates and the error is reported in the <dialogexit> event by setting the status attribute (see Section 4.3.2). Details about the error are specified in the reason attribute.

2. A counter is initialized to 0.

3. A duration timer is started for the value of the repeatDur attribute. If the timer expires before the dialog is complete,

then dialog is terminated and the dialogexit status attribute is set to 3 (see Section 4.2.5.1). The dialogexit MAY report information gathered in the last execution cycle (if any).

4. A dialog execution cycle is initiated. Each cycle executes the operations associated with the child elements of the dialog. If subscriptions are specified for the dialog, then a notification event is sent when the specified event occurs. If a child element reports an execution error, dialog execution is terminated (other child element operations are stopped) and the dialogexit status event sent, reporting any information gathered.

5. If the dialog execution cycle completes successfully, then the counter is incremented by one. If the value of the repeatCount attribute is greater than zero and the counter is equal to the value of the repeatCount attribute, then dialog execution is terminated and the dialogexit (with a status of 1) reports operation information collected in the dialog execution cycle. Otherwise, another dialog execution cycle is initiated.

4.3.1.1. <prompt>

The <prompt> element specifies a sequence of media resources to play.

A <prompt> element has the following attributes:

xml:base: A string declaring the base URI from which relative URIs in child elements are resolved prior to fetching. A valid value is a URI (see Section 4.6.9). The attribute is optional. There is no default value.

bargein: Indicates whether user input stops prompt playback unless the input is assocated with a specified runtime <control> operation (input matching control operations never interrupts prompt playback). A valid value is a boolean (see Section 4.6.1). A value of true indicates that bargein is permitted and prompt playback is stopped. A value of false indicates that bargein is not permitted: user input does not terminate prompt playback. The attribute is optional. The default value is true.

The <prompt> element has the following child elements (any order, multiple occurrences of each element permitted):

<media>: media resource (see Section 4.3.1.1.1) to play. The element is optional.

<variable>: specifies a variable media announcement (see Section 4.3.1.1.2) to play. The element is optional.

<dtmf>: generates one or more DTMF tones (see Section 4.3.1.1.3) to play. The element is optional.

It is an error if no child element is specified.

Prompt playing has the following execution model upon initialization:

1. Prompt playback is initiated playing each <media>, <variable> and <dtmf> in document order.

2. If an error (including fetching errors) occurs during execution, then playback terminates and the error is reported to the dialog container. The <promptinfo> termmode attribute is set to stopped (see Section 4.3.2.1) and any additional information is set.

3. If the dialog container signals a bargein event and the value of the bargein attribute is true, then prompt playback is terminated and the <promptinfo> termmode attribute is set to bargein (see Section 4.3.2.1) and any additional information is set.

4. If prompt playback is stopped by the dialog container, then <promptinfo> termmode attribute is set to stopped (see Section 4.3.2.1).

5. If prompt playback completes successfully, <promptinfo> termmode attribute set to completed (see Section 4.3.2.1) and any additional information is set. This completion status is of this element reported to the dialog.

4.3.1.1.1. <media>

The <media> element specifies a media resource to play.

A <media> element has the following attributes:

src: specifies the location of the media resource. A valid value is a URI (see Section 4.6.9). The attribute is mandatory.

type: specifies the type of the media resource indicated in the 'src' attribute. The MS MAY use the value to assist the remote

source in selecting the appropriate resource type (e.g. with HTTP 'accept' header) and to determine how the document is to be processed. The value may include additional parameters for guiding playback; for example, [RFC4281] defines a 'codec' parameter for 'bucket' media types like video/3gpp. A valid value

is a MIME type (see Section 4.6.10). The attribute is optional. There is no default value.

fetchtimeout: the maximum interval to wait when fetching a media resource. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 30s.

soundLevel: playback soundLevel (volume) for the media resource. A valid value is a percentage (see Section 4.6.4). The value indicates increase or decrease relative to the original recorded volume of the media. A value of 100% (the default) plays the media at its recorded volume, a value of 200% will play the media twice recorded volume, 50% at half its recorded volume, a value of 0% will play the media silently, and so on. See 'soundLevel' in SMIL ([W3C.REC-SMIL2-20051213]) for further information. The attribute is optional. The default value is 100%.

clipBegin: offset from start of media resource to begin playing. A valid value is a Time Designation (see Section 4.6.7). The offset is measured in normal media playback time from the beginning of the media resource. If the clipBegin offset is after the end of media, no media is played. See 'clipBegin' in SMIL ([W3C.REC-SMIL2-20051213]) for further information. The attribute is optional. The default value is 0s.

The <media> element has no children.

It is an error if the media resource cannot be fetched (e.g. fetch timeout occurs) or played (e.g. unsupported format).

4.3.1.1.2. <variable>

The <variable> element specifies variable announcements using predefined media resources. Each variable has at least a type (e.g. date) and a value (e.g. 2008-02-25). The value is rendered according

to the variable type (e.g. 25th February 2008) as well as other defined attributes. The precise mechanism for generating variable announcements (including the location of associated media resources) is implementation specific.

A <variable> element has the following attributes:

value: specifies the string to be rendered. A valid value is a string (see Section 4.6.6). The attribute is mandatory.

type: specifies the type to use for rendering. A valid value is a string (see Section 4.6.6). The attribute is mandatory.

format: specifies format information to use in conjunction with the type for the rendering. A valid value is a string (see Section 4.6.6). The attribute is optional. There is no default value.

gender: specifies the gender to use when rendering the variable. Valid values are "male" or "female". The attribute is optional. There is no default value.

xml:lang: specifies the language to use when rendering the variable. A valid value is a language identifier (see Section 4.6.11). The attribute is optional. There is no default value.

The <variable> element has no children.

This package is agnostic to which <variable> values, types and formats are supported by an implementation. However it is RECOMMENDED that an implementation support the following type/format combinations:

type=date Supported formats: "mdy" (month day year), "ymd" (year month day), "dym" (day month year), "dm" (day month)

type=time Supported formats: "t12" (12 hour format with am/pm), "t24" (24 hour format)

type=digits Supported formats: "gen" (XXX), "ndn" (XXX), "crn" (cardinal), "ord" (ordinal)

[Editors Note: IVR-202. What do 'gen' and 'ndn' format mean for digits? Terms defined in MSCML.]

This specification is agnostic to the type and codec of media resources into which variable are rendered as well as the rendering mechanism itself. For example, an MS implementation supporting audio rendering may map the <variable> into one or more audio media resources.

It is an error if a <variable> element cannot be rendered successfully on the MS implementation.

Depending on the specific implementation of the <variable> rendering on the MS, execution of this element may be seen as conversion of a <variable> into a list of <media> elements. For example,


```
<variable value="2008-02-25" type="date" format="dmy"
xml:lang="en" gender="male"/>
```

could be transformed into audio saying "twenty-fifth of February two thousand and eight" using a list of <media> resources:

```
<media src="nfs://voicebase/en/male/25th.wav"/>
<media src="nfs://voicebase/en/male/of.wav"/>
<media src="nfs://voicebase/en/male/february.wav"/>
<media src="nfs://voicebase/en/male/2000.wav"/>
<media src="nfs://voicebase/en/male/and.wav"/>
<media src="nfs://voicebase/en/male/8.wav"/>
```

4.3.1.1.3. <dtmf>

The <dtmf> element specifies a sequence of DTMF tones for output.

DTMF tones could be generated using <media> resources where the output is transported as RTP audio packets. However, <media> resources are not sufficient for cases where DTMF tones are to be transported as DTMF RTP ([RFC4733]) or in event packages.

A <dtmf> element has the following attributes:

digits: specifies the DTMF sequence to output. A valid value is a DTMF string (see Section 4.6.3). The attribute is mandatory.

level: used to define the power level for which the DTMF tones will be generated. Values are expressed in dBm0. A valid value is an integer in the range of 0 to -96 (dBm0). Larger negative values express lower power levels. Note that values lower than -55 dBm0 will be rejected by most receivers (TR-TSY-000181, ITU-T Q.24A). The attribute is optional. The default value is -6 (dBm0).

duration: specifies the duration for which each DTMF tone is generated. A valid value is a time designation (see Section 4.6.7). Implementations may round the value if they only support discrete durations. The attribute is optional. The default value is 100ms.

interval: specifies the duration of a silence interval following each generated DTMF tone. A valid value is a time designation (see Section 4.6.7). Implementations may round the value if they only support discrete durations. The attribute is optional. The default value is 100ms.

The <dtmf> element has no children.

It is an error if a <dtmf> element cannot be processed successfully.

4.3.1.2. <control>

The <control> element defines how DTMF input is mapped to runtime controls, including prompt playback controls.

DTMF input matching these controls MUST NOT cause prompt playback to interrupted (i.e. no prompt bargein), but causes the appropriate operation to be applied; for examples, speeding up prompt playback.

DTMF input matching these controls has priority over <collect> input for the duration of prompt playback. If incoming DTMF matches a specified runtime control, then the DTMF is not available to the <collect> operation, including its digit buffer. Once prompt

playback is complete, runtime controls are no longer active.

The <control> element has the following attributes:

gotostartkey: maps a DTMF key to skip directly to the start of the prompt. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

gotoendkey: maps a DTMF key to skip directly to the end of the prompt. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

skipinterval: indicates how far a MS should skip backwards or forwards through prompt playback when the rewind (rwkey) of fast forward key (ffkey) is pressed. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 6s.

ffkey: maps a DTMF key to a fast forward operation equal to the value of 'skipinterval'. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

rwkey: maps a DTMF key to a rewind operation equal to the value of 'skipinterval'. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

pauseinterval: indicates how long a MS should pause prompt playback when the pausekey is pressed. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 10s.

pausekey: maps a DTMF key to a pause operation equal to the value of 'pauseinterval'. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

resumekey: maps a DTMF key to a resume operation. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

volumeinterval: indicates the increase or decrease in playback volume (relative to the current volume) when the volupkey or voldnkey is pressed. A valid value is a percentage (see Section 4.6.8). The attribute is optional. The default value is 10%.

volupkey: maps a DTMF key to a volume increase operation equal to the value of 'volumeinterval'. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

voldnkey: maps a DTMF key to a volume decrease operation equal to the value of 'volumeinterval'. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

speedinterval: indicates the increase or decrease in playback speed (relative to the current speed) when the speedupkey or speeddnkey is pressed. A valid value is a percentage (see Section 4.6.8). The attribute is optional. The default value is 10%.

speedupkey: maps a DTMF key to a speed increase operation equal to the value of the speedinterval attribute. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

speeddnkey: maps a DTMF key to a speed decrease operation equal to the value of the speedinterval attribute. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

external: allows one or more DTMF keys to be declared as external controls (for example: video camera controls); the MS can send notifications when a matching key is activated using <dtmfnotify> (Section 4.2.5.2). A valid value is a DTMF String (see Section 4.6.3). The attribute is optional. There is no default value.

