Internet Engineering Task Force (IETF)
Request for Comments: 6231
Category: Standards Track
ISSN: 2070-1721

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

May 2011

An Interactive Voice Response (IVR) Control Package for the Media Control Channel Framework

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 request elements for preparing, starting, and terminating dialog interactions, as well as associated responses and notifications. Dialog interactions are specified in a dialog language. This package defines a lightweight IVR dialog language (supporting prompt playback, runtime controls, Dual-Tone Multi-Frequency (DTMF) collection, and media recording) and allows other dialog languages to be used. The package also defines elements for auditing package capabilities and IVR dialogs.

Status of This Memo

This is an Internet Standards Track document.

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

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6231.

Table of Contents

1. Introduction

The Media Control Channel Framework [RFC6230] provides a generic approach for establishment and reporting capabilities of remotely initiated commands. The Channel Framework -- an equivalent term for the Media Control Channel Framework -- utilizes many functions provided by the Session Initiation Protocol (SIP) [RFC3261] for the rendezvous and establishment of a reliable channel for control interactions. The Control Framework also introduces the concept of a Control Package. A Control Package is an explicit usage of the Control Framework for a particular interaction set. This document defines a Control Package for Interactive Voice Response (IVR) dialogs on media connections and conferences. The term 'dialog' in this document refers to an IVR dialog and is completely unrelated to the notion of a SIP dialog. The term 'IVR' is used in its inclusive sense, allowing media other than voice for dialog interaction.

The package defines dialog management request elements for preparing, starting, and terminating dialog interactions, as well as associated responses and notifications. Dialog interactions are specified using a dialog language where the language specifies a well-defined syntax and semantics for permitted operations (play a prompt, record input from the user, etc.). This package defines a lightweight IVR dialog language (supporting prompt playback, runtime controls, DTMF collection, and media recording) and allows other dialog languages to be used. These dialog languages are specified inside dialog management elements for preparing and starting dialog interactions. The package also defines elements for auditing package capabilities and IVR dialogs.

This package has been designed to satisfy IVR requirements documented in "Media Server Control Protocol Requirements" [RFC5167] -- more specifically, REQ-MCP-28, REQ-MCP-29, and REQ-MCP-30. It achieves this 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 that are extensively deployed today.

First, the package is designed to provide the major IVR functionality of SIP media server languages such as netann [RFC4240], Media Server Control Markup Language (MSCML) [RFC5022], and Media Server Markup Language (MSML) [RFC5707], 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 Channel Framework.

Second, its design is aligned with key concepts of the web model as defined in W3C Voice Browser languages. The key dialog management mechanism is closely aligned with Call Control XML (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 [VXML30], we expect to see further alignment, especially in providing a set of basic independent primitive elements (such as prompt, collect, record, and runtime controls) that can be reused 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 that is familiar to current IVR developers and sufficient for most IVR applications, as well as a path to other languages that address more advanced applications.

This Control Package defines a lightweight IVR dialog language. The scope of this dialog language 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 language are more advanced functions including ASR (Automatic Speech Recognition), TTS (Text-to-Speech), fax, automatic prompt recovery ('media fallback'), and media transformation. Such functionality can be addressed by other dialog languages (such as VoiceXML) used with this package, extensions to this package (addition of foreign elements or attributes from another namespace), or other Control 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 that defines a lightweight IVR dialog language used with 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 message if it can execute in time) with a response element indicating whether or not the operation was successful (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 can 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 successfully, 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.

The dialog management elements for preparing and starting a dialog specify the dialog using a dialog language. A dialog language has well-defined syntax and semantics for defined dialog operations. Typically, dialog languages are written in XML where the root element has a designated XML namespace and, when used as standalone documents, have an associated MIME media type. For example, VoiceXML is an XML dialog language with the root element <vxml> with the designated namespace 'http://www.w3.org/2001/vxml' and standalone documents are associated with the MIME media type 'application/ voicexml+xml' [RFC4267].

This Control Package defines its own lightweight IVR dialog language. The language has a root element (<dialog>) with the same designated namespace as used for other elements defined in this package (see Section 8.2). The root element 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 so as to provide 'play announcement', 'prompt and collect', as well as 'prompt and record' functionality.

The dialog management elements for preparing and starting a dialog can specify the dialog language either by including inline a fragment with the root element or by referencing an external dialog document. The dialog language defined in this package is specified inline. Other dialog languages, such as VoiceXML, can be used by referencing an external dialog document.

The document is organized as follows. Section 3 describes how this Control Package fulfills the requirements for a Media Control Channel Framework Control Package. Section 4 describes the syntax and semantics of defined elements, including 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. Section 7 describes important security considerations for use of this Control Package. Section 8 provides information on IANA registration of this Control Package, including its name, XML namespace, and MIME media type. It also establishes a registry for prompt variables. Finally, Section 9 provides additional information on using VoiceXML when supported as an external dialog language.

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 following the concept of an IVR (Interactive Voice Response) dialog (this sense of 'dialog' is completely unrelated to a SIP dialog). A dialog is specified as inline XML or via a URI reference to an external dialog document. Traditional IVR dialogs typically feature capabilities such as playing audio prompts, collecting DTMF input, and recording audio input from the user. More inclusive definitions 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 that 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 7 of [RFC6230].

3.1. Control Package Name

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

The name 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.

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

3.2. Framework Message Usage

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

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

In this package, the MS operates as a Control Server in receiving requests from, and sending responses to, the AS (operating as Control 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 7 of [RFC6230]) are used when the request or event notification is invalid; for example, a request is invalid XML (400), or not understood (500).

The MS also operates as a Control Client in sending event notification to the AS (Control Server). Event notifications (Section 4.2.5) 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 A.1 of [RFC6230] MUST be supported for media dialogs and conferences.

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

3.4. CONTROL Message Body

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

When operating as Control Server, the MS receives Control message bodies with the MIME media type defined in Section 8.4 and containing an <mscivr> element (Section 4.1) with either a dialog management or audit request child element.

The following dialog management request elements are carried in CONTROL message bodies to the 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 Client, the MS sends CONTROL messages with the MIME media type defined in Section 8.4 and a body containing an <mscivr> element (Section 4.1) with a notification <event> child 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 indicates the location of detailed syntax definitions and semantics for the appropriate body types.

When operating as Control Server, the MS sends REPORT bodies with the MIME media type defined in Section 8.4 and containing a <mscivr> element (Section 4.1) 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 an <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 Package supports auditing of package capabilities and dialogs on the MS. An audit request is carried in a CONTROL message (see Section 3.4) and an audit response in a REPORT message (or a 200 response to the CONTROL if it can execute the audit in time) (see Section 3.5).

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

3.7. Examples

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

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

4. Element Definitions

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

The root element is <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 address the Security Considerations described in Section 7.

Implementation of this Control Package MUST adhere to the syntax and semantics of XML elements defined in this section and the schema (Section 5). Since XML schema is unable to support some types of syntactic constraints (such as attribute and element co-occurrence), some elements in this package specify additional syntactic constraints in their textual definition. If there is a difference in constraints between the XML schema and the textual description of elements in this section, the textual definition takes priority.

The XML schema supports extensibility by allowing attributes and elements from other namespaces. Implementations MAY support additional capabilities by means of attributes and elements from other (foreign) namespaces. Attributes and elements from foreign namespaces are not described in this section.

Some elements in this Control Package contain attributes whose value is a URI. These elements include: <dialogprepare> (Section 4.2.1), <dialogstart> (Section 4.2.2), <media> (Section 4.3.1.5), <grammar> (Section 4.3.1.3.1), and <record> (Section 4.3.1.4). The MS MUST support both HTTP [RFC2616] and HTTPS [RFC2818] protocol schemes for fetching and uploading resources, and the MS MAY support other schemes. The implementation SHOULD support storage of authentication information as part of its configuration, including security certificates for use with HTTPS. If the implementation wants to support user authentication, user certifications and passwords can also be stored as part of its configuration or the implementation can extend the schema (adding, for example, an http-password attribute in its own namespace) and then map user authentication information onto the appropriate headers following the HTTP authentication model [RFC2616].

Some elements in this Control Package contain attributes whose value is descriptive text primarily for diagnostic use. The implementation can indicate the language used in the descriptive text by means of a 'desclang' attribute ([RFC2277], [RFC5646]). The desclang attribute can appear on the root element as well as selected subordinate elements (see Section 4.1). The desclang attribute value on the root element applies to all desclang attributes in subordinate elements unless the subordinate element has an explicit desclang attribute that overrides it.

Usage examples are provided in Section 6.

4.1. <mscivr>

The <mscivr> element has the following attributes (in addition to standard XML namespace 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.

desclang: specifies the language used in descriptive text attributes of subordinate elements (unless the subordinate element provides a desclang attribute that overrides the value for its descriptive text attributes). The descriptive text attributes on subordinate elements include: the reason attribute on <response> (Section 4.2.4), <dialogexit> (Section 4.2.5.1), and <auditresponse> (Section 4.4.2); desc attribute on <variabletype> and <format> (Section 4.4.2.2.5.1). A valid value is a language identifier (Section 4.6.11). The attribute is optional. The default value is i-default (BCP 47 [RFC5646]).