It is an error if any control key is specified with the same value

except that the pausekey and resumekey may have the same value.

Runtime control has the following execution model upon initialization:

1. If an error occurs during execution, then runtime control terminates and the error is reported to the dialog container. Controls executed successfully before the error MAY be reported in <controlinfo> (see Section 4.3.2.2).

2. Runtime controls are active only during prompt playback. If DTMF input matches any specified keys (for example the ffkey), then the appropriate operation is applied immediately. If a seek operation (ffkey, rwkey) attempts to go beyond the beginning or end of the prompt queue, then it is automatically truncated to the prompt beginning or end respectively. If the pause operation attempts to pause output when it is already paused, then the operation is ignored. If the resume operation attempts to resume when the prompts are not paused, then the operation is ignored. If a volume operations attempts to go beyond the minimum or maximum volume supported by the platform, then the operation is ignored.

3. If DTMF control subscription has been specified for the dialog, then each DTMF match of a control operation is reported in a <dtmfnotify> notification evemt (Section 4.2.5.2).

4. When the dialog exits, all control matches are reported in a <controlinfo> element (Section 4.3.2.2).

4.3.1.3. <collect>

The <collect> element defines how DTMF input is collected.

The <collect> element has the following attributes:

cleardigitbuffer: indicates whether the digit buffer is to be cleared. A valid value is a boolean (see Section 4.6.1). A value of true indicates that the digit buffer is to be cleared. A value of false indicates that the digit buffer is not to be cleared. The attribute is optional. The default value is true.

timeout: indicates the maximum time to wait for user input to begin. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 5s.

interdigittimeout: indicates inter-digit timeout value to use when recognizing DTMF input. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 2s.

termtimeout: indicates the terminating timeout value to use when recognizing DTMF input. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 0s.

escapekey: specifies a DTMF key that indicates the DTMF collection is to be re-initiated. A valid value is a DTMF Character (see Section 4.6.2). The attribute is optional. There is no default value.

termchar: specifies a DTMF character for terminating DTMF input collection using the internal grammar. A valid value is a DTMF character (see Section 4.6.2). To disable termination by a conventional DTMF character, set the parameter to an unconventional character like 'A'. The attribute is optional. The default value is '#'.

maxdigits: The maximum number of digits to collect using an internal digits (0-9 only) grammar. A valid value is a positive integer (see Section 4.6.5). The attribute is optional. The default value is 5.

The <collect> element has the following child elements:

<grammar>: indicates a custom grammar format (see Section 4.3.1.3.1). The element is optional.

The custom grammar takes priority over the internal grammar. If a <grammar> element is specified, the MS MUST use it for DTMF collection.

DTMF collection has the following execution model upon initialization:

1. The DTMF collection buffer MUST NOT receive DTMF input matching <control> operations (see Section 4.3.1.2).

2. If an error occurs during execution, then collection terminates and the error is reported to the dialog container. DTMF collected before the error MAY be reported in <collectinfo> (see Section 4.3.2.3).

3. The digit buffer is cleared if the value of the cleardigitbuffer attribute is true.

4. A timer with the duration of the value of the timeout attribute is activated. If the timer expires before DTMF input collection begins, then collection execution terminates, the <collectinfo> (see Section 4.3.2.3) has the termmode attribute set to noinput and the execution status reported to the dialog container.

5. If DTMF collect input matches the value of the escapekey attribute, then the timer is canceled and DTMF collection is re-initialized.

6. Other DTMF collect input is matched to the grammar. Valid DTMF patterns are either a simple digit string where the maximum length is determined by the maxdigits attribute and may be terminated by the character in the termchar attribute; or a custom DTMF grammar specified with the <grammar> element. The attributes interdigittimeout and termtimeout control interdigit timeout and the terminating timeout respectively.

7. If the collect input completely matches the grammar, the timer is canceled, collection execution terminates and the execution status is reported to the dialog container with <collectinfo> (see Section 4.3.2.3) where the termmode attribute set to match.

8. If the collect input does not match the grammar, the timer is canceled, collection execution terminates and execution status is reported to the dialog container with a <collectinfo> (see Section 4.3.2.3) where the termmode attribute set to nomatch.

4.3.1.3.1. <grammar>

The <grammar> element allows a custom grammar, inline or external, to be specified. Custom grammars permit the full range of DTMF characters including '*' and '#' to be specified for DTMF pattern matching.

The <grammar> element has the following attributes:

src: specifies the location of an external grammar document. A valid value is a URI (see Section 4.6.9). The attribute is optional. There is no default value.

type: identifies the preferred type of the grammar document identified by the src attribute. The MS MAY use the value to assist the remote source in selecting the appropriate resource type (e.g. with HTTP 'accept' header) and to determine how the

document is processed. A valid value is a MIME type (see Section 4.6.10). The attribute is optional. There is no default value.

fetchtimeout: the maximum interval to wait when fetching a grammar resource. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 30s.

The <grammar> element allows inline grammars to be specified. XML grammar formats MUST use a namespace other than the one used in this specification. Non-XML grammar formats MAY use a CDATA section.

The MS MUST support the [SRGS] XML grammar format ("application/srgs+xml") and MS MAY support KPML ([RFC4730]) or other grammar formats.

It is an error if a grammar format is specified which is not supported by the MS.

For example, the following fragment shows DTMF collection with an inline SRGS grammar:

```
   <collect cleardigitbuffer="false" timeout="20s"
           interdigittimeout="1s">
           <grammar>
                   <grammar xmlns="http://www.w3.org/2001/06/grammar"
                                   version="1.0" mode="dtmf">
                     <rule id="digit">
                      <one-of>
                           <item>0</item>
                           <item>1</item>
                           <item>2</item>
                           <item>3</item>
                           <item>4</item>
                           <item>5</item>
                           <item>6</item>
                           <item>7</item>
                           <item>8</item>
                           <item>9</item>
                      </one-of>
                   </rule>

                   <rule id="pin" scope="public">
                    <one-of>
                           <item>
                            <item repeat="4">
                             <ruleref uri="#digit"/>
```

```
                                    </item>#</item>
                                    <item>* 9</item>
                     </one-of>
                    </rule>

          </grammar>
            </grammar>
           </collect>
```

The same grammar could also be referenced externally (and take advantage of HTTP caching):

```
<collect cleardigitbuffer="false" timeout="20s">
   <grammar type=""application/srgs+xml"
            src="http://example.org/pin.grxml"/>
</collect>
```

4.3.1.4. <record>

The <record> element defines how media input is recorded.

The <record> element has the following attributes:

timeout: indicates the time to wait for user input to begin. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 5s.

type: specifies the type of the recording format. The type value may include additional parameters for guiding recording; for example, [RFC4281] defines a 'codec' parameter for 'bucket' media types like video/3gpp. A valid value is a MIME type (see Section 4.6.10). The attribute is optional. There is no default value (recording format is MS-specific).

dest: specifies the location where recorded data is to be stored. The MS uploads the recorded data to this location during or after the recording operation. A valid value is a URI (see Section 4.6.9). The attribute is optional. If not specified, the MS MUST use a local recording location (reported in <recordinfo> Section 4.3.2.4); this recording is available until the connection or conference associated with the dialog terminates.

vadinitial: Control whether voice activity detection can be used to initiate the recording operation. A valid value is a boolean (see Section 4.6.1). A value of true indicates recording may be initiated using voice activity detection. A value of false indicates that recording must not be initiated using voice activity detection. The attribute is optional. The default value is true.

vadfinal: Control whether voice activity detection can be used to terminate the recording operation. A valid value is a boolean (see Section 4.6.1). A value of true indicates recording may be terminated using voice activity detection. A value of false indicates that recording must not be terminated using voice activity detection. The attribute is optional. The default value is true.

dtmfterm: Indicates whether the recording operation is terminated by DTMF input. A valid value is a boolean (see Section 4.6.1). A value of true indicates that recording is terminated by DTMF input. A value of false indicates that recording is not terminated by DTMF input. The attribute is optional. The default value is true.

maxtime: indicates The maximum duration of the recording. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 15s.

beep: indicates whether a 'beep' should be played immediately prior to initiation of the recording operation. A valid value is a boolean (see Section 4.6.1). The attribute is optional. The default value is false.

finalsilence: indicates the interval of silence that indicates end of speech. This interval is not part of the recording itself. This parameter is ignored if the vadfinal attribute has the value false. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 5s.

It is an error if a dest attribute is specified where the MS does not understand the URI protocol, cannot locate the URI, or cannot send recorded data successfully to the location.

The <record> element has no child elements.

Recording has the following execution model upon initialization:

1. If an error occurs during execution, then record execution terminates and the error is reported to the dialog container. Data recorded before the error MAY be reported in <recordinfo> (see Section 4.3.2.4).

2. If DTMF input (not matching a <control> operation) is received during prompt playback and the prompt bargein attribute is set to true, then record execution is activated. Otherwise, it is activated after the completion of prompt playback.

3. If a beep attribute with the value of true is specified, then a beep tone is played.

4. A timer with the duration of the value of the timeout attribute is activated. If the timer expires before the recording operation begins, then recording execution terminates and the status is reported to dialog container with <recordinfo> (see Section 4.3.2.4) where the termmode attribute is set to noinput.

5. Initiation of the recording operation depends on the value of the vadinitial attribute. If vadinitial has the value false, then the recording operation is initiated immediately. Otherwise, the recording operations is initiated when voice activity is detected.

6. When the recording operation is initiated, a timer is started for the value of the maxtime attribute (maximum duration of the recording). If the timer expires before the recording operation is complete, then recording execution terminates and the

dialogexit result contains a <recordinfo> (see Section 4.3.2.4) with the termmode attribute set to maxtime.

7. During the record operation user media input is recording in the

format specified by the value of the type attribute. If the dest attribute is specified, then recorded input is sent to that location. Otherwise, MS uses an internal location.

8. If the dtmfterm attribute has the value true and DTMF input is detected during the record operation, then the recording terminates and and status is reported to the dialog container with a <recordinfo> (see Section 4.3.2.4) where the termmode attribute is set to dtmf.

9. If vadfinal attribute has the value true, then the recording operation is terminated when a period of silence, with the duration specified by the value of the finalsilence attribute, is detected. This period of silence is not part of the final recording. The status is reported to the dialog container with a <recordinfo> (see Section 4.3.2.4) where the termmode attribute is set to finalsilence.

4.3.2. Exit Information

When the dialog exits, information about the specified operations is reported in a <dialogexit> notification event (Section 4.2.5.1).