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 connectionid="ssd3r3:sds345b">
  <dialog>
   <prompt>
    <media loc="http://www.example.com/welcome.wav"/>
   </prompt>
  </dialog>
 </dialogstart>
</mscivr>
```

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

```
<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"
        desclang="en">
 <event dialogid="d1">
  <dialogexit status="1" reason="successful completion of the dialog">
   <promptinfo termmode="completed"/>
  </dialogexit>
 </event>
</mscivr>
```

The language of the descriptive text in the reason attribute of <dialogexit> is explicitly indicated by the desclang attribute of the <mscivr> root element.

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 (prepared) dialog on a connection or conference

<dialogterminate>: terminate a dialog

Responses from the MS describe the status of the requested operation. Responses are specified in a <response> element (Section 4.2.4) that includes a mandatory attribute describing the status in terms of a numeric code. Response status codes are defined in Section 4.5. The MS MUST respond to a request message with a response message. If the MS is not able to process the request and carry out the dialog operation, the request has failed and the MS MUST indicate the class of failure using an appropriate 4xx response code. Unless an error response code is specified for a class of error within this section, implementations follow Section 4.5 in determining the appropriate status code for 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).

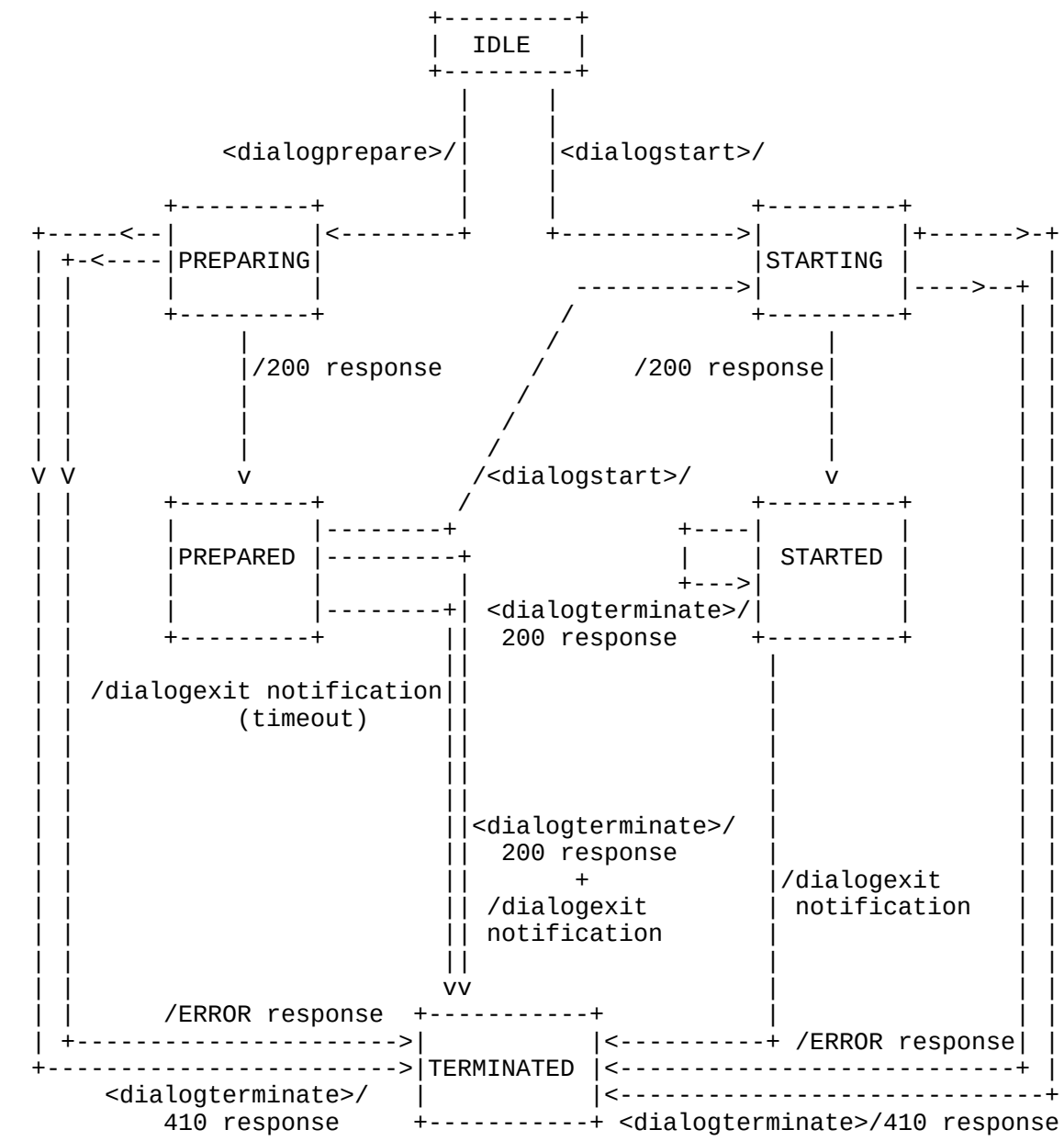


Figure 1: Dialog Lifecycle

The MS implementation MUST adhere to the dialog lifecycle shown in Figure 1, where each dialog has the following states:

IDLE: the dialog is uninitialized.

PREPARING: the dialog is being prepared. The dialog is assigned a valid dialog identifier (see below). If an error occurs, the dialog transitions to the TERMINATED state and the MS MUST send a response indicating the error. If the dialog is terminated before preparation is complete, the dialog transitions to the TERMINATED state and the MS MUST send a 410 response (Section 4.5) for the prepare request.

PREPARED: the dialog has been successfully prepared and the MS MUST send a 200 response indicating the prepare operation was successful. If the dialog is terminated, then the MS MUST send a 200 response, the dialog transitions to the TERMINATED state and the MS MUST send a dialogexit notification event (see Section 4.2.5.1). 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 the appropriate error status code (see Section 4.2.5.1). A maximum preparation duration of 300s is RECOMMENDED.

STARTING: the dialog is being started. If the dialog has not already been prepared, it is first prepared and assigned a valid dialog identifier (see below). If an error occurs the dialog transitions to the TERMINATED state and the MS MUST send a response indicating the error. If the dialog is terminated, the dialog transitions to the TERMINATED state and the MS MUST send a 410 response (Section 4.5) for the start request.

STARTED: the dialog has been successfully started and is now active. The MS MUST send a 200 response indicating the start operation was successful. If any dialog events occur that were subscribed to, the MS MUST send a notifications when the dialog event occurs. When the dialog exits (due to normal termination, an error, or a terminate request), the MS MUST send a dialogexit notification event (see Section 4.2.5.1) and the dialog transitions to the TERMINATED state.

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

Each dialog has a valid identifier until it transitions to a TERMINATED state. The dialog identifier is assigned by the MS unless the <dialogprepare> or <dialogstart> request already specifies a

identifier (dialogid) that is not associated with any other dialog on the MS. Once a dialog is in a TERMINATED state, its dialog identifier is no longer valid and can be reused for another dialog.

The identifier is used to reference the dialog in subsequent requests, responses, and notifications. In a <dialogstart> request, the dialog identifier can be specified in the prepareddialogid attribute indicating the prepared dialog to start. In <dialogterminate> and <audit> requests, the dialog identifier is specified in the dialogid attribute, indicating which dialog is to be terminated or audited, respectively. If these requests specify a dialog identifier already associated with another dialog on the MS, the MS sends a response with a 405 status code (see Section 4.5) and the same dialogid as in the request. The MS MUST specify a dialog identifier in notifications associated with the dialog. The MS MUST specify a dialog identifier in responses unless it is a response to a syntactically invalid request.

For a given dialog, the <dialogprepare> or <dialogstart> request elements specify the dialog content to execute either by including inline a <dialog> element (the dialog language defined in this package; see Section 4.3) or by referencing an external dialog document (a dialog language defined outside this package). When referencing an external dialog document, the request element contains a URI reference to the remote document (specifying the dialog definition) and, optionally, a type attribute indicating the MIME media type associated with the dialog document. Consequently, the dialog language associated with a dialog on the MS is identified either inline by a <dialog> child element or by a src attribute referencing a document containing the dialog language. The MS MUST support inline the IVR dialog language defined in Section 4.3. The MS MAY support other dialog languages by reference.

4.2.1. <dialogprepare>

The <dialogprepare> request is sent to the MS to request preparation of a dialog. Dialog preparation consists of (a) retrieving an external dialog document and/or external resources referenced within an inline <dialog> element 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). The MS MUST support both HTTP [RFC2616] and HTTPS [RFC2818] schemes and the MS MAY support other schemes. If the URI scheme is unsupported, the MS sends a <response> with a 420 status code (Section 4.5). If the document cannot be retrieved within the timeout interval, the MS sends a <response> with a 409 status code. If the document contains a type of dialog language that the MS does not support, the MS sends a <response> with a 421 status code. The attribute is optional. There is no default value.

type: specifies the type of the external dialog document indicated in the 'src' attribute. A valid value is a MIME media type (see Section 4.6.10). If the URI scheme used in the src attribute defines a mechanism for establishing the authoritative MIME media type of the media resource, the value returned by that mechanism takes precedence over this attribute. The attribute is optional. There is no default value.

maxage: Used to set the max-age value of the 'Cache-Control' header in conjunction with an external dialog document fetched using HTTP, as per [RFC2616]. A valid value is a non-negative integer (see Section 4.6.4). The attribute is optional. There is no default value.

maxstale: Used to set the max-stale value of the 'Cache-Control' header in conjunction with an external dialog document fetched using HTTP, as per [RFC2616]. A valid value is a non-negative integer (see Section 4.6.4). The attribute is optional. There is no default value.

fetchtimeout: the maximum timeout 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. If a dialog with the same name already exists on the MS, the MS sends a <response> with a 405 status code (Section 4.5). If this attribute is not specified, the MS MUST create a unique name for the dialog (see Section 4.2 for dialog identifier assignment). The attribute is optional. There is no default value.

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

<dialog> an IVR dialog (Section 4.3) to prepare. The element is optional.

<params>: specifies input parameters (Section 4.2.6) for dialog languages defined outside this specification. The element is optional. If a parameter is not supported by the MS for the external dialog language, the MS sends a <response> with a 427 status code (Section 4.5).

The dialog to prepare can be specified either inline with a <dialog> child element or externally (for dialog languages defined outside this specification) using the src attribute. It is a syntax error if both an inline <dialog> element and a src attribute are specified and the MS sends a <response> with a 400 status code (see Section 4.5). The type, maxage, maxstale, 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 loc="http://www.example.com/welcome.wav"/>
   </prompt>
  </dialog>
 </dialogprepare>
</mscivr>
```

In this example, a request with a specified dialogid to prepare a VoiceXML dialog document located externally:

```
<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 languages other than the IVR dialog language defined in this package is optional, if the MS does not support the dialog language, it would send a response with the status code 421 (Section 4.5). Further information on using VoiceXML can be found in Section 9.

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 external document and/or external resources referenced within <dialog> element 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). The MS MUST support both HTTP [RFC2616] and HTTPS [RFC2818] schemes and the MS MAY support other schemes. If the URI scheme is unsupported, the MS sends a <response> with a 420 status code (Section 4.5). If the document cannot be retrieved with the timeout interval, the MS sends a <response> with a 409 status code. If the document contains a type of dialog language that the MS does not support, the MS sends a <response> with a 421 status code. The attribute is optional. There is no default value.

type: specifies the type of the external dialog document indicated in the 'src' attribute. A valid value is a MIME media type (see Section 4.6.10). If the URI scheme used in the src attribute defines a mechanism for establishing the authoritative MIME media type of the media resource, the value returned by that mechanism takes precedence over this attribute. The attribute is optional. There is no default value.

maxage: Used to set the max-age value of the 'Cache-Control' header in conjunction with an external dialog document fetched using HTTP, as per [RFC2616]. A valid value is a non-negative integer (see Section 4.6.4). The attribute is optional. There is no default value.

maxstale: Used to set the max-stale value of the 'Cache-Control' header in conjunction with an external dialog document fetched using HTTP, as per [RFC2616]. A valid value is a non-negative integer (see Section 4.6.4). The attribute is optional. There is no default value.

fetchtimeout: the maximum timeout 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. If a dialog with the same name already exists on the MS, the MS sends a <response> with a 405 status code (Section 4.5). If neither the dialogid attribute nor the prepareddialogid attribute is specified, the MS MUST create a unique name for the dialog (see Section 4.2 for dialog identifier assignment). The attribute is optional. There is no default value.

prepareddialogid: string identifying a dialog previously prepared using a dialogprepare (Section 4.2.1) request. If neither the dialogid attribute nor the prepareddialogid attribute is specified, the MS MUST create a unique name for the dialog (see Section 4.2 for dialog identifier assignment). 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 Appendix A.1 of [RFC6230]). The attribute is optional. There is no default value.

conferenceid: string identifying the conference on which this dialog is to be started (see Appendix A.1 of [RFC6230]). The attribute is optional. There is no default value.

Exactly one of the connectionid or conferenceid attributes MUST be specified. If both the connectionid and conferenceid attributes are specified or neither is specified, it is a syntax error and the MS sends a <response> with a 400 status code (Section 4.5).

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. If an invalid connectionid is specified, the MS sends a <response> with a 407 status code (Section 4.5). If an invalid conferenceid is specified, the MS sends a <response> with a 408 status code.

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

<dialog>: specifies an IVR dialog (Section 4.3) 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 dialog languages defined outside this specification. The element is optional. If a parameter is not supported by the MS for the external dialog language, the MS sends a <response> with a 427 status code (Section 4.5).

<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 can be specified.

The dialog to start can be specified either (a) inline with a <dialog> child element, (b) externally using the src attribute (for dialog languages defined outside this specification), or (c) by referencing a previously prepared dialog using the prepareddialogid attribute. If exactly one of the src attribute, the prepareddialogid, or a <dialog> child element is not specified, it is a syntax error and the MS sends a <response> with a 400 status code (Section 4.5). If the prepareddialogid and dialogid attributes are specified, it is also a syntax error and the MS sends a <response> with a 400 status code. The type, maxage, maxstale, 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, the AS MUST use <stream> elements to explicitly specify the configuration. If no <stream> elements are specified, then the default media configuration is that defined for the connection or conference.

If a <stream> element is in conflict (a) with another <stream> element, (b) with specified connection or conference media capabilities, or (c) with a Session Description Protocol (SDP) label value as part of the connectionid (see Appendix A.1 of [RFC6230]), then the MS sends a <response> with a 411 status code (Section 4.5). If the media stream configuration is not supported by the MS, then the MS sends a <response> with a 428 status code (Section 4.5).

The MS MAY support multiple, simultaneous dialogs being started on the same connection or conference. For example, the same connection can receive different media streams (e.g., audio and video) from different dialogs, or receive (and implicitly mix where appropriate) the same type of media streams from different dialogs. If the MS does not support starting another dialog on the same connection or conference, it sends a <response> with a 432 status code (Section 4.5) when it receives the second (or subsequent) dialog request.

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

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

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

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

4.2.2.1. <subscribe>

The <subscribe> element allows the AS to subscribe to, and be notified of, specific events that 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 occurrences):

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

If a request has a <subscribe> with no child elements, the MS treats the request as if no <subscribe> element were specified.

The MS MUST support <dtmfsub> subscription for the IVR dialog language defined in this specification (Section 4.3). It MAY support other dialog subscriptions (specified using attributes and child elements from a foreign namespace). If the MS does not support a

subscription specified in a foreign namespace, the MS sends a response with a 431 status code (see Section 4.5).

4.2.2.1.1. <dtmfsub>

The <dtmfsub> element has the following attributes:

matchmode: controls which DTMF input is subscribed to. Valid values are "all" - notify all DTMF key presses 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 key press matching a runtime control:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart dialogid="d3" connectionid="connection1">
  <dialog>
   <prompt>
    <media loc="http://www.example.com/getpin.wav"/>
   </prompt>
   <control ffkey="2" rwkey="3"/>
  </dialog>
 <subscribe>
  <dtmfsub 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. A valid value is a MIME type-name as defined in Section 4.2 of [RFC4288]. The following values MUST be used for common types of media: "audio" for audio media, and "video" for video media. See [IANA] for registered MIME type names. The attribute is mandatory.

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

direction: a string indicating the direction of the media flow relative to the endpoint conference or connection. Defined values are "sendrecv" (the endpoint can send media to, and receive media from, the dialog), "sendonly" (the endpoint can only send media to the dialog), "recvonly" (the endpoint can only receive media from the dialog), and "inactive" (stream is not to be used). The default value is "sendrecv". The attribute is optional.

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

<region>: an element to specify the area within a mixer video layout where a media stream is displayed (Section 4.2.2.2.1). The element is optional.

<priority>: an element to configure priority associated with the stream in the conference mix (Section 4.2.2.2.2). The element is optional.

If conferenceid is not specified or if the "media" attribute does not have the value of "video", then the MS ignores the <region> and <priority> 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="sendonly"/>
  <stream media="video" label="uservideo" direction="sendonly"/>
 </dialogstart>
</mscivr>
```

Using the <region> element, the dialog can be started on a conference mixer so that the video output from the dialog is directed to a specific area within a video layout. For example:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart conferenceid="conference1">
  <dialog>
   <prompt>
     <media loc="http://www.example.com/presentation.3gp"/>
   </prompt>
  </dialog>
  <stream media="video" direction="recvonly">
    <region>1</region>
  </stream>
 </dialogstart>
</mscivr>
```

4.2.2.2.1. <region>

The <region> element is used to specify a named area within a presentation layout where a video media stream is displayed. The MS could, for example, play video media into an area of a video layout where the layout and its named regions are specified using the Mixer Control Package [MIXER-CP].

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

If the region name is invalid, then the MS reports a 416 status code (Section 4.5) in the response to the request element containing the <region> element.

4.2.2.2.2. <priority>

The <priority> element is used to explicitly specify the priority of the dialog for presentation in a conference mix.

The <priority> element has no attributes and its content model specifies a positive integer (see Section 4.6.5). The lower the value, the higher the priority.

4.2.3. <dialogterminate>

A dialog 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. If the specified dialog identifier is invalid, the MS sends a response with a 405 status code (Section 4.5). The attribute is mandatory.

immediate: indicates whether or not a dialog in the STARTED state is to be terminated immediately (in other states, termination is always immediate). A valid value is a boolean (see Section 4.6.1). A value of true indicates that the dialog is terminated immediately and the MS MUST send a dialogexit notification (Section 4.2.5.1) without report information. 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.

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

For example, immediately terminating a STARTED 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"/>
</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.

desclang: specifies the language used in the value of the reason attribute. A valid value is a language identifier (Section 4.6.11). The attribute is optional. If not specified, the value of the desclang attribute on <mscivr> (Section 4.1) applies.

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 because the request is syntactically invalid (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 Appendix A.1 of [RFC6230]). The attribute is optional. There is no default value.

conferenceid: string identifying the conference associated with the dialog (see Appendix A.1 of [RFC6230]). 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 language:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="421" dialogid="d5"
    reason="Unsupported dialog language: 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 400 (Syntax error) status code and the dialogid attribute has an empty string value:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="400" 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 that 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 occurred (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, it has been terminated, or 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 the status of the dialog when it exits. A valid value is a non-negative integer (see Section 4.6.4). The MS MUST support the following values:

0 indicates the dialog has been terminated by a <dialogterminate> request.

1 indicates successful completion of the dialog.

2 indicates the dialog terminated because the connection or conference associated with the dialog has terminated.

3 indicates the dialog terminated due to exceeding its maximum duration.

4 indicates the dialog terminated due to an execution error.

All other valid but undefined values are reserved for future use, where new status codes are assigned using the Standards Action process defined in [RFC5226]. The AS MUST treat any status code it does not recognize as being equivalent to 4 (dialog execution error). The attribute is mandatory.

reason: a textual description that the MS SHOULD use to provide 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.

desclang: specifies the language used in the value of the reason attribute. A valid value is a language identifier (Section 4.6.11). The attribute is optional. If not specified, the value of the desclang attribute on <mscivr> (Section 4.1) applies.

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 language defined outside this specification. The element is optional.

For example, when 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>
</mscivr>
```

4.2.5.2. <dtmfnotify>

The <dtmfnotify> element provides 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 as follows:

"all" - all DTMF key presses notified individually;

"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 key presses received according to the matchmode. 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 last key press occurred according to the matchmode. A valid value is a dateTime expression (Section 4.6.12). The attribute is mandatory.

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

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="d3">
   <dtmfnotify matchmode="collect" dtmf="3123"
     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 language 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 type indicating how the inline value of the parameter is to be interpreted. A valid value is a MIME media type (see Section 4.6.10). The attribute is optional. The default value is "text/plain".