4.3.2.1. <promptinfo>

The <promptinfo> element reports the information about prompt execution. It has the following attributes:

duration: indicates the duration of prompt playback in milliseconds. A valid value is a non-negative integer (see Section 4.6.4). The attribute is optional. There is no default value.

termmode: indicates how playback was terminated. Valid values are: 'stopped', 'completed' or 'bargein'. The attribute is mandatory.

The <promptinfo> element has no child elements.

4.3.2.2. <controlinfo>

The <controlinfo> element reports information about control execution.

The <controlinfo> element has no attributes and has 0 or more <controlmatch> child elements each describing an individual runtime

control match.

4.3.2.2.1. <controlmatch>

The <controlmatch> element has the following attributes:

dtmf: DTMF input triggering the runtime control. A valid value is a DTMF string (see Section 4.6.3) with no space between characters. The attribute is mandatory.

timestamp: indicates the time (on the MS) at which the control was triggered. A valid value is an dateTime expression (Section 4.6.12). The attribute is mandatory.

The <controlmatch> element has no child elements.

4.3.2.3. <collectinfo>

The <collectinfo> element reports the information about collect execution.

The <collectinfo> element has the following attributes:

dtmf: DTMF input collected from the user. A valid value is a DTMF string (see Section 4.6.3) with no space between characters. The attribute is optional. There is no default value.

termmode: indicates how collection was terminated. Valid values are: 'stopped', 'match', 'noinput' or 'nomatch'. The attribute is mandatory.

The <collectinfo> element has no child elements.

4.3.2.4. <recordinfo>

The <recordinfo> element reports the information about record execution.

The <recordinfo> element has the following attributes:

recording: references the location to which media is recorded. A valid value is a URI (see Section 4.6.9). The attribute is optional. There is no default value.

type: indicates the format of the recording. A valid value is a MIME type (see Section 4.6.10). The attribute is optional. There is no default value.

duration: indicates the duration of the recording in milliseconds. A valid value is a non-negative integer (see Section 4.6.4). The attribute is optional. There is no default value.

size: indicates the size of the recording in bytes. A valid value is a non-negative integer (see Section 4.6.4). The attribute is optional. There is no default value.

termmode: indicates how recording was terminated. Valid values are: 'stopped', 'noinput', 'dtmf', 'maxtime', and 'finalsilence'. The attribute is mandatory.

The <recordinfo> element has no child elements.

4.4. Audit Elements

The audit elements defined in this section allow the MS to be audited for package capabilities as well as dialogs managed by the package. Auditing is particularly important for two use cases. First, it enables discovery of package capabilities supported on an MS before an AS starts a dialog on connection or conference. The AS may then use this informaton to create request elements using supported capabilities and, in the case of codecs, to negotiate an appropriate SDP for a user agent's connection. Second, auditing enables discovery of the existence and status of dialogs currently managed by the package on the MS. This allows one AS to take over management of the dialogs when the AS which initiated the dialogs fails or is no longer available.

4.4.1. <audit>

The <audit> request element is sent to the MS to request information about the capabilities of, and dialogs currently managed with, this control package. Capabilities include supported dialog types, grammar formats, record and media types as well as codecs. Dialog information includes the status of managed dialogs as well as codecs.

The <audit> element has the following attributes:

capabilities: indicates whether package capabilities are to be

audited. A valid value is a boolean (see Section 4.6.1). A value of true indicates that capability information is to be reported. A value of false indicates that capability information is not to be reported. The attribute is optional. The default value is true.

dialogs: indicates whether dialogs currently managed by the package are to be audited. A valid value is a boolean (see Section 4.6.1). A value of true indicates that dialog information is to be reported. A value of false indicates that dialog information is not to be reported. The attribute is optional. The default value is true.

dialogid: string identifying a specific dialog to audit. It is an error if the dialogid attribute is specified and the dialog identifier is not valid. The attribute is optional. There is no default value.

If the dialogs attribute has the value true and dialogid attribute is specified, then only audit information about the specified dialog is reported. If the dialogs attribute has the value false, then no dialog audit information is reported even if a dialogid attribute is specified.

The <audit> element has no child elements.

When the MS receives a <audit> request, it MUST reply with a <auditresponse> element (Section 4.4.2). If the request is successful, <auditresponse> contain (depending on atrribute values) a <capabilities> element (Section 4.4.2.2) reporting package capabilities and a <dialogs> element (Section 4.4.2.3) reporting managed dialog information.

For example, a request to audit capabilities and dialogs managed by the package:

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

In this example, only capabilities are to be audited:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
  <audit dialogs="false"/>
</mscivr>
```

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

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
  <audit capabilities="false" dialogid="d4"/>
</mscivr>
```


4.4.2. <auditresponse>

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

The <auditresponse> element has the following attributes:

status: numeric code indicating the audit response status. The attribute is mandatory. Valid values are defined in Section 4.5.

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

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

<capabilities> element (Section 4.4.2.2) describing capabilities of the package. The element is optional.

<dialogs> element (Section 4.4.2.3) describing information about managed dialogs. The element is optional.

For example, a successful response to a <audit> request requesting capabilities and dialogs information:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
```

```
    <auditresponse status="200">
     <capabilities>
      <dialogtypes>
       <mimetype>application/msc-ivr+xml</mimetype>
      </dialogtypes>
      <grammartypes>
       <mimetype>application/srgs+xml</mimetype>
      </grammartypes>
      <recordtypes>
       <mimetype>application/x-wav</mimetype>
       <mimetype>video/3gpp</mimetype>
      </recordtypes>
      <mediatypes>
       <mimetype>application/x-wav</mimetype>
       <mimetype>video/3gpp</mimetype>
      </mediatypes>
      <codecs>
       <codec>
        <subtype>H.263</subtype>
       </codec>
       <codec>
        <subtype>H.264</subtype>
```



```
       </codec>
       <codec>
        <subtype>PCMU</subtype>
       </codec>
       <codec>
        <subtype>PCMA</subtype>
       </codec>
       <codec>
        <subtype>telephone-event</subtype>
       </codec>
      </codecs>
     </capabilities>
     <dialogs>
      <dialogaudit dialogid="4532" state="preparing"/>
      <dialogaudit dialogid="4599" state="prepared"/>
      <dialogaudit dialogid="1234" state="started" conferenceid="conf1">
       <codecs>
        <codec>
         <subtype>PCMA</subtype>
```

```
        </codec>
        <codec>
         <subtype>telephone-event</subtype>
        </codec>
       </codecs>
      </dialogaudit>
     </dialogs>
    </auditresponse>
   </mscivr>
```

4.4.2.1. <codecs>

The <codecs> provides audit information about codecs.

The <codecs> element has no attributes.

The <codecs> element has the following sequence of child elements (0 or more occurences):

<codec>: audit information for a codec (Section 4.4.2.1.1). The element is optional.

For example, a fragment describing two codecs:


```
   <codecs>
     <codec>
      <subtype>PCMA</subtype>
     </codec>
     <codec>
      <subtype>telephone-event</subtype>
     </codec>
   </codecs>
```

4.4.2.1.1. <codec>

The <codec> element describes a codec on the MS. The element is

defined in the XCON conference information data model ([I-D.ietf-xcon-common-data-model]).

Note that additional information about the codec can be provided through schema extensibility (see Section 5).

[Editors Note: IVR-203. Do we need to define additional information? specific (rate, speed, etc) or generic using <params>?]

For example, a fragment with a <codec> element describing the H.263 codec:

```
<codec>
 <subtype>H.263</subtype>
</codec>
```

4.4.2.2. <capabilities>

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

The <capabilities> element has no attributes.

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

<dialogtypes>: element (Section 4.4.2.2.1) describing supported dialog types. The element is mandatory.

<grammartypes>: element (Section 4.4.2.2.2) describing supported <grammar> (Section 4.3.1.3.1) format types. The element is mandatory.

<recordtypes>: element (Section 4.4.2.2.3) describing supported <record> (Section 4.3.1.4) format types. The element is mandatory.

<mediatypes>: element (Section 4.4.2.2.4) describing supported

<media> (Section 4.3.1.1.1) format types. The element is mandatory.

<codecs>: element (Section 4.4.2.1) describing codecs available to the package. The element is mandatory.

For example, a fragment describing capabilities:

```
  <capabilities>
   <dialogtypes>
    <mimetype>application/msc-ivr+xml</mimetype>
    <mimetype>application/voicexml+xml</mimetype>
   </dialogtypes>
   <grammartypes>
    <mimetype>application/srgs+xml</mimetype>
   </grammartypes>
   <recordtypes>
    <mimetype>application/x-wav</mimetype>
    <mimetype>video/3gpp</mimetype>
   </recordtypes>
   <mediatypes>
    <mimetype>application/x-wav</mimetype>
    <mimetype>video/3gpp</mimetype>
   </mediatypes>
   <codecs>
    <codec>
     <subtype>H.263</subtype>
    </codec>
    <codec>
     <subtype>H.264</subtype>
    </codec>
    <codec>
     <subtype>PCMU</subtype>
    </codec>
    <codec>
     <subtype>PCMA</subtype>
    </codec>
    <codec>
     <subtype>telephone-event</subtype>
    </codec>
   </codecs>
  </capabilities>
```

4.4.2.2.1. <dialogtypes>

The <dialogtypes> element provides information about dialog types supported by the package. The MS MUST include the mandatory dialog type for this package, "application/msc-ivr+xml" (Section 8.3) associated with the IVR dialog type (Section 4.3).

The <dialogtypes> element has no attributes.

The <dialogtypes> element has the following sequence of child elements (1 or more occurences):

<mimetype>: element whose content model describes a mime type (Section 4.6.10). The element is optional.

4.4.2.2.2. <grammartypes>

The <grammartypes> element provides information about <grammar> format types supported by the package. The MS MUST include the mandatory SRGS format type, "application/srgs+xml" (Section 4.3.1.3.1).

The <grammartypes> element has no attributes.

The <grammartypes> element has the following sequence of child elements (1 or more occurences):

<mimetype>: element whose content model describes a mime type (Section 4.6.10). The element is optional.

4.4.2.2.3. <recordtypes>

The <recordtypes> element provides information about <record> format types supported by the package (Section 4.3.1.4).

The <recordtypes> element has no attributes.

The <recordtypes> element has the following sequence of child elements (0 or more occurences):

<mimetype>: element whose content model describes a mime type (Section 4.6.10). The element is optional.

4.4.2.2.4. <mediatypes>

The <mediatypes> element provides information about <media> format types supported by the package (Section 4.3.1.1.1).

The <mediatypes> element has no attributes.

The <mediatypes> element has the following sequence of child elements (0 or more occurences):

<mimetype>: element whose content model describes a mime type (Section 4.6.10). The element is optional.

4.4.2.3. <dialogs>

The <dialogs> element provides audit information about dialogs.

The <dialogs> element has no attributes.

The <dialogs> element has the following sequence of child elements (0 or more occurences):

<dialogaudit>: audit information for a dialog (Section 4.4.2.3.1). The element is optional.

4.4.2.3.1. <dialogaudit>

The <dialogaudit> element has the following attributes:

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

state: string indicating the state of the dialog. Valid values are: preparing, prepared, starting, started. The attribute is mandatory.

connectionid: string identifying the SIP dialog connection associated with the dialog (see Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework]). The attribute is optional. There is no default value.

conferenceid: string identifying the conference associated with the dialog (see Section 17.1 of [I-D.ietf-mediactrl-sip-control-framework]). The attribute is optional. There is no default value.

The <dialogaudit> element has the following child element:

<codecs> element describing codecs used in the dialog. See Section 4.4.2.1. The element is optional.

For example, a fragment describing a started dialog which is using PCMU and telephony-event codecs:


```
<dialogaudit dialogid="1234" state="started">
 <codecs>
  <codec>
   <subtype>PCMU</subtype>
  </codec>
  <codec>
   <subtype>telephone-event</subtype>
  </codec>
 </codecs>
</dialogaudit>
```

4.5. Response Status Codes

The following status codes for dialog management (Section 4.2.4) and audit (Section 4.4.2) responses are defined:

code	description
200	OK
401	dialogid already exists
402	dialogid does not exist
403	connectionid does not exist
404	conferenceid does not exist
405	Unknown or unsupported element
406	Element required
407	Unknown or unsupported attribute
408	Attribute required
409	dialog type not supported
410	Retrieving resource failed
411	Invalid attribute value
412	subscription not supported

413	invalid stream configuration
414	dialog execution cancelled
499	other error

Table 1: status codes

The MS MAY define other status codes.

4.6. Type Definitions

This section defines types referenced in attribute definitions.

4.6.1. Boolean

The value space of boolean is the set {true, false}.

4.6.2. DTMFChar

A DTMF character. The value space is the set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, #, *, A, B, C, D}.

4.6.3. DTMFString

A String composed of one or more DTMFChars.

4.6.4. Non-Negative Integer

The value space of non-negative integer is the infinite set {0,1,2,...}.

4.6.5. Positive Integer

The value space of positive integer is the infinite set {1,2,...}.

4.6.6. String

A string in the character encoding associated with the XML element.

4.6.7. Time Designation

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

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

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

4.6.8. Percentage

A percentage consists of a Positive Integer followed by "%".

Examples include: "100%", "500%" and "10%".

4.6.9. URI

Uniform Resource Indicator as defined in [RFC3986].

4.6.10. mimetype

A string formated as a IANA mimetype.

4.6.11. Language Identifier

A language identifier labels information content as being of a particular human language variant. Following the XML specification for language identification [XML], a legal language identifier is identified by a RFC4646 ([RFC4646]) and RFC4647 ([RFC4647]) code where the language code is required and a country code or other subtag identifier is optional.

4.6.12. DateTime

A string formated according to the XML schema definition of a dateTime type ([XMLSchema:Part2]).

5. Formal Syntax

This section defines the XML schema for IVR Control Package.

The schema defines datatypes, attributes, dialog management and IVR dialog elements in the urn:ietf:params:xml:ns:msc-ivr namespace. In most elements the order of child elements is significant. The schema

is extensible: elements allow attributes and child elements from other namespaces. Elements from outside this package's namespace can occur after elements defined in this package.

The schema is dependent upon the schema (framework.xsd) defined in Section 17.1 of the Control Framework [I-D.ietf-mediactrl-sip-control-framework]. It is also dependent upon the W3C (xml.xsd) schema for definitions of XML attributes (e.g. xml:base).

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

 <xsd:annotation>
  <xsd:documentation>
   IETF MediaCtrl IVR 1.0 (20080610)

   This is the schema of the IETF MediaCtrl IVR control
   package.

   The schema namespace is urn:ietf:params:xml:ns:msc-ivr
  </xsd:documentation>
 </xsd:annotation>


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

 SCHEMA IMPORTS

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

<xsd:import namespace="http://www.w3.org/XML/1998/namespace"
 schemaLocation="xml.xsd">
 <xsd:annotation>
  <xsd:documentation>
```

```
    This import brings in the XML attributes for
    xml:base, xml:lang, etc

    See http://www.w3.org/2001/xml.xsd for latest version
   </xsd:documentation>
  </xsd:annotation>
</xsd:import>

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


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

 Extensible core type

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


<xsd:complexType name="Tcore">
 <xsd:annotation>
  <xsd:documentation>
   This type is extended by other component types to
   allow elements and attributes from other namespaces
   to be added.
  </xsd:documentation>
 </xsd:annotation>
 <xsd:sequence>
  <xsd:any namespace="##other" minOccurs="0"
   maxOccurs="unbounded" processContents="lax" />
 </xsd:sequence>
 <xsd:anyAttribute namespace="##other" processContents="lax" />
</xsd:complexType>


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

```
    TOP LEVEL ELEMENT: mscivr

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

   <xsd:complexType name="mscivrType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:choice>
        <xsd:element ref="dialogprepare" />
        <xsd:element ref="dialogstart" />
        <xsd:element ref="dialogterminate" />
        <xsd:element ref="response" />
        <xsd:element ref="event" />
        <xsd:element ref="audit" />
        <xsd:element ref="auditresponse" />
        <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded" processContents="lax" />
       </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="version" type="version.datatype"
       use="required" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="mscivr" type="mscivrType" />


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

    DIALOG MANAGEMENT TYPES

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

   <!--  dialogprepare -->

   <xsd:complexType name="dialogprepareType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
```

```
      <xsd:sequence>
       <xsd:element ref="dialog" minOccurs="0"
        maxOccurs="1" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
```



```
      </xsd:sequence>
      <xsd:attribute name="src" type="xsd:anyURI" />
      <xsd:attribute name="type" type="mime.datatype"
       default="application/msc-ivr+xml" />
      <xsd:attribute name="fetchtimeout"
       type="timedesignation.datatype" default="30s" />
      <xsd:attribute name="dialogid"
       type="dialogid.datatype" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="dialogprepare" type="dialogprepareType" />

   <!--  dialogstart -->

   <xsd:complexType name="dialogstartType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:element ref="dialog" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="subscribe" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="params" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="stream" minOccurs="0"
        maxOccurs="unbounded" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="src" type="xsd:anyURI" />
      <xsd:attribute name="type" type="mime.datatype" />
      <xsd:attribute name="fetchtimeout"
       type="timedesignation.datatype" default="30s" />
      <xsd:attribute name="dialogid"
```

```
      type="dialogid.datatype" />
     <xsd:attribute name="prepareddialogid"
      type="dialogid.datatype" />
     <xsd:attributeGroup ref="fw:framework-attributes" />
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="dialogstart" type="dialogstartType" />

  <!--  dialogterminate -->
```



```
  <xsd:complexType name="dialogterminateType">
   <xsd:complexContent>
    <xsd:extension base="Tcore">
     <xsd:attribute name="dialogid"
      type="dialogid.datatype" use="required" />
     <xsd:attribute name="immediate"
      type="boolean.datatype" default="false" />
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="dialogterminate" type="dialogterminateType" />

  <!--  response -->

  <xsd:complexType name="responseType">
   <xsd:complexContent>
    <xsd:extension base="Tcore">
     <xsd:attribute name="status" type="status.datatype"
      use="required" />
     <xsd:attribute name="reason" type="xsd:string" />
     <xsd:attribute name="dialogid"
      type="dialogid.datatype" use="required" />
     <xsd:attributeGroup ref="fw:framework-attributes" />
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>

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

```
<!--  event -->

<xsd:complexType name="eventType">
 <xsd:complexContent>
  <xsd:extension base="Tcore">
   <xsd:sequence>
    <xsd:choice>
     <xsd:element ref="dialogexit" minOccurs="0"
      maxOccurs="1" />
     <xsd:element ref="dtmfnotify" minOccurs="0"
      maxOccurs="1" />
     <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:choice>
   </xsd:sequence>
   <xsd:attribute name="dialogid"
    type="dialogid.datatype" use="required" />
  </xsd:extension>
```



```
 </xsd:complexContent>
</xsd:complexType>

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

<!--  dialogexit-->

<xsd:complexType name="dialogexitType">
 <xsd:complexContent>
  <xsd:extension base="Tcore">
   <xsd:sequence>
    <xsd:element ref="promptinfo" minOccurs="0"
     maxOccurs="1" />
    <xsd:element ref="controlinfo" minOccurs="0"
     maxOccurs="1" />
    <xsd:element ref="collectinfo" minOccurs="0"
     maxOccurs="1" />
    <xsd:element ref="recordinfo" minOccurs="0"
     maxOccurs="1" />
    <xsd:element ref="params" minOccurs="0"
     maxOccurs="1" />
    <xsd:any namespace="##other" minOccurs="0"
```

```
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="status"
     type="xsd:positiveInteger" use="required" />
    <xsd:attribute name="reason" type="xsd:string" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:element name="dialogexit" type="dialogexitType" />


 <!--  dtmfnotify-->

 <xsd:complexType name="dtmfnotifyType">
  <xsd:complexContent>
   <xsd:extension base="Tcore">
    <xsd:attribute name="matchmode"
     type="matchmode.datatype" default="all" />
    <xsd:attribute name="dtmf" type="dtmfstring.datatype"
     use="required" />
    <xsd:attribute name="timestamp" type="xsd:dateTime"
     use="required" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```



```
 <xsd:element name="dtmfnotify" type="dtmfnotifyType" />


 <!-- promptinfo -->

 <xsd:complexType name="promptinfoType">
  <xsd:complexContent>
   <xsd:extension base="Tcore">
    <xsd:attribute name="duration"
     type="xsd:nonNegativeInteger" />
    <xsd:attribute name="termmode"
     type="prompt_termmode.datatype" use="required" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

```
   <xsd:element name="promptinfo" type="promptinfoType" />


   <!-- controlinfo -->

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

   <xsd:element name="controlinfo" type="controlinfoType" />


   <!-- controlmatch -->

   <xsd:complexType name="controlmatchType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="dtmf"
       type="dtmfstring.datatype" />
      <xsd:attribute name="timestamp" type="xsd:dateTime" />
     </xsd:extension>
    </xsd:complexContent>
```



```
   </xsd:complexType>

   <xsd:element name="controlmatch" type="controlmatchType" />


   <!--  collectinfo -->

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

```
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="dtmf"
       type="dtmfstring.datatype" />
      <xsd:attribute name="termmode"
       type="collect_termmode.datatype" use="required" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="collectinfo" type="collectinfoType" />