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

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

For example, usage with a dialog language 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" type="audio/x-wav" encoding="base64">
    <![CDATA[
     R0lGODlhZABqALMAAFrMYr/BvlKOVJKOg2xZUKmenMfDw8tgWJpV
    ]]>
   </param>
  </params>
  </dialogexit>
 </event>
</mscivr>
```

4.3. IVR Dialog Elements

This section describes the IVR dialog language defined as part of this specification. The MS MUST support this dialog language.

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 language 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 reuse 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 termmode attribute and one <mediainfo> element 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 that need to be supported by an implementation. For example, an MS implementation might only support 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 that 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.

repeatUntilComplete: indicates whether the MS terminates dialog execution when an input operation is completed successfully. A valid value is a boolean (see Section 4.6.1). A value of true indicates that dialog execution is terminated when an input operation associated with its child elements is completed successfully (see execution model below for precise conditions). A value of false indicates that dialog execution is terminated by other means. The attribute is optional. The default value is false.

The repeatDur attribute takes priority over the repeatCount attribute in determining maximum duration of the dialog. See 'repeatCount' and 'repeatDur' in the Synchronized Multimedia Integration Language (SMIL) [W3C.REC-SMIL2-20051213] for further information. In the situation where a dialog is repeated more than once, only the results of operations in the last dialog iteration are reported.

The <dialog> element has the following sequence of child elements (at least one, any order):

<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.

Although the behavior when both <collect> and <record> elements are specified in a request is not defined in this Control Package, the MS MAY support this configuration. If the MS does not support this configuration, the MS sends a <response> with a 433 status code.

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

1. If an error occurs during execution, then the MS terminates the dialog and reports the error 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. The MS initializes a counter to 0.

3. The MS starts a duration timer for the value of the repeatDur attribute. If the timer expires before the dialog is complete, then the MS terminates the dialog and sends a dialogexit whose status attribute is set to 3 (see Section 4.2.5.1). The MS MAY report information in the dialogexit gathered in the last execution cycle (if any).

4. The MS initiates a dialog execution cycle. Each cycle executes the operations associated with the child elements of the dialog. If a <prompt> element is specified, then execute the element's prompt playing operation and activate any controls (if the <control> element is specified). If no <prompt> is specified or when a specified <prompt> terminates, then start the collect operation or the record operation if the <collect> or <record> elements, respectively, are specified. If subscriptions are specified for the dialog, then the MS sends a notification event when the specified event occurs. If execution of a child element results in an error, the MS terminates dialog execution (and stops other child element operations) and the MS sends a dialogexit status event, reporting any information gathered.

5. If the dialog execution cycle completes successfully, then the MS increments the counter by one. The MS terminates dialog execution if either of the following conditions is true:

   * the value of the repeatCount attribute is greater than zero, and the counter is equal to the value of the repeatCount attribute.

   * the value of the repeatUntilComplete attribute is true and one of the following conditions is true:

     + <collect> reports termination status of 'match' or 'stopped'.

     + <record> reports termination status of 'stopped', 'dtmf', 'maxtime', or 'finalsilence'.

When the MS terminates dialog execution, it sends a dialogexit (with a status of 1) reporting operation information collected in the last dialog execution cycle only. Otherwise, another dialog execution cycle is initiated.

4.3.1.1. <prompt>

The <prompt> element specifies a sequence of media resources to play back in document order.

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 associated 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 (at least one, any order, multiple occurrences of elements permitted):

<media>: specifies a media resource (see Section 4.3.1.5) to play. The element is optional.

<variable>: specifies a variable media announcement (see Section 4.3.1.1.1) to play. The element is optional.

<dtmf>: generates one or more DTMF tones (see Section 4.3.1.1.2) to play. The element is optional.

<par>: specifies media resources to play in parallel (see Section 4.3.1.1.3). The element is optional.

If the MS does not support the configuration required for prompt playback to the output media streams and a more specific error code is not defined for its child elements, the MS sends a <response> with a 429 status code (Section 4.5). The MS MAY support transcoding between the media resource format and the output stream format.

The MS has the following execution model for prompt playing after initialization:

1. The MS initiates prompt playback playing its child elements (<media>, <variable>, <dtmf>, and <par>) one after another in document order.

2. If any error (including fetching and rendering errors) occurs during prompt execution, then the MS terminates playback and reports its error status to the dialog container (see Section 4.3) with a <promptinfo> (see Section 4.3.2.1) where the termmode attribute is set to stopped and any additional information is set.

3. If DTMF input is received and the value of the bargein attribute is true, then the MS terminates prompt playback and reports its execution status to the dialog container (see Section 4.3) with a <promptinfo> (see Section 4.3.2.1) where the termmode attribute is set to bargein and any additional information is set.

4. If prompt playback is stopped by the dialog container, then the MS reports its execution status to the dialog container (see Section 4.3) with a <promptinfo> (see Section 4.3.2.1) where the termmode attribute is set to stopped and any additional information is set.

5. If prompt playback completes successfully, then the MS reports its execution status to the dialog container (see Section 4.3) with a <promptinfo> (see Section 4.3.2.1) where the termmode attribute is set to completed and any additional information is set.

4.3.1.1.1. <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 prompt variable type (e.g., 2008-02-25 is rendered as the date 25th February 2008). The precise mechanism for generating variable announcements (including the location of associated media resources) is implementation specific.

A <variable> element has the following attributes:

type: specifies the type of prompt variable to render. This specification defines three values -- date (Section 4.3.1.1.1.1), time (Section 4.3.1.1.1.2), and digits (Section 4.3.1.1.1.3). All other valid but undefined values are reserved for future use,

where new values are assigned as described in Section 8.5. A valid value is a string (see Section 4.6.6). The attribute is mandatory.

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

format: specifies format information that the prompt variable type uses to render the value attribute. A valid value is a string (see Section 4.6.6). The attribute is optional. There is no default value.

gender: specifies the gender that the prompt variable type uses to render the value attribute. Valid values are "male" or "female". The attribute is optional. There is no default value.

xml:lang: specifies the language that the prompt variable type uses to render the value attribute. 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 specification is agnostic to the type and codec of media resources into which variables are rendered as well as the rendering mechanism itself. For example, an MS implementation supporting audio rendering could map the <variable> into one or more audio media resources.

This package is agnostic to which <variable> types are supported by an implementation. If a <variable> element configuration specified in a request is not supported by the MS, the MS sends a <response> with a 425 status code (Section 4.5).

4.3.1.1.1.1. Date Type

The date variable type provides a mechanism for dynamically rendering a date prompt.

The <variable> type attribute MUST have the value "date".

The <variable> format attribute MUST be one of the following values and comply with its rendering of the value attribute:

mdy indicating that the <variable> value attribute is to be rendered as sequence composed of month, then day, then year.

ymd indicating that the <variable> value attribute is to be rendered as sequence composed of year, then month, then day.

dym indicating that the <variable> value attribute is to be rendered as sequence composed of day, then year, then month.

dm indicating that the <variable> value attribute is to be rendered as sequence composed of day then month.

The <variable> value attribute MUST comply with a lexical representation of date where

yyyy '-' mm '-' dd

as defined in Section 3.2.9 of [XMLSchema:Part2].

For example,

```
<variable type="date" format="dmy" value="2010-11-25"
xml:lang="en" gender="male"/>
```

describes a variable date prompt where the date can be rendered in audio as "twenty-fifth of November two thousand and ten" using a list of <media> resources:

```
<media loc="nfs://voicebase/en/male/25th.wav"/>
<media loc="nfs://voicebase/en/male/of.wav"/>
<media loc="nfs://voicebase/en/male/november.wav"/>
<media loc="nfs://voicebase/en/male/2000.wav"/>
<media loc="nfs://voicebase/en/male/and.wav"/>
<media loc="nfs://voicebase/en/male/10.wav"/>
```

4.3.1.1.1.2. Time Type

The time variable type provides a mechanism for dynamically rendering a time prompt.

The <variable> type attribute MUST have the value "time".

The <variable> format attribute MUST be one of the following values and comply with its rendering of the value attribute:

t12 indicating that the <variable> value attribute is to be rendered as a time in traditional 12-hour format using am or pm (for example, "twenty-five minutes past 2 pm" for "14:25").

t24 indicating that the <variable> value attribute is to be rendered as a time in 24-hour format (for example, "fourteen twenty-five" for "14:25").

The <variable> value attribute MUST comply with a lexical representation of time where

hh ':' mm (':' ss)?

as defined in Section 3.2.8 of [XMLSchema:Part2].

4.3.1.1.1.3. Digits Type

The digits variable type provides a mechanism for dynamically rendering a digit sequence.

The <variable> type attribute MUST have the value "digits".

The <variable> format attribute MUST be one of the following values and comply with its rendering of the value attribute:

gen indicating that the <variable> value attribute is to be rendered as a general digit string (for example, "one two three" for "123").

crn indicating that the <variable> value attribute is to be rendered as a cardinal number (for example, "one hundred and twenty-three" for "123").

ord indicating that the <variable> value attribute is to be rendered as an ordinal number (for example, "one hundred and twenty-third" for "123").

The <variable> value attribute MUST comply with the lexical representation

```
d+
```

i.e., one or more digits.

4.3.1.1.2. <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). The MS MAY round the value if it only supports discrete durations. The attribute is optional. The default value is 100 ms.

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). The MS MAY round the value if it only supports discrete durations. The attribute is optional. The default value is 100 ms.

The <dtmf> element has no children.

If a <dtmf> element configuration is not supported, the MS sends a <response> with a 426 status code (Section 4.5).

4.3.1.1.3. <par>

The <par> element allows media resources to be played in parallel. Each of its child elements specifies a media resource (or a sequence of media resources using the <seq> element). When playback of the <par> element is initiated, the MS begins playback of all its child elements at the same time. This element is modeled after the <par> element in SMIL [W3C.REC-SMIL2-20051213].

The <par> element has the following attributes:

endsync: indicates when playback of the element is complete. Valid values are "first" (indicates that the element is complete when any child element reports that it is complete) and "last" (indicates it is complete when every child elements are complete). The attribute is optional. The default value is "last".

If the value is "first", then playback of other child elements is stopped when one child element reports it is complete.

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

<seq>: specifies a sequence of media resources to play in parallel with other <par> child elements (see Section 4.3.1.1.3.1). The element is optional.

<media>: specifies a media resource (see Section 4.3.1.5) to play. The MS is responsible for assigning the appropriate media stream(s) when more than one is available. The element is optional.

<variable>: specifies a variable media announcement (see Section 4.3.1.1.1) to play. The element is optional.

<dtmf>: generates one or more DTMF tones (see Section 4.3.1.1.2) to play. The element is optional.

It is RECOMMENDED that a <par> element contains only one <media> element of the same media type (i.e., same type-name as defined in Section 4.6.10). If a <par> element configuration is not supported, the MS sends a <response> with a 435 status code (Section 4.5).

Runtime <control>s (Section 4.3.1.2) apply to each child element playing in parallel. For example, pause and resume controls cause all child elements to be paused and resumed, respectively.

If the <par> element is stopped by the prompt container (e.g., bargein or dialog termination), then playback of all child elements is stopped. The playback duration (Section 4.3.2.1) reported for the <par> element is the duration of parallel playback, not the cumulative duration of each child element played in parallel.

For example, a request to playback audio and video media in parallel:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
<dialogstart connectionid="c1">
  <dialog>
   <prompt>
    <par>
     <media type="audio/x-wav"
            loc="http://www.example.com/media/comments.wav"/>
     <media type="video/3gpp;codecs='s263'"
            loc="http://www.example.com/media/camera.3gp"/>
    </par>
   </prompt>
  </dialog>
 </dialogstart>
</mscivr>
```

When the <prompt> element is executed, it begins playback of its child element in document-order sequence. In this case, there is only one child element, a <par> element itself containing audio and video <media> child elements. Consequently, playback of both audio and video media resources is initiated at the same time. Since the endsync attribute is not specified, the default value "last" applies. The <par> element playback is complete when the media resource with the longest duration is complete.

4.3.1.1.3.1. <seq>

The <seq> element specifies media resources to be played back in sequence. This allows a sequence of media resources to be played at the same time as other children of a <par> element are played in parallel, for example, a sequence of audio resources while a video resource is played in parallel. This element is modeled after the <seq> element in SMIL [W3C.REC-SMIL2-20051213].

The <seq> element has no attributes.

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

<media>: specifies a media resource (see Section 4.3.1.5) to play. The element is optional.

<variable>: specifies a variable media announcement (see Section 4.3.1.1.1) to play. The element is optional.

<dtmf>: generates one or more DTMF tones (see Section 4.3.1.1.2) to play. The element is optional.

Playback of a <seq> element is complete when all child elements in the sequence are complete. If the <seq> element is stopped by the <par> container, then playback of the current child element is stopped (remaining child elements in the sequence are not played).

For example, a request to play a sequence of audio resources in parallel with a video media:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
<dialogstart connectionid="c1">
  <dialog>
   <prompt>
    <par endsync="first">
     <seq>
       <media type="audio/x-wav"
            loc="http://www.example.com/media/date.wav"/>
       <media type="audio/x-wav"
            loc="http://www.example.com/media/intro.wav"/>
       <media type="audio/x-wav"
            loc="http://www.example.com/media/main.wav"/>
       <media type="audio/x-wav"
            loc="http://www.example.com/media/end.wav"/>
     </seq>
     <media type="video/3gpp;codecs='s263'"
            loc="rtsp://www.example.com/media/camera.3gp"/>
    </par>
   </prompt>
  </dialog>
 </dialogstart>
</mscivr>
```

When the <prompt> element is executed, it begins playback of the <par> element containing a <seq> element and a video <media> element. The <seq> element itself contains a sequence of audio <media> elements. Consequently, playback of the video media resource is initiated at the same time as playback of the sequence of the audio media resources is initiated. Each audio resource is played back after the previous one completes. Since the endsync attribute is set to "first", the <par> element playback is complete when either all the audio resources in <seq> have been played to completion or the video <media> is complete, whichever occurs first.

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 be interrupted (i.e., no prompt bargein), but causes the appropriate operation to be applied, for example, speeding up prompt playback.

DTMF input matching these controls has priority over <collect> input for the duration of prompt playback. If an incoming DTMF character matches a specified runtime control, then the DTMF character is consumed: it is not added to the digit buffer and so is not available to the <collect> operation. 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 an MS skips 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 an MS pauses 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.

If the same DTMF is specified in more than one DTMF key control attribute -- except the pausekey and resumekey attributes -- the MS sends a <response> with a 413 status code (Section 4.5).

The MS has the following execution model for runtime control after initialization:

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

2. Runtime controls are active only during prompt playback (if no <prompt> element is specified, then runtime controls are ignored). If DTMF input matches any specified keys (for example, the ffkey), then the MS applies the appropriate operation immediately. If a seek operation (ffkey, rwkey) attempts to go beyond the beginning or end of the prompt queue, then the MS automatically truncates it to the prompt queue beginning or end, respectively. If a volume operation (voldnkey, volupkey) attempts to go beyond the minimum or maximum volume supported by the platform, then the MS automatically limits the operation to minimum or maximum supported volume, respectively. If a speed operation (speeddnkey, speedupkey) attempts to go beyond the minimum or maximum playback speed supported by the platform, then the MS automatically limits the operation to minimum or maximum supported speed, 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 seek, volume, or speed operation is applied when output is paused, then the MS also resumes output automatically.

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 event (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 the maximum time to wait for another DTMF when the collected input is incomplete with respect to the grammar. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 2s.

termtimeout: indicates the maximum time to wait for the termchar character when the collected input is complete with respect to the grammar. A valid value is a Time Designation (see Section 4.6.7). The attribute is optional. The default value is 0s (no delay).

escapekey: specifies a DTMF key that indicates collected grammar matches are discarded and 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. It is ignored when a custom grammar is specified. 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. It is ignored when a custom grammar is specified. A valid value is a positive integer (see Section 4.6.5). The attribute is optional. The default value is 5.

The following matching priority is defined for incoming DTMF: termchar attribute, escapekey attribute, and then as part of a grammar. For example, if "1" is defined as the escapekey attribute and as part of a grammar, then its interpretation as an escapekey takes priority.

The <collect> element has the following child element:

<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.

The MS has the following execution model for DTMF collection after 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 the MS terminates collection and reports the error to the dialog container (see Section 4.3). The MS MAY report DTMF collected before the error in <collectinfo> (see Section 4.3.2.3).

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

4. The MS activates an initial timer with the duration of the value of the timeout attribute. If the initial timer expires before any DTMF input is received, then collection execution terminates, the <collectinfo> (see Section 4.3.2.3) has the termmode attribute set to noinput and the execution status is reported to the dialog container.

5. When the first DTMF collect input is received, the initial timer is canceled and DTMF collection begins. Each DTMF input is collected unless it matches the value of the escapekey attribute or the termchar attribute when the internal grammar is used. Collected input is matched against the grammar to determine if it is valid and, if valid, whether collection is complete. Valid DTMF patterns are either a simple digit string where the maximum length is determined by the maxdigits attribute and that can be optionally terminated by the character in the termchar attribute, or a custom DTMF grammar specified with the <grammar> element.

6. After escapekey input, or a valid input that does not complete the grammar, the MS activates a timer for the value of the interdigittimeout attribute or the termtimeout attribute. The MS only uses the termtimeout value when the grammar does not allow any additional input; otherwise, the MS uses the interdigittimeout.

7. If DTMF collect input matches the value of the escapekey attribute, then the MS re-initializes DTMF collection: i.e., the MS discards collected DTMFs already matched against the grammar, and the MS attempts to match incoming DTMF (including any pending in the digit buffer) as described in Step 5 above.

8. If the collect input is not valid with respect to the grammar or an interdigittimeout timer expires, the MS terminates collection execution and reports execution status to the dialog container with a <collectinfo> (see Section 4.3.2.3) where the termmode attribute is set to nomatch.

9. If the collect input completes the grammar or if a termtimeout timer expires, then the MS terminates collection execution and reports execution status to the dialog container with

<collectinfo> (see Section 4.3.2.3) where the termmode attribute is set to match.

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 MS MUST support both HTTP [RFC2616] and HTTPS [RFC2818] schemes and the MS MAY support other schemes. If the URI scheme is unsupported, the MS sends a <response> with a 420 status code (Section 4.5). If the resource cannot be retrieved within the timeout interval, the MS sends a <response> with a 409 status code. If the grammar format is not supported, the MS sends a <response> with a 424 status code. The attribute is optional. There is no default value.

type: identifies the preferred type of the grammar document identified by the src attribute. A valid value is a MIME media type (see Section 4.6.10). If the URI scheme used in the src attribute defines a mechanism for establishing the authoritative MIME media type of the media resource, the value returned by that mechanism takes precedence over this attribute. 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 Speech Recognition Grammar Specification [SRGS] XML grammar format ("application/srgs+xml") and MS MAY support the Key Press Markup Language (KPML) [RFC4730] or other grammar formats. If the grammar format is not supported by the MS, then the MS sends a <response> with a 424 status code (Section 4.5).

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 specifies 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.

vadinitial: controls whether Voice Activity Detection (VAD) is used to initiate the recording operation. A valid value is a boolean (see Section 4.6.1). A value of true indicates the MS MUST initiate recording if the VAD detects voice on the configured inbound audio streams. A value of false indicates that the MS MUST NOT initiate recording using VAD. The attribute is optional. The default value is false.

vadfinal: controls whether VAD is used to terminate the recording operation. A valid value is a boolean (see Section 4.6.1). A value of true indicates the MS MUST terminate recording if the VAD detects a period of silence (whose duration is specified by the finalsilence attribute) on configured inbound audio streams. A value of false indicates that the MS MUST NOT terminate recording using VAD. The attribute is optional. The default value is false.

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' is to 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 the end of voice input. 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.

append: indicates whether recorded data is appended or not to a recording location if a resource already exists. A valid value is a boolean (see Section 4.6.1). A value of true indicates that recorded data is appended to the existing resource at a recording

location. A value of false indicates that recorded data is to overwrite the existing resource. The attribute is optional. The default value is false.

When a recording location is specified using the HTTP or HTTPS protocol, the recording operation SHOULD be performed using the HTTP GET and PUT methods, unless the HTTP server provides a special interface for recording uploads and appends (e.g., using POST). When the append attribute has the value false, the recording data is uploaded to the specified location using HTTP PUT and replaces any data at that location on the HTTP origin server. When append has the value true, the existing data (if any) is first downloaded from the specified location using HTTP GET, then the recording data is appended to the existing recording (note that this might require codec conversion and modification to the existing data), then the combined recording is uploaded to the specified location using HTTP PUT. HTTP errors are handled as described in [RFC2616].

When the recording location is specified using protocols other than HTTP or HTTPS, the mapping of the append operation onto the upload protocol scheme is implementation specific.

If either the vadinitial or vadfinal attribute is set to true and the MS does not support VAD, the MS sends a <response> with a 434 status code (Section 4.5).

The <record> element has the following child element (0 or more occurrences):

<media>: specifies the location and type of the media resource for uploading recorded data (see Section 4.3.1.5). The MS MUST support both HTTP [RFC2616] and HTTPS [RFC2818] schemes for uploading recorded data and the MS MAY support other schemes. The MS uploads recorded data to this resource as soon as possible after recording is complete. The element is optional.

If multiple <media> elements are specified, then media input is to be recorded in parallel to multiple resource locations.

If no <media> child element is specified, the MS MUST record media input but the recording location and the recording format are implementation specific (e.g., the MS records audio in the WAV format to a local disk accessible by HTTP). The recording location and format are reported in <recordinfo> (Section 4.3.2.4) when the dialog terminates. The recording MUST be available from this location until the connection or conference associated with the dialog on the MS terminates.

If the MS does not support the configuration required for recording from the input media streams to one or more <media> elements and a more specific error code is not defined for its child elements, the MS sends a <response> with a 423 status code (Section 4.5).

Note that an MS MAY support uploading recorded data to recording locations at the same time the recording operation takes place. Such implementations need to be aware of the requirements of certain recording formats (e.g., WAV) for metadata at the beginning of the uploaded file, that the finalsilence interval is not part of the recording and how these requirements interact with the URI scheme.

The MS has the following execution model for recording after initialization:

1. If an error occurs during execution (e.g., authentication or communication error when trying to upload to a recording location), then the MS terminates record execution and reports the error to the dialog container (see Section 4.3). The MS MAY report data recorded before the error 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 the MS activates the record execution. Otherwise, the MS activates it after the completion of prompt playback.

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

4. The MS activates a timer with the duration of the value of the timeout attribute. If the timer expires before the recording operation begins, then the MS terminates the recording execution and reports the status 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 operation 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 the MS terminates recording execution and reports the execution status to the dialog container with <recordinfo> (see Section 4.3.2.4) where the termmode attribute set to maxtime.

7. During the record operation input, media streams are recording to a location and format specified in one or more <media> child elements. If no <media> child element is specified, the MS records input to an implementation-specific location and format.

8. If the dtmfterm attribute has the value true and DTMF input is detected during the record operation, then the MS terminates recording and its 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 MS terminates the recording operation 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.

For example, a request to record audio and video input to separate locations:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
<dialogstart connectionid="c1">
  <dialog>
   <record maxtime="30s" vadinitial="false" vadfinal="false">
    <media type="audio/x-wav"
        loc="http://www.example.com/upload/audio.wav"/>
    <media type="video/3gpp;codecs='s263'"
        loc="http://www.example.com/upload/video.3gp"/>
   </record>
  </dialog>
 </dialogstart>
</mscivr>
```