   <!-- recordinfo -->

   <xsd:complexType name="recordinfoType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="recording" type="xsd:anyURI" />
      <xsd:attribute name="type" type="mime.datatype" />
      <xsd:attribute name="duration"
       type="xsd:nonNegativeInteger" />
      <xsd:attribute name="size"
       type="xsd:nonNegativeInteger" />
      <xsd:attribute name="termmode"
       type="record_termmode.datatype" use="required" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="recordinfo" type="recordinfoType" />



   <!--  subscribe -->

   <xsd:complexType name="subscribeType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
```



```
       <xsd:element ref="dtmfsub" minOccurs="0"
```

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

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


 <!--  dtmfsub -->

 <xsd:complexType name="dtmfsubType">
  <xsd:complexContent>
   <xsd:extension base="Tcore">
    <xsd:attribute name="matchmode"
     type="matchmode.datatype" default="all" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:element name="dtmfsub" type="dtmfsubType" />


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

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


 <!--  param -->
 <!--  doesn't extend tCore since its content model is mixed -->
 <xsd:complexType name="paramType" mixed="true">
  <xsd:sequence>
   <xsd:any namespace="##other" minOccurs="0"
```

```
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="type" type="mime.datatype" />
    <xsd:attribute name="valuetype" type="valuetype.datatype"
     default="string" />
    <xsd:anyAttribute namespace="##other" processContents="lax" />
   </xsd:complexType>

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

   <!--  stream -->

   <xsd:complexType name="streamType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="media" type="media.datatype"
       use="required" />
      <xsd:attribute name="label" type="label.datatype" />
      <xsd:attribute name="direction"
       type="direction.datatype" default="sendrecv" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

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

   <!-- dialog -->

   <xsd:complexType name="dialogType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:element ref="prompt" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="control" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="collect" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="record" minOccurs="0"
        maxOccurs="1" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="repeatCount"
       type="xsd:nonNegativeInteger" default="1" />
```

```
      <xsd:attribute name="repeatDur"
       type="timedesignation.datatype" />
```



```
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="dialog" type="dialogType" />


   <!--  prompt  -->

   <xsd:complexType name="promptType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
       <xsd:element ref="media" />
       <xsd:element ref="variable" />
       <xsd:element ref="dtmf" />
       <xsd:any namespace="##other"
        processContents="lax" />
      </xsd:choice>
      <xsd:attribute ref="xml:base" />
      <xsd:attribute name="bargein" type="boolean.datatype"
       default="true" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="prompt" type="promptType" />


   <!--  media -->

   <xsd:complexType name="mediaType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="src" type="xsd:anyURI"
       use="required" />
      <xsd:attribute name="type" type="mime.datatype" />
      <xsd:attribute name="fetchtimeout"
       type="timedesignation.datatype" default="30s" />
```

```
      <xsd:attribute name="soundLevel"
       type="percentage.datatype" default="100%" />
      <xsd:attribute name="clipBegin"
       type="timedesignation.datatype" default="0s" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="media" type="mediaType" />
```



```
   <!--  variable -->

   <xsd:complexType name="variableType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="value" type="xsd:string"
       use="required" />
      <xsd:attribute name="type" type="xsd:string"
       use="required" />
      <xsd:attribute name="format" type="xsd:string" />
      <xsd:attribute name="gender" type="gender.datatype" />
      <xsd:attribute ref="xml:lang" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="variable" type="variableType" />

   <!-- dtmf -->

   <xsd:complexType name="dtmfType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="digits"
       type="dtmfstring.datatype" use="required" />
      <xsd:attribute name="level" type="xsd:integer"
       default="-6" />
      <xsd:attribute name="duration"
       type="timedesignation.datatype" default="100ms" />
      <xsd:attribute name="interval"
       type="timedesignation.datatype" default="100ms" />
     </xsd:extension>
```

```
 </xsd:complexContent>
</xsd:complexType>

<xsd:element name="dtmf" type="dtmfType" />


<!-- control -->

<xsd:complexType name="controlType">
 <xsd:complexContent>
  <xsd:extension base="Tcore">
   <xsd:attribute name="skipinterval"
    type="timedesignation.datatype" default="6s" />
   <xsd:attribute name="ffkey" type="dtmfchar.datatype" />
   <xsd:attribute name="rwkey" type="dtmfchar.datatype" />
   <xsd:attribute name="pauseinterval"
```



```
    type="timedesignation.datatype" default="10s" />
   <xsd:attribute name="pausekey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="resumekey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="volumeinterval"
    type="percentage.datatype" default="10%" />
   <xsd:attribute name="volupkey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="voldnkey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="speedinterval"
    type="percentage.datatype" default="10%" />
   <xsd:attribute name="speedupkey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="speeddnkey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="gotostartkey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="gotoendkey"
    type="dtmfchar.datatype" />
   <xsd:attribute name="external"
    type="dtmfstring.datatype" />
  </xsd:extension>
 </xsd:complexContent>
```

```
   </xsd:complexType>

   <xsd:element name="control" type="controlType" />


   <!-- collect -->

   <xsd:complexType name="collectType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:element ref="grammar" minOccurs="0"
        maxOccurs="1" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="cleardigitbuffer"
       type="boolean.datatype" default="true" />
      <xsd:attribute name="timeout"
       type="timedesignation.datatype" default="5s" />
      <xsd:attribute name="interdigittimeout"
       type="timedesignation.datatype" default="2s" />
      <xsd:attribute name="termtimeout"
```



```
       type="timedesignation.datatype" default="0s" />
      <xsd:attribute name="escapekey"
       type="dtmfchar.datatype" />
      <xsd:attribute name="termchar"
       type="dtmfchar.datatype" default="#" />
      <xsd:attribute name="maxdigits"
       type="xsd:positiveInteger" default="5" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="collect" type="collectType" />

   <!--  grammar -->
   <!--  doesn't extend tCore since its content model is mixed -->
   <xsd:complexType name="grammarType" mixed="true">
    <xsd:sequence>
     <xsd:any namespace="##other" minOccurs="0"
```

```
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="src" type="xsd:anyURI" />
    <xsd:attribute name="type" type="mime.datatype" />
    <xsd:attribute name="fetchtimeout"
     type="timedesignation.datatype" default="30s" />
    <xsd:anyAttribute namespace="##other" processContents="lax" />
   </xsd:complexType>

   <xsd:element name="grammar" type="grammarType" />

   <!-- record -->

   <xsd:complexType name="recordType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="timeout"
       type="timedesignation.datatype" default="5s" />
      <xsd:attribute name="type" type="mime.datatype" />
      <xsd:attribute name="dest" type="xsd:anyURI" />
      <xsd:attribute name="beep" type="boolean.datatype"
       default="false" />
      <xsd:attribute name="vadinitial"
       type="boolean.datatype" default="true" />
      <xsd:attribute name="vadfinal"
       type="boolean.datatype" default="true" />
      <xsd:attribute name="dtmfterm"
       type="boolean.datatype" default="true" />
      <xsd:attribute name="maxtime"
       type="timedesignation.datatype" default="15s" />
```



```
      <xsd:attribute name="finalsilence"
       type="timedesignation.datatype" default="5s" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="record" type="recordType" />

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

```
    AUDIT TYPES

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


   <!-- audit -->

   <xsd:complexType name="auditType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:attribute name="capabilities"
       type="boolean.datatype" default="true" />
      <xsd:attribute name="dialogs" type="boolean.datatype"
       default="true" />
      <xsd:attribute name="dialogid"
       type="dialogid.datatype" />
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

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

   <!-- auditresponse -->

   <xsd:complexType name="auditresponseType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:element ref="capabilities" minOccurs="0"
        maxOccurs="1" />
       <xsd:element ref="dialogs" minOccurs="0"
        maxOccurs="1" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="status" type="status.datatype"
```



```
       use="required" />
      <xsd:attribute name="reason" type="xsd:string" />
     </xsd:extension>
    </xsd:complexContent>
```

```
</xsd:complexType>

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

<!-- codec -->

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

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

<!-- subtype -->

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

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


<!-- codecs -->

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

```
    </xsd:complexContent>
   </xsd:complexType>

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


   <!-- capabilities -->

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

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


   <!-- mimetype -->

   <xsd:element name="mimetype" type="mime.datatype" />


   <!-- dialogtypes -->

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

```
      </xsd:sequence>
```



```
     </xsd:extension>
    </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="dialogtypes" type="dialogtypesType" />

   <!-- grammartypes -->

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

   <xsd:element name="grammartypes" type="grammartypesType" />


   <!-- recordtypes -->

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

   <xsd:element name="recordtypes" type="recordtypesType" />
```

```
   <!-- mediatypes -->

   <xsd:complexType name="mediatypesType">
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
```



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

   <xsd:element name="mediatypes" type="mediatypesType" />


   <!-- dialogs -->

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

   <xsd:element name="dialogs" type="dialogsType" />


   <!-- dialogaudit -->

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

```
    <xsd:complexContent>
     <xsd:extension base="Tcore">
      <xsd:sequence>
       <xsd:element ref="codecs" minOccurs="0"
        maxOccurs="1" />
       <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" processContents="lax" />
      </xsd:sequence>
      <xsd:attribute name="dialogid"
       type="dialogid.datatype" use="required" />
      <xsd:attribute name="state" type="state.datatype"
       use="required" />
      <xsd:attributeGroup ref="fw:framework-attributes" />
     </xsd:extension>
    </xsd:complexContent>
```



```
   </xsd:complexType>

   <xsd:element name="dialogaudit" type="dialogauditType" />


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

    DATATYPES

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


    <xsd:simpleType name="version.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="1.0" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="mime.datatype">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>
    <xsd:simpleType name="dialogid.datatype">
     <xsd:restriction base="xsd:string" />
    </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="valuetype.datatype">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>
    <xsd:simpleType name="gender.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="female" />
      <xsd:enumeration value="male" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="policy.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="allowed" />
      <xsd:enumeration value="disallowed" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="state.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
```



```
      <xsd:enumeration value="preparing" />
      <xsd:enumeration value="prepared" />
      <xsd:enumeration value="starting" />
      <xsd:enumeration value="started" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="status.datatype">
     <xsd:restriction base="xsd:positiveInteger">
      <xsd:pattern value="[0-9][0-9][0-9]" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="media.datatype">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>
    <xsd:simpleType name="label.datatype">
     <xsd:restriction base="xsd:string" />
    </xsd:simpleType>
    <xsd:simpleType name="direction.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
```

```
      <xsd:enumeration value="sendrecv" />
      <xsd:enumeration value="sendonly" />
      <xsd:enumeration value="recvonly" />
      <xsd:enumeration value="inactive" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="timedesignation.datatype">
     <xsd:annotation>
      <xsd:documentation>
       Time designation following Time in CSS2
      </xsd:documentation>
     </xsd:annotation>
     <xsd:restriction base="xsd:string">
      <xsd:pattern value="(\+)?([0-9]*\.)?[0-9]+(ms|s)" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="dtmfchar.datatype">
     <xsd:annotation>
      <xsd:documentation>
       DTMF character [0-9#*A-D]
      </xsd:documentation>
     </xsd:annotation>
     <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9#*A-D]" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="dtmfstring.datatype">
     <xsd:annotation>
      <xsd:documentation>
```