When the <record> element is executed, it immediately begins recording of the audio and video (since vadinitial is false) where the destination locations are specified in the <media> child elements. Recording is completed when the duration reaches 30s or the connection is terminated.

4.3.1.5. <media>

The <media> element specifies a media resource to playback from (see Section 4.3.1.1) or record to (see Section 4.3.1.4). In the playback case, the resource is retrieved and in the recording case, recording data is uploaded to the resource location.

A <media> element has the following attributes:

loc: specifies the location of the media resource. A valid value is a URI (see Section 4.6.9). The MS MUST support both HTTP [RFC2616] and HTTPS [RFC2818] schemes and the MS MAY support other schemes. If the URI scheme is not supported by the MS, the MS sends a <response> with a 420 status code (Section 4.5). If the resource is to be retrieved but the MS cannot retrieve it within the timeout interval, the MS sends a <response> with a 409 status code. If the format of the media resource is not supported, the MS sends a <response> with a 429 status code. The attribute is mandatory.

type: specifies the type of the media resource indicated in the loc attribute. A valid value is a MIME media type (see Section 4.6.10) that, depending on its definition, can include additional parameters (e.g., [RFC4281]). If the URI scheme used in the loc attribute defines a mechanism for establishing the authoratitive MIME media type of the media resource, the value returned by that mechanism takes precedence over this attribute. If additional media parameters are specified, the MS MUST use them to determine media processing. For example, [RFC4281] defines a 'codec' parameter for media types like video/3gpp that would determine which media streams are played or recorded. 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 4.6.8). 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 playback. 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 (or the clipEnd offset), no media is played. See 'clipBegin' in SMIL [W3C.REC-SMIL2-20051213] for further information. The attribute is optional. The default value is 0s.

clipEnd: offset from start of media resource to end playback. 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 clipEnd offset is after the end of media, then the media is played to the end. If clipBegin is after clipEnd, then no media is played. See 'clipEnd' in SMIL [W3C.REC-SMIL2-20051213] for further information. The attribute is optional. There is no default value.

The fetchtimeout, soundLevel, clipBegin, and clipEnd attributes are only relevant in the playback use case. The MS ignores these attributes when using the <media> for recording.

The <media> element has no children.

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 a 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 information about record execution (Section 4.3.1.4).

The <recordinfo> element has the following attributes:

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

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.

The <recordinfo> element has the following child element (0 or more occurrences):

<mediainfo>: indicates information about a recorded media resource (see Section 4.3.2.4.1). The element is optional.

When the record operation is successful, the MS MUST specify a <mediainfo> element for each recording location. For example, if the <record> element contained three <media> child elements, then the <recordinfo> would contain three <mediainfo> child elements.

4.3.2.4.1. <mediainfo>

The <mediainfo> element reports information about a recorded media resource.

The <mediainfo> element has the following attributes:

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

type: indicates the format of the media resource. A valid value is a MIME media type (see Section 4.6.10). The attribute is mandatory.

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

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 can then use this information to create request elements using supported capabilities and, in the case of codecs, to negotiate an appropriate SDP for a User Agent's connection. Second, auditing enables discovery of the existence and status of dialogs currently managed by the package on the MS. This could be used when one AS takes over management of the dialogs if the AS that initiated the dialogs fails or is no longer available (see Security Considerations described in Section 7).

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 languages, 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. The MS sends a response with a 406 status code (Section 4.5) if the specified dialog identifier is invalid. 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 an <audit> request, it MUST reply with an <auditresponse> element (Section 4.4.2), which includes a mandatory attribute describing the status in terms of a numeric code. Response status codes are defined in Section 4.5. If the request is successful, the <auditresponse> contains (depending on attribute 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. If the MS is not able to process the request and carry out the audit operation, the audit request has failed and the MS MUST indicate the class of failure using an appropriate 4xx response code. Unless an error response code is specified for a class of error within this section, implementations follow Section 4.5 in determining the appropriate status code for the response.

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 an <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.

desclang: specifies the language used in the value of the reason attribute. A valid value is a language identifier (Section 4.6.11). The attribute is optional. If not specified, the value of the desclang attribute on <mscivr> (Section 4.1) applies.

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 an <audit> request requesting capabilities and dialogs information:

```
   <mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
    <auditresponse status="200">
     <capabilities>
      <dialoglanguages>
       <mimetype>application/voicexml+xml</mimetype>
      </dialoglanguages>
      <grammartypes/>
      <recordtypes>
       <mimetype>audio/x-wav</mimetype>
       <mimetype>video/3gpp</mimetype>
      </recordtypes>
      <prompttypes>
       <mimetype>audio/x-wav</mimetype>
       <mimetype>video/3gpp</mimetype>
      </prompttypes>
      <variables>
       <variabletype type="date" desc="value formatted as YYYYMMDD">
        <format desc="month year day">mdy</format>
        <format desc="year month day">ymd</format>
        <format desc="day month year">dmy</format>
        <format desc="day month">dm</format>
       </variabletype>
      </variables>
      <maxpreparedduration>600s</maxpreparedduration>
      <maxrecordduration>1800s</maxrecordduration>
      <codecs>
       <codec name="video">
        <subtype>H263</subtype>
       </codec>
       <codec name="video">
        <subtype>H264</subtype>
       </codec>
       <codec name="audio">
        <subtype>PCMU</subtype>
       </codec>
       <codec name="audio">
        <subtype>PCMA</subtype>
       </codec>
       <codec name="audio">
        <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 name="audio">
         <subtype>PCMA</subtype>
        </codec>
        <codec name="audio">
         <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 occurrences):

<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 name="audio">
   <subtype>PCMA</subtype>
  </codec>
  <codec name="audio">
   <subtype>telephone-event</subtype>
  </codec>
</codecs>
```

4.4.2.1.1. <codec>

The <codec> element describes a codec on the MS. The element is modeled on the <codec> element in the XCON conference information data model [XCON-DATA-MODEL] but allows addition information (e.g., rate, speed, etc.) to be specified.

The <codec> element has the following attributes:

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

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

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

<params>: element (Section 4.2.6) describing additional information about the codec. This package is agnostic to the names and values of the codec parameters supported by an implementation. The element is optional.

For example, a fragment with a <codec> element describing the H263 video codec:

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

<dialoglanguages>: element (Section 4.4.2.2.1) describing additional dialog languages supported by the MS. 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 <media> (Section 4.3.1.5) format types supported for <record> (Section 4.3.1.4). The element is mandatory.

<prompttypes>: element (Section 4.4.2.2.4) describing supported <media> (Section 4.3.1.5) format types for playback within a <prompt> (Section 4.3.1.1). The element is mandatory.

<variables>: element (Section 4.4.2.2.5) describing supported types and formats for the <variable> element (Section 4.3.1.1.1). The element is mandatory.

<maxpreparedduration>: element (Section 4.4.2.2.6) describing the supported maximum duration for a prepared dialog following a <dialogprepare> (Section 4.2.1) request. The element is mandatory.

<maxrecordduration>: element (Section 4.4.2.2.7) describing the supported maximum duration for a recording <record> (Section 4.3.1.4) request. 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>
   <dialoglanguages>
    <mimetype>application/voicexml+xml</mimetype>
   </dialoglanguages>
   <grammartypes/>
   <recordtypes>
    <mimetype>audio/x-wav</mimetype>
    <mimetype>video/3gpp</mimetype>
   </recordtypes>
   <prompttypes>
    <mimetype>audio/x-wav</mimetype>
    <mimetype>video/3gpp</mimetype>
   </prompttypes>
   <variables/>
   <maxpreparedduration>30s</maxpreparedduration>
   <maxrecordduration>60s</maxrecordduration>
   <codecs>
    <codec name="video">
     <subtype>H263</subtype>
    </codec>
    <codec name="video">
     <subtype>H264</subtype>
    </codec>
    <codec name="audio">
     <subtype>PCMU</subtype>
    </codec>
    <codec name="audio">
     <subtype>PCMA</subtype>
    </codec>
    <codec name="audio">
     <subtype>telephone-event</subtype>
    </codec>
   </codecs>
  </capabilities>
```

4.4.2.2.1. <dialoglanguages>

The <dialoglanguages> element provides information about additional dialog languages supported by the package. Dialog languages are identified by their associated MIME media types. The MS MUST NOT include the mandatory dialog language for this package (Section 4.3).

The <dialoglanguages> element has no attributes.

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

<mimetype>: element whose content model describes a MIME media type (Section 4.6.10) associated with a supported dialog language. 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 NOT 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 (0 or more occurrences):

<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 media resource format types of <record> 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 occurrences):

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

4.4.2.2.4. <prompttypes>

The <prompttypes> element provides information about media resource format types of <prompt> supported by the package (Section 4.3.1.1).

The <prompttypes> element has no attributes.

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

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

4.4.2.2.5. <variables>

The <variables> element provides information about types and formats for the <variable> element (Section 4.3.1.1.1) supported by the package.

The <variables> element has no attributes.

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

<variabletype>: element describing the formats support for a given type (Section 4.4.2.2.5.1). The element is optional.

For example, a fragment describing support for <variable> with a "date" type according to the formats specified in Section 4.3.1.1.1.1.

```
<variables>
 <variabletype type="date" desc="value formatted as YYYYMMDD">
   <format desc="month year day">mdy</format>
   <format desc="year month day">ymd</format>
   <format desc="day month year">dmy</format>
   <format desc="day month">dm</format>
 </variabletype>
</variables>
```

4.4.2.2.5.1. <variabletype>

The <variabletype> element describes the formats supported for <variable> supported type.

The <variabletype> element has the following attributes:

type: indicates a supported value associated with the type attribute of the <variable> element. The attribute is mandatory.

desc: a string providing some textual description of the type and format. The attribute is optional.

desclang: specifies the language used in the value of the desc attribute. A valid value is a language identifier (Section 4.6.11). The attribute is optional. If not specified, the value of the desclang attribute on <mscivr> (Section 4.1) applies.

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

<format>: element with a desc attribute (optional description), desclang (optional language identifier for the description), and a content model describing a supported format in the <variable> format attribute. The element is optional.

4.4.2.2.6. <maxpreparedduration>

The <maxpreparedduration> element describes the maximum duration for a dialog to remain in the prepared state (Section 4.2) following a <dialogprepare> (Section 4.2.1) request.

The <maxpreparedduration> element has no attributes.

The <maxpreparedduration> element has a content model describing the maximum prepared dialog duration as a time designation (Section 4.6.7).

4.4.2.2.7. <maxrecordduration>

The <maxrecordduration> element describes the maximum recording duration for <record> Section 4.3.1.4) request supported by the MS.

The <maxrecordduration> element has no attributes.

The <maxrecordduration> element has a content model describing the maximum duration of recording as a time designation (Section 4.6.7).

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 occurrences):

<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, and started. The attribute is mandatory.

connectionid: string identifying the SIP dialog connection associated with the dialog (see Appendix A.1 of [RFC6230]). The attribute is optional. There is no default value.

conferenceid: string identifying the conference associated with the dialog (see Appendix A.1 of [RFC6230]). 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 that is using PCMU and telephony-event audio codecs:

```
<dialogaudit dialogid="1234" state="started" conferenceid="conf1">
 <codecs>
  <codec name="audio">
   <subtype>PCMU</subtype>
  </codec>
  <codec name="audio">
   <subtype>telephone-event</subtype>
  </codec>
 </codecs>
</dialogaudit>
```

4.5. Response Status Codes

This section describes the response codes in Table 1 for the status attribute of dialog management <response> (Section 4.2.4) and audit <auditresponse> (Section 4.4.2) responses. The MS MUST support the status response codes defined here. All other valid but undefined values are reserved for future use, where new status codes are assigned using the Standards Action process defined in [RFC5226]. The AS MUST treat any responses it does not recognize as being equivalent to the x00 response code for all classes. For example, if

an AS receives an unrecognized response code of 499, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 (Syntax error) response code.

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

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

4xx failure responses can be grouped into three classes: failure due to a syntax error in the request (400); failure due to an error executing the request on the MS (405-419); and failure due to the request requiring a capability not supported by the MS (420-439).

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

Code	Summary	Description	Informational: AS Possible Recovery Action
200	OK	request has succeeded.	
400	Syntax error	request is syntactically invalid: it is not valid with respect to the XML schema specified in Section 5 or it violates a co-occurrence constraint for a request element defined in Section 4.	Change the request so that it is syntactically valid.
405	dialogid already exists	request uses a dialogid identifier for a new dialog that is already used by another dialog on the MS (see Section 4.2).	Send a request for a new dialog without specifying the dialogid and let the MS generate a unique dialogid in the response.
406	dialogid does not exist	request uses a dialogid identifier for an dialog that does not exist on the MS (see Section 4.2).	Send an <audit> request (Section 4.4.1) requesting the list of dialog identifiers already used by the MS and then use one of the listed dialog identifiers.
407	connectionid does not exist	request uses a connectionid identifier for a connection that does not exist on the MS.	Use another method to determine which connections are available on the MS.
408	conferenceid does not exist	request uses a conferenceid identifier for a conference that does not exist on the MS.	Use another method to determine which conferences are available on the MS.

409	Resource cannot be retrieved	request uses a URI to reference an external resource (e.g., dialog, media, or grammar) that cannot be retrieved within the timeout interval.	Check that the resource URI is valid, can be reached from the MS, and that the appropriate authentication is used.
410	Dialog execution canceled	request to prepare or start a dialog that has been terminated by a <dialogterminate/> request (see Section 4.2).	
411	Incompatible stream configuration	request specifies a media stream configuration that is in conflict with itself, or the connection or conference capabilities (see Section 4.2.2).	Change the media stream configuration to match the capabilities of the connection or conference.
412	Media stream not available	request specifies an operation for which a media stream is not available. For example, playing a video media resource on an connection or conference without video streams.	Check the media stream capability of the connection or conference and use an operation that only uses these capabilities.
413	Control keys with same value	request contains a <control> element (Section 4.3.1.2) where some keys have the same value.	Use different keys for the different control operations.
419	Other execution error	requested operation cannot be executed by the MS.	
420	Unsupported URI scheme	request specifies a URI whose scheme is not supported by the MS.	Use a URI scheme that is supported.

421	Unsupported dialog language	request references an external dialog language not supported by the MS.	Send an <audit> request (Section 4.4.1) requesting the MS capabilities and then use one of the listed dialog languages.
422	Unsupported playback format	request references a media resource for playback whose format is not supported by the MS.	Send an <audit> request (Section 4.4.1) requesting the MS capabilities and then use one of the listed playback media formats.
423	Unsupported record format	request references a media resource for recording whose format is not supported by the MS.	Send an <audit> request (Section 4.4.1) requesting the MS capabilities and then use one of the listed record media formats.
424	Unsupported grammar format	request references a grammar whose format is not supported by the MS.	Send an <audit> request (Section 4.4.1) requesting the MS capabilities and then use one of the listed grammar types.
425	Unsupported variable configuration	request contains a prompt <variable> element (Section 4.3.1.1.1) not supported by the MS.	Send an <audit> request (Section 4.4.1) requesting the MS capabilities and then use one of the listed variable types.
426	Unsupported DTMF configuration	request contains a prompt <dtmf> element (Section 4.3.1.1.2) not supported by the MS.	

427	Unsupported parameter	request contains a <param> element (Section 4.2.6.1) not supported by the MS.	
428	Unsupported media stream configuration	request contains a <stream> element (Section 4.2.2.2) whose configuration is not supported by the MS.	
429	Unsupported playback configuration	request contains a <prompt> element (Section 4.3.1.1) that the MS is unable to play on the available output media streams.	
430	Unsupported record configuration	request contains a <record> element (Section 4.3.1.4) that the MS is unable to record with on the available input media streams.	
431	Unsupported foreign namespace attribute or element	request contains attributes or elements from another namespace that the MS does not support.	
432	Unsupported multiple dialog capability	request tries to start another dialog on the same conference or connection where a dialog is already running.	
433	Unsupported collect and record capability	request contains <collect> and <record> elements and the MS does support these operations simultaneously.	
434	Unsupported VAD capability	request contains a <record> element where Voice Activity Detection (VAD) is required, but the MS does not support VAD.	

435	Unsupported parallel playback	request contains a prompt <par> element whose configuration is not supported by the MS.	
439	Other unsupported capability	request requires another capability not supported by the MS.	

Table 1: Status Codes

4.6. Type Definitions

This section defines types referenced in attribute and element definitions.

4.6.1. Boolean

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

4.6.2. 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,...} as defined in Section 3.3.20 of [XMLSchema:Part2].

Implementation Note: It is RECOMMENDED that implementations at least support a maximum value of a 32-bit integer (2,147,483,647).

4.6.5. Positive Integer

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

Implementation Note: It is RECOMMENDED that implementations at least support a maximum value of a 32-bit integer (2,147,483,647).

4.6.6. String

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

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. MIME Media Type

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

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

parameter = parameter-name "=" value

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

4.6.11. 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 [RFC5646] code and matched according to [RFC4647].

4.6.12. DateTime

A string formatted 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 is normative.

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 Appendix A.1 of the Control Framework [RFC6230]. 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 (20110104)

   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="http://www.w3.org/2001/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 (non-mixed) component types to
    allow attributes from other namespaces.
   </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence/>
  <xsd:anyAttribute namespace="##other" processContents="lax" />
 </xsd:complexType>


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