```
       DTMF sequence [0-9#*A-D]
      </xsd:documentation>
     </xsd:annotation>
     <xsd:restriction base="xsd:string">
      <xsd:pattern value="([0-9#*A-D])+" />
     </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="percentage.datatype">
     <xsd:annotation>
      <xsd:documentation>
       whole integer followed by '%'
      </xsd:documentation>
```

```
     </xsd:annotation>
     <xsd:restriction base="xsd:string">
      <xsd:pattern value="([0-9])+%" />
     </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="prompt_termmode.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="completed" />
      <xsd:enumeration value="maxduration" />
      <xsd:enumeration value="bargein" />
      <xsd:enumeration value="stopped" />
     </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="collect_termmode.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="match" />
      <xsd:enumeration value="noinput" />
      <xsd:enumeration value="nomatch" />
      <xsd:enumeration value="stopped" />
     </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="record_termmode.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="noinput" />
      <xsd:enumeration value="dtmf" />
      <xsd:enumeration value="maxtime" />
      <xsd:enumeration value="finalsilence" />
      <xsd:enumeration value="stopped" />
     </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="matchmode.datatype">
     <xsd:restriction base="xsd:NMTOKEN">
```



```
      <xsd:enumeration value="all" />
      <xsd:enumeration value="collect" />
      <xsd:enumeration value="control" />
     </xsd:restriction>
    </xsd:simpleType>
```

```
</xsd:schema>
```

6. Examples

This section provides examples of the IVR Control package.

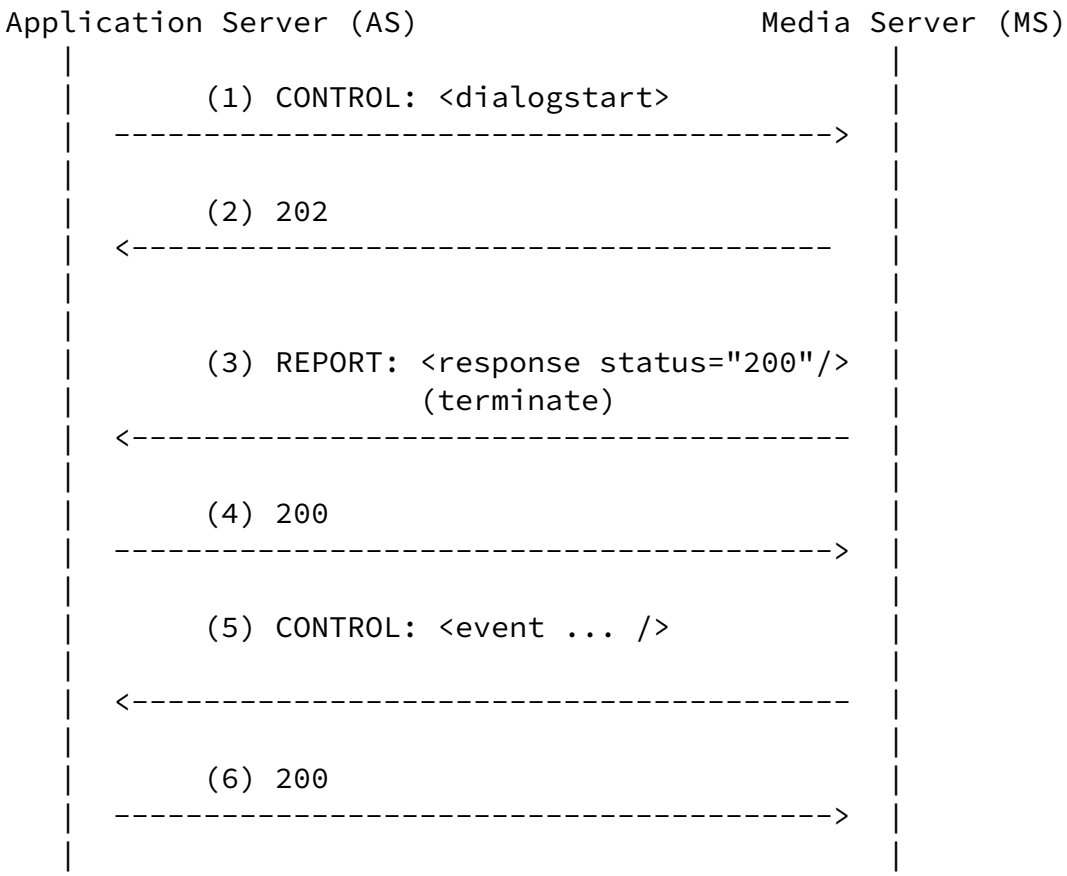
6.1. AS-MS Dialog Interaction Examples

The following example assume a control channel has been established and synced as described in the Media Control Channel Framework ([I-D.ietf-mediactrl-sip-control-framework]).

The XML messages are in angled brackets (with the root <mscivr> omitted); the REPORT status is in round brackets. Other aspects of the protocol are omitted for readability.

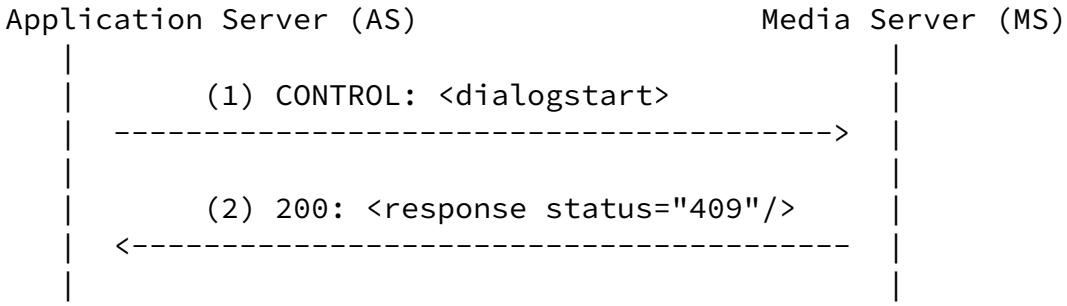
6.1.1. Starting an IVR dialog

An IVR dialog is started successfully, and dialogexit notification <event> is sent from the MS to the AS when the dialog exits normally.



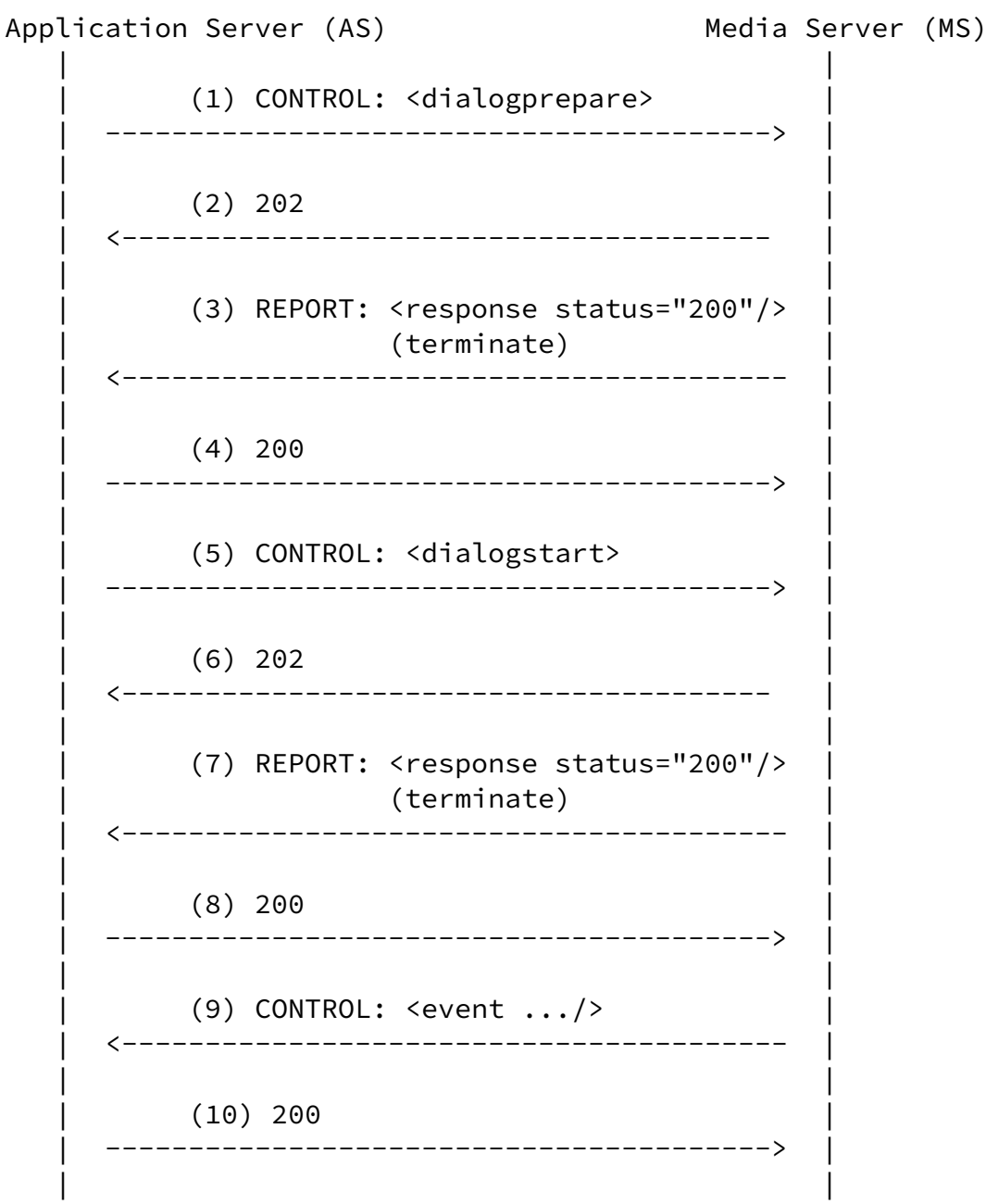
6.1.2. IVR dialog fails to start

An IVR dialog fails to start due to an unknown dialog type. The <sreponse> is reported in a framework 200 message.



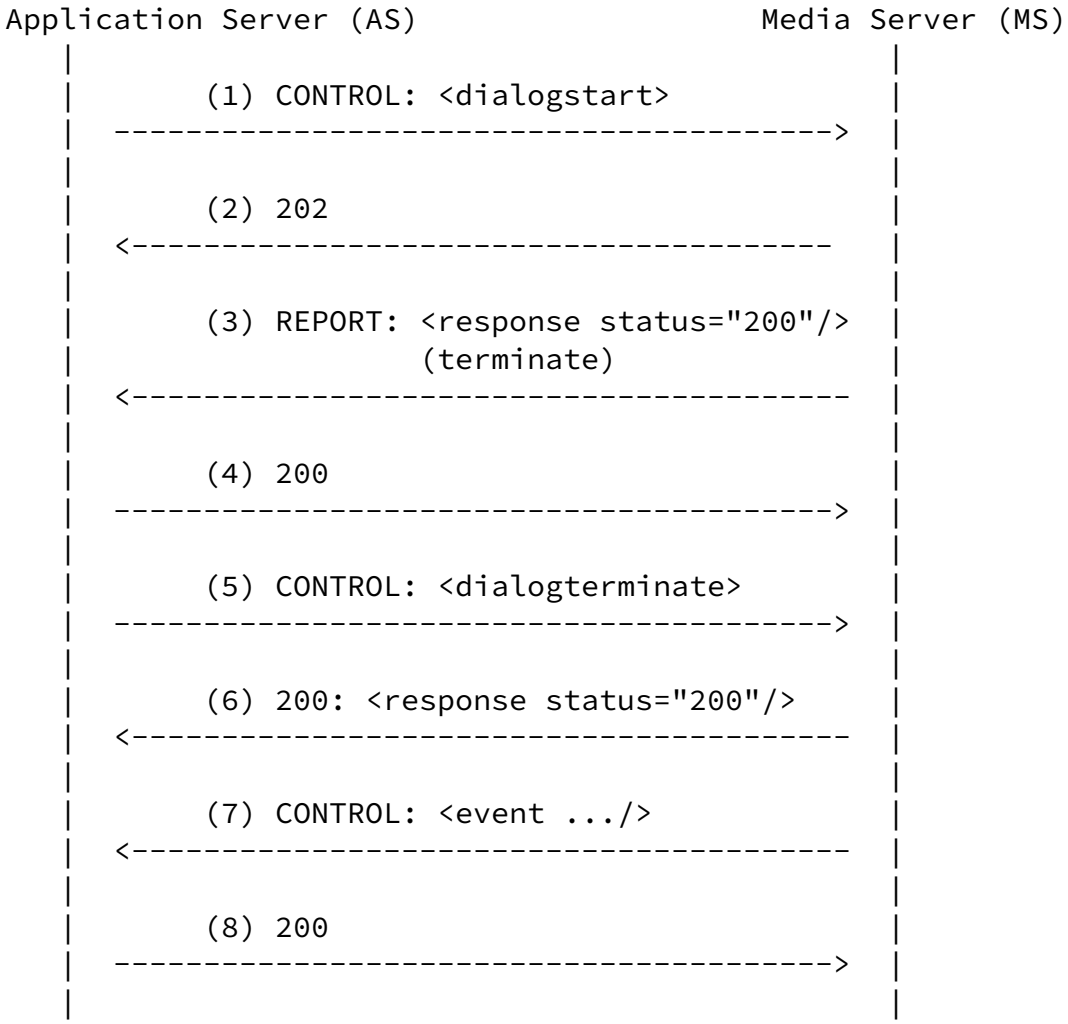
6.1.3. Preparing and starting an IVR dialog

An IVR dialog is prepared and started successfully, and then the dialog exits normally.


Application Server (AS)
Media Server (MS)
(1) CONTROL: <dialogprepare>
(2) 202
(3) REPORT: <response status="200"/>
(terminate)
(4) 200
(5) CONTROL: <dialogstart>
(6) 202
(7) REPORT: <response status="200"/>
(terminate)
(8) 200
(9) CONTROL: <event .../>
(10) 200

6.1.4. Terminating a dialog

An IVR dialog is started successfully, and then terminated by the AS. The dialogexit event is sent by to the AS when the dialog exits.



Note that in (6) the <response> payload to the <dialogterminate/> request is carried on a framework 200 response since it could complete the requested operation before the transaction timeout.

6.2. IVR Dialog Examples

The following examples show how <dialog> is used with <dialogprepare>, <dialogstart> and <event> elements to play prompts, set runtime controls, collect DTMF input and record user input.

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

6.2.1. Playing announcements

This example prepares an announcement composed of two prompts.


```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
<dialogprepare>
   <dialog>
    <prompt>
      <media src="http://www.example.com/media/Number_09.wav"/>
      <media src="http://www.example.com/media/Number_11.wav"/>
    </prompt>
   </dialog>
</dialogprepare>
</mscivr>
```

If the dialog is prepared successfully, a <response> with status 200 is returned:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
  <response status="200" dialogid="vxi78"/>
</mscivr>
```

The prepared dialog is then started on a conference playing the prompts twice:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart prepareddialogid="vxi78" conferenceid="conference11"/>
</mscivr>
```

In the case of a successful dialog, the output is provided in <event>; for example

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="vxi78">
    <dialogexit status="1">
       <promptinfo termmode="completed"/>
    </dialogexit>
 </event>
</mscivr>
```

6.2.2. Prompt and collect

This example plays no prompts and just waits for DTMF input from the user:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="7HDY839~HJKSkyHS~HUwkuh7ns">
  <dialog>
   <collect/>
  </dialog>
 </dialogstart>
```



```
</mscivr>
```

If the dialog is successful, then dialogexit <event> contains the dtmf collected in its result parameter:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="vxi80">
    <dialogexit status="1">
       <collectinfo dtmf="12345" termmode="match"/>
    </dialogexit>
</event>
/mscivr>
```

In this example, a prompt is played and then we wait for 3 hours for a two digit sequence:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="7HDY839~HJKSkyHS~HUwkuh7ns">
 <dialog>
  <prompt>
   <media src="http://www.example.com/prompt1.wav"/>
  </prompt>
  <collect timeout="1080s" maxdigits="2"/>
 </dialog>
 </dialogstart>
</mscivr>
```

If no user input is collected within 3 hours, then following would be returned:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="vxi81">
    <dialogexit status="1" >
       <collectinfo termmode="noinput"/>
    </dialogexit>
 </event>
</mscivr>
```

And finally in this example, one of the input parameters is invalid:


```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
<dialogstart connectionid="7HDY839~HJKSkyHS~HUwkuh7ns">
 <dialog repeatCount="two">
   <prompt>
     <media src="http://www.example.com/prompt1.wav"/>
   </prompt>
   <collect cleardigitbuffer="true" bargein="true"
```

```
      timeout="4s" interdigittimeout="2s"
      termtimeout="0s maxdigits="2"/>
    </dialog>
   </dialogstart>
   </mscivr>
```

The error is reported in the response:

```
   <mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
    <response status="411" dialogid="vxi82"
     reason="repeatCount value invalid: two"/>
   </mscivr>
```

6.2.3. Prompt and record

In this example, the user is prompted, then their input is recorded for a maximum of 30 seconds.

```
   <mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
   <dialogstart connectionid="7HDY839~HJKSkyHS~HUwkuh7ns">
       <dialog>
         <prompt>
          <media src="http://www.example.com/media/sayname.wav"/>
         </prompt>
         <record dtmfterm="false" maxtime="30s" beep="true"/>
       </dialog>
   </dialogstart>
   </mscivr>
```

If successful and the recording is terminated by DTMF, the following is returned in a dialogexit <event>:

```
   <mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
    <event dialogid="vxi83">
     <dialogexit status="1">
     <recordinfo recording="http://www.example.com/recording1.wav"
                 termmode="dtmf"/>
     </dialogexit>
    </event>
```

```
</mscivr>
```

6.2.4. Runtime controls

In this example, a prompt is played with collect and runtime controls are activated.

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="7HDY839~HJKSkyHS~HUwkuh7ns">
 <dialog>
  <prompt bargein="true">
   <media src="http://www.example.com/prompt1.wav"/>
  </prompt>
  <control ffkey=5" rwkey="6" speedupkey="3"
           speeddownkey="4"/>
  <collect maxdigits="2"/>
 </dialog>
 </dialogstart>
</mscivr>
```

Once the dialog is active, the user can press keys 3, 4, 5 and 6 to execute runtime controls on the prompt queue. The keys do not cause bargein to occur. If the user presses any other key, then the prompt is interrupted and DTMF collect begins. Note that runtime controls are not active during the collect operation.

When the dialog is completed successfully, then both control and collect information is reported.

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="vxi81">
    <dialogexit status="1">
       <promptinfo termmode="bargein"/>
       <collectinfo termmode="match" dtmf="14"/>
       <controlinfo>
        <controlmatch dtmf="4" timestamp="2008-05-12T12:13:14Z"/>
        <controlmatch dtmf="3" timestamp="2008-05-12T12:13:15Z"/>
        <controlmatch dtmf="5" timestamp="2008-05-12T12:13:16Z"/>
       <controlinfo>
    </dialogexit>
 </event>
</mscivr>
```

6.2.5. Subscriptions and notifications

In this example, a looped dialog is started with subscription for notifications each time the user input matches the collect grammar:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart repeatCount="0" connectionid="7HDY839~HJKSkyHS~HUwkuh7ns">
 <dialog>
   <collect maxdigits="2"/>
   <subscribe>
    <dtmfsub matchmode="collect"/>
   </subscribe>
 </dialog>
 </dialogstart>
</mscivr>
```

Each time the user input the DTMF matching the grammar, the following notification event would be sent:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="vxi81">
    <dtmfnotify matchmode="collect" dtmf="12"
     timestamp="2008-05-12T12:13:14Z"/>
 </event>
</mscivr>
```

If no user input was provided, or the input did not match the grammar, the dialgo would continue to loop until terminated (or an error occured).

6.3. Other Dialog types

The following example requests that a VoiceXML dialog is started:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart dialogid="d2" type="application/voicexml+xml"
                src="http://www.example.com/mydialog.vxml"
                fetchtimeout="15s">
  <params>
   <param name="prompt1">nfs://nas01/media1.3gp"</param>
   <param name="prompt2">nfs://nas01/media2.3gp"</param>
  </params>
 </dialogstart>
</mscivr>
```

If the MS does not support this dialog type, rhen the response would have the status code 409. However, if it does support the VoiceXML dialog type, it would respond with a 200 status, activate the

VoiceXML dialog and may make the <params> available in the VoiceXML script through the "connection.ccxml.values" object.

When the VoiceXML dialog exits, exit parameters may be specified on the dialogexit event:


```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="d2">
   <dialogexit status="1">
    <params>
     <param name="username">peter</param>
     <param name="pin">1234</param>
    </params>
   </dialogexit>
  </event>
</mscivr>
```

7. Security Considerations

As this control package uses XML markup, implementation MUST address the security considerations of [RFC3023].

8. IANA Considerations

This specification instructs IANA to register a new Media Control Channel Framework Package, a new XML namespace and a new mime type.

8.1. Control Package Registration

Control Package name: msc-ivr/1.0

8.2. URN Sub-Namespace Registration

XML namespace: urn:ietf:params:xml:ns:msc-ivr

8.3. Mime Type Registration

MIME type: application/msc-ivr+xml

9. Change Summary

Note to RFC Editor: Please remove this whole section.

The following are the major changes between the -00 of this work group item draft and the individual submission -05 version.

- [IVR01] When the MS sends a notification event in a CONTROL, the AS sends mandatory 200 response (no extended transaction).
- [IVR23] Added a top-level container element, <mscivr>, with version attribute.
- Removed term 'basic' in title, description, elements and IANA registration. Control package name is now 'msc-ivr/1.0'.

Namespace is now 'urn:ietf:params:xml:ns:msc-ivr'. Mime type is now 'application/msc-ivr+xml'. Renamed 'basicivr' element to 'dialog' and moved version attribute to mscivr element.

- [IVR15] Updated and simplified XML schema. Ordering of child elements is significant.

- [IVR06] Removed 'volume' and 'offset' from prompt element. Added 'soundLevel' and 'clipBegin' to media element.

- [IVR17]/[IVR06] Removed 'iterations' and 'duration' from prompt. Added 'repeatCount' and 'repeatDur' to dialog element.

- Moved VCR commands from <collect> into separate <control> element. Defined controlinfo element to report rumtime control match information.

- [IVR05] Added <subscribe> to <dialogstart> where AS can subscribe to DTMF keypresses (all, control match only, collect match only). Extended <event> to support associated notification.

- Moved definition of <stream> into a separate section.

- [IVR21] Added audit capability: auditing of package capabilities and managed dialogs

- [IVR21] Explicitly stated that an error must be reported if the connection or conference referenced in a <dialogstart> is not available at the time the request is processed on the MS.

- Clarified that the <variable> rendering mechanism is MS implementation specific.

- [IVR09]/[IVR10] Clarified <variable> attribute definitions and added 'gender' attrribute.

- [IVR16] Clarified that info must be reported in dialogexit, if the correpsonding element is specified in a <dialog>. For example, if <prompt> is specified, then <promptinfo> must be specified if the dialog terminates normally.

- [IVR18] Added 'inactive' value for direction attribute of <stream>.

- [IVR19] Clarified case of <dialogstart> on connection/conference with multiple streams of the same type: recommended to be set explcitly with <stream>s.

- [IVR02] Clarified that multiple dialogs may started simultaneously on the same connection or conference.

- [IVR20] Added maximum duration (10 minutes) for a dialog to remain in the PREPARED state.

- Added <params> in <dialogstart> and <dialogexit> for input/output in other dialog types

- [IVR22] Added fetchtimeout parameter to dialogprepare, dialogstart, media and grammar elements.

- [IVR04] Added dialogexit status to indicate the connection or conference has been terminated. Added others status errors.

- [IVR08] Clarified that the <control> operation does not interrupt playing prompts and that matched DTMF is not available to <collect> or <record> operations during prompt playback.

- [IVR11] Added runtime controls for speed, goto start/end and external controls.

- Clarified that recordings can be uploaded to dest during or after the recording operation.

- <record>/<collect>: clarified timer handling - timeout refers to waiting time for collect/record to begin.

- Clarified behavior of immediate attribute on <dialogterminate>.

- clarified dialogid lifecycle: dialogids can be re-cycled.

- Clarified error handling.

- Editorial tidy up of sections.

- dialogid attribute on <response> is now mandatory.

- Clarified that the duration specified in finalsilence attribute of <record> is not part of the final recording.

- Clarified that the SRGS XML grammar format is mandatory

The following are the major changes between the -06 of the draft and the -05 version.

- Event notifications are sent in CONTROL messages with the MS acting as Control Framework Client. Compared with the previous approach, this means that a <dialogstart> transaction is now complete when the MS sends a <response>. A new transaction is initiated by the MS each time the MS sends a notification <event> to the AS.

- Changed conf-id to conferenceid and connection-id to connectionid.

- Clarification of the state model for dialogs

- <dialogprepare>: modified definition of src attribute to allow reference to external dialog documents; added (MIME) type attribute; removed <data> child element.

- <dialogstart>: modified definition of src attribute to allow reference to external dialog documents; added (MIME) type attribute; removed <data> child element;

- <dialogterminate>: modified so that a dialogexit event is always sent for active dialogs (i.e. the dialogexit event is a terminating notification)

- <event> notification simplified and make more extensible. Manual notifications (via <subscribe> element) are removed from the basic package. A <dialogexit> event is defined as <event> child and it can be extended with additional child elements

- <data> element is removed.

- <subscribe> element removed.

- Replaced dialog templates with a general <dialog> element. It has child elements for playing media resource (<prompt>), collecting

DTMF (<collect>) and recording (<record>). The functionality is largely unchanged.

- <dialogprepare> and <dialogstart> are extended with <dialog> child element.

- <event> is extended with a <dialogexit> element which contains status and reported information (replacement for output parameters in template dialogs)

- Prompts: now structured as a <prompt> element with <media>, <variable> and <dtmf> children. The <prompt> element has xml:base attribute, bargein, iterations, duration, volume and offset attributes. The speed attribute is removed. A <media> element has src and type attributes. The maxage and maxstale attributes are removed.

- DTMF input: parameters now specified as attributes of a <collect> element. Custom grammar specified with a <grammar> element as child of <collect> element. Added 'escapekey' to allow the dialog to be retried. Added 'pauseinterval', 'pausekey' and 'resumekey' to allow the prompts to paused/resumed. Added 'volumeinterval', 'volupkey' and voldnkey' to add prompt volume to be increased/ decreased. Moved 'bargein' attribute to prompt.

- Recording: parameters now specified as attributes of <record> element. Added 'dest' and 'beep' attributes.

The following are the major changes between the -05 of the draft and the -04 version.

- Mainly an alignment/evaluation exercise with requirements produced by MEDIACTRL IVR design team.

- playannouncement parameters from Table 7 of '04' version are now reflected in text - schema to be updated.

- Added VCR commands based on MSCML.

The following are the major changes between the -04 of the draft and the -03 version.

- None.

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

- added "basicivr:" protocol to template dialog types which must be supported as values of the "src" attribute in <dialogprepare> and <dialogstart>. Note alternative: "/basicivr/playannouncement" offered in [RFC4240].

- added "basicivr:" URI schema to IANA considerations

- Added mimetype, vadinitial and vadfinal parameters to 'promptandrecord' dialog type

- updated references

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

- added version 1.0 to package name

- separate section for element definitions

- dialogterminate treated as request rather than notification

- simplified responses: single element <response>

- removed response elements: <dialogprepared>, <dialogstarted>, <errordialognotprepared>, <errordialognotstarted>

- simplified event notifications to single <event> element carried in a REPORT

- <dialogexit> element replaced with <event name="dialogexit">

- removed <dialoguser> element

- added <stream> element as child of <dialogstart>

- removed 'type' attribute from <dialogprepare> and <dialogstart>

- added dialogid attribute to <dialogprepare> and <dialogstart>

- removed template "Sample Implementation" section
- renamed <namelist> to <data>
- re-organized so that template details after general package framework and element description.

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

- Removed requirement for VoiceXML dialog support
- Added requirement for template dialog support

10. Contributors

Asher Shiratzky from Radvision provided valuable support and contributions to the early versions of this document.

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

11. Acknowledgments

The authors would like to thank Adnan Saleem of Radisys, Gene Shtirmer of Intel, Dave Burke of Google, Dan York of Voxeo and Steve Buko of dialogic for expert reviews of this work.

12. References

12.1. Normative References

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

12.2. Informative References

[CCXML10] Auburn, R J., "Voice Browser Call Control: CCXML Version 1.0", W3C Working Draft (work in progress), January 2007.

[H.248.1] "Gateway control protocol: Version 3", ITU-T Recommendation H.248.1.

[H.248.9] "Gateway control protocol: Advanced media server packages", ITU-T Recommendation H.248.9.

[I-D.ietf-mediactrl-sip-control-framework] Boulton, C., Melanchuk, T., and S. McGlashan, "Media Control Channel Framework", draft-ietf-mediactrl-sip-control-framework-02 (work in progress), April 2008.

[I-D.ietf-xcon-common-data-model] Novo, O., Camarillo, G., Morgan, D., and R. Even, "Conference Information Data Model for Centralized Conferencing (XCON)", draft-ietf-xcon-common-data-model-10 (work in progress), March 2008.

[MSML] Saleem, A., Xin, Y., and G. Sharratt, "Media Session Markup Language (MSML)", draft-saleem-msml-06 (work in progress), February 2008.

[RFC2897] Cromwell, D., "Proposal for an MGCP Advanced Audio Package", RFC 2897, August 2000.

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

[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.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4240] Burger, E., Van Dyke, J., and A. Spitzer, "Basic Network Media Services with SIP", RFC 4240, December 2005.

[RFC4281] Gellens, R., Singer, D., and P. Frojdh, "The Codecs

Parameter for "Bucket" Media Types", RFC 4281, November 2005.

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

[RFC4646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 4646, September 2006.

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

[RFC4730] Burger, E. and M. Dolly, "A Session Initiation Protocol (SIP) Event Package for Key Press Stimulus (KPML)", RFC 4730, November 2006.

[RFC4733] Schulzrinne, H. and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals", RFC 4733, December 2006.

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

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

[SRGS] Hunt, A. and S. McGlashan, "Speech Recognition Grammar Specification Version 1.0", W3C Recommendation, March 2004.

[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.

[VXML21] Oshry, M., Auburn, RJ., Baggia, P., Bodell, M., Burke, D., Burnett, D., Candell, E., Carter, J., McGlashan, S., Lee, A., Porter, B., and K. Rehor, "Voice Extensible Markup Language (VoiceXML) Version 2.1", W3C Recommendation, June 2007.

[W3C.REC-SMIL2-20051213]
Michel, T., Mullender, S., Jansen, J., Koivisto, A.,

Zucker, D., Grassel, G., Bulterman, D., and N. Layaida, "Synchronized Multimedia Integration Language (SMIL 2.1)", World Wide Web Consortium Recommendation REC-SMIL2-20051213, December 2005, <http://www.w3.org/TR/2005/REC-SMIL2-20051213>.

[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.

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

Authors' Addresses

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

Email: scott.mcglashan@hp.com

Tim Melanchuk
Rain Willow Communications

Email: tim.melanchuk@gmail.com

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

Email: cboulton@avaya.com

Full Copyright Statement

Intellectual Property

http://www.ietf.org/ipr.

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