```
  TOP LEVEL ELEMENT: 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:attribute name="desclang" type="xsd:language"
     default="i-default" />
   </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:element ref="params" 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"/>
    <xsd:attribute name="maxage" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="maxstale" type="xsd:nonNegativeInteger"/>
    <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="maxage" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="maxstale" type="xsd:nonNegativeInteger"/>
    <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:sequence>
    <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="dialogid"
     type="dialogid.datatype" use="required" />
    <xsd:attribute name="immediate"
     type="xsd:boolean" 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:sequence>
    <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="status" type="status.datatype"
     use="required" />
    <xsd:attribute name="reason" type="xsd:string" />
    <xsd:attribute name="desclang" type="xsd:language"/>
    <xsd: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:nonNegativeInteger" use="required" />
    <xsd:attribute name="reason" type="xsd:string" />
    <xsd:attribute name="desclang" type="xsd:language"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

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


 <!--  dtmfnotify-->

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

 <!--  mediainfo -->

 <xsd:complexType name="mediainfoType">
  <xsd:complexContent>
   <xsd:extension base="Tcore">
 <xsd:sequence>
    <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="loc" type="xsd:anyURI"
     use="required" />
    <xsd:attribute name="type" type="mime.datatype"
    use="required"/>
    <xsd:attribute name="size"
     type="xsd:nonNegativeInteger" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
 <xsd:element name="mediainfo" type="mediainfoType" />
```

```
 <!--  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:sequence>
    <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <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:attribute name="name" type="xsd:string" use="required" />
  <xsd:attribute name="type" type="mime.datatype" default="text/plain"/>
     <xsd:attribute name="encoding" type="xsd:string"/>
  <xsd: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:sequence>
     <xsd:element ref="region" minOccurs="0"
      maxOccurs="1" />
    <xsd:element ref="priority" minOccurs="0"
      maxOccurs="1" />
     <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="media" type="media.datatype"
     use="required" />
    <xsd:attribute name="label" type="label.datatype" />
    <xsd:attribute name="direction"
     type="direction.datatype" default="sendrecv" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

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

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

```
 <!--  priority  -->
 <xsd:simpleType name="priorityType">
  <xsd:restriction base="xsd:positiveInteger" />
 </xsd:simpleType>

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


<!-- 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:attribute name="repeatUntilComplete"
     type="xsd:boolean" default="false"/>
   </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:element ref="par" />
```

```
     <xsd:any namespace="##other"
      processContents="lax" />
    </xsd:choice>
    <xsd:attribute ref="xml:base" />
    <xsd:attribute name="bargein" type="xsd:boolean"
     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:sequence>
    <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="loc" 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:attribute name="clipEnd"
     type="timedesignation.datatype"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

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

 <!--  variable -->

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

```
    <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="variableT" />

 <!-- dtmf -->

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


 <!--  par  -->

 <xsd:complexType name="parType">
  <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:element ref="seq" />
     <xsd:any namespace="##other"
```

```
      processContents="lax" />
    </xsd:choice>
    <xsd:attribute name="endsync" type="endsync.datatype"
    default="last"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:element name="par" type="parType" />

 <!--  seq  -->

 <xsd:complexType name="seqType">
  <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:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:element name="seq" type="seqType" />

 <!-- control -->

 <xsd:complexType name="controlType">
  <xsd:complexContent>
   <xsd:extension base="Tcore">
 <xsd:sequence>
    <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <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="xsd:boolean" 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:sequence>
     <xsd:element ref="media" minOccurs="0"
      maxOccurs="unbounded" />
     <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
    <xsd:attribute name="timeout"
     type="timedesignation.datatype" default="5s" />
    <xsd:attribute name="beep" type="xsd:boolean"
     default="false" />
    <xsd:attribute name="vadinitial"
     type="xsd:boolean" default="false" />
    <xsd:attribute name="vadfinal"
     type="xsd:boolean" default="false" />
    <xsd:attribute name="dtmfterm"
     type="xsd:boolean" default="true" />
    <xsd:attribute name="maxtime"
     type="timedesignation.datatype" default="15s" />
    <xsd:attribute name="finalsilence"
     type="timedesignation.datatype" default="5s" />
```

```
    <xsd:attribute name="append" type="xsd:boolean"
     default="false" />
   </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:sequence>
    <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
   <xsd:attribute name="capabilities"
     type="xsd:boolean" default="true" />
   <xsd:attribute name="dialogs"
     type="xsd:boolean" 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:attribute name="desclang" type="xsd:language"/>
   </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:element ref="params" minOccurs="0"
      maxOccurs="1" />
     <xsd:any namespace="##other" minOccurs="0"
      maxOccurs="unbounded" processContents="lax" />
    </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string"
     use="required" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

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

 <!-- subtype -->

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

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


 <!-- 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="dialoglanguages" minOccurs="1"
      maxOccurs="1" />
     <xsd:element ref="grammartypes" minOccurs="1"
      maxOccurs="1" />
     <xsd:element ref="recordtypes" minOccurs="1"
      maxOccurs="1" />
     <xsd:element ref="prompttypes" minOccurs="1"
      maxOccurs="1" />
     <xsd:element ref="variables" minOccurs="1"
      maxOccurs="1" />
      <xsd:element ref="maxpreparedduration" minOccurs="1"
      maxOccurs="1" />
      <xsd:element ref="maxrecordduration" 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" />
```

```
 <!-- dialoglanguages -->

 <xsd:complexType name="dialoglanguagesType">
  <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="dialoglanguages" type="dialoglanguagesType" />

 <!-- grammartypes -->

 <xsd:complexType name="grammartypesType">
  <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="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" />


    <!-- prompttypes -->

 <xsd:complexType name="prompttypesType">
  <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="prompttypes" type="prompttypesType" />

<!--  variables -->

 <xsd:complexType name="variablesType">
  <xsd:complexContent>
   <xsd:extension base="Tcore">
    <xsd:sequence>
     <xsd:element ref="variabletype" 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="variables" type="variablesType" />

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

```
    <xsd:attribute name="type" type="xsd:string" use="required" />
    <xsd:attribute name="desc" type="xsd:string"/>
    <xsd:attribute name="desclang" type="xsd:language"/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>

 <xsd:element name="variabletype" type="variabletypeType" />

<!--  format  -->
<!--  doesn't extend tCore since its content model is mixed -->
 <xsd:complexType name="formatType" mixed="true">
  <xsd:sequence>
   <xsd:any namespace="##other" minOccurs="0"
    maxOccurs="unbounded" processContents="lax" />
  </xsd:sequence>
  <xsd:attribute name="desc" type="xsd:string" />
  <xsd:attribute name="desclang" type="xsd:language"/>
  <xsd:anyAttribute namespace="##other" processContents="lax" />
 </xsd:complexType>

 <xsd:element name="format" type="formatType" />

<!--  maxpreparedduration -->

 <xsd:element name="maxpreparedduration"
type="timedesignation.datatype"/>


<!--  maxrecordduration -->

 <xsd:element name="maxrecordduration"
type="timedesignation.datatype"/>


 <!-- 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="gender.datatype">
  <xsd:restriction base="xsd:NMTOKEN">
   <xsd:enumeration value="female" />
   <xsd:enumeration value="male" />
  </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="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:simpleType name="endsync.datatype">
  <xsd:restriction base="xsd:NMTOKEN">
   <xsd:enumeration value="first" />
   <xsd:enumeration value="last" />
  </xsd:restriction>
 </xsd:simpleType>

</xsd:schema>
```

6. Examples

This section provides examples of the IVR Control Package.

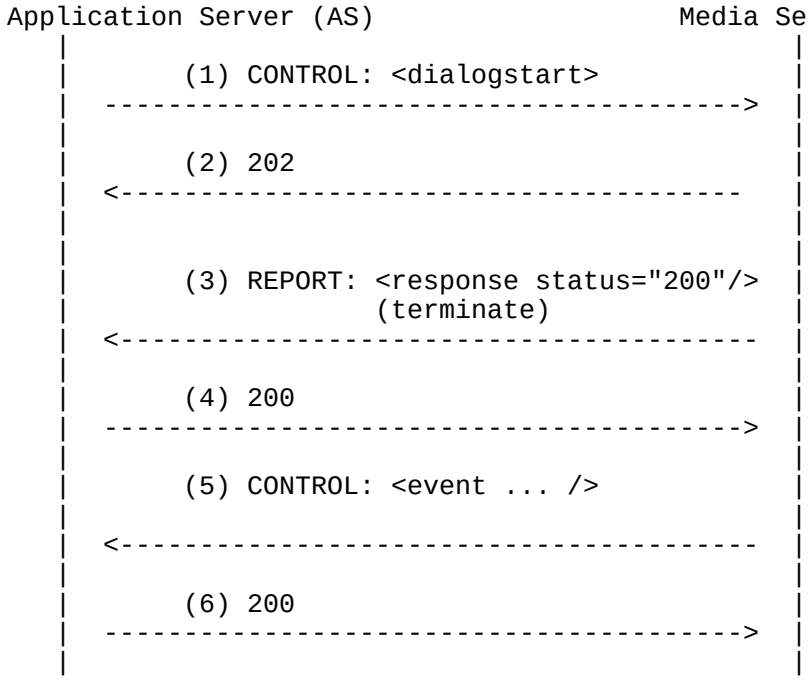
6.1. AS-MS Dialog Interaction Examples

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

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.

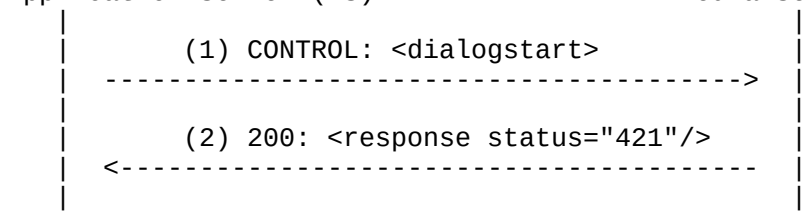
6.1.1. Starting an IVR Dialog

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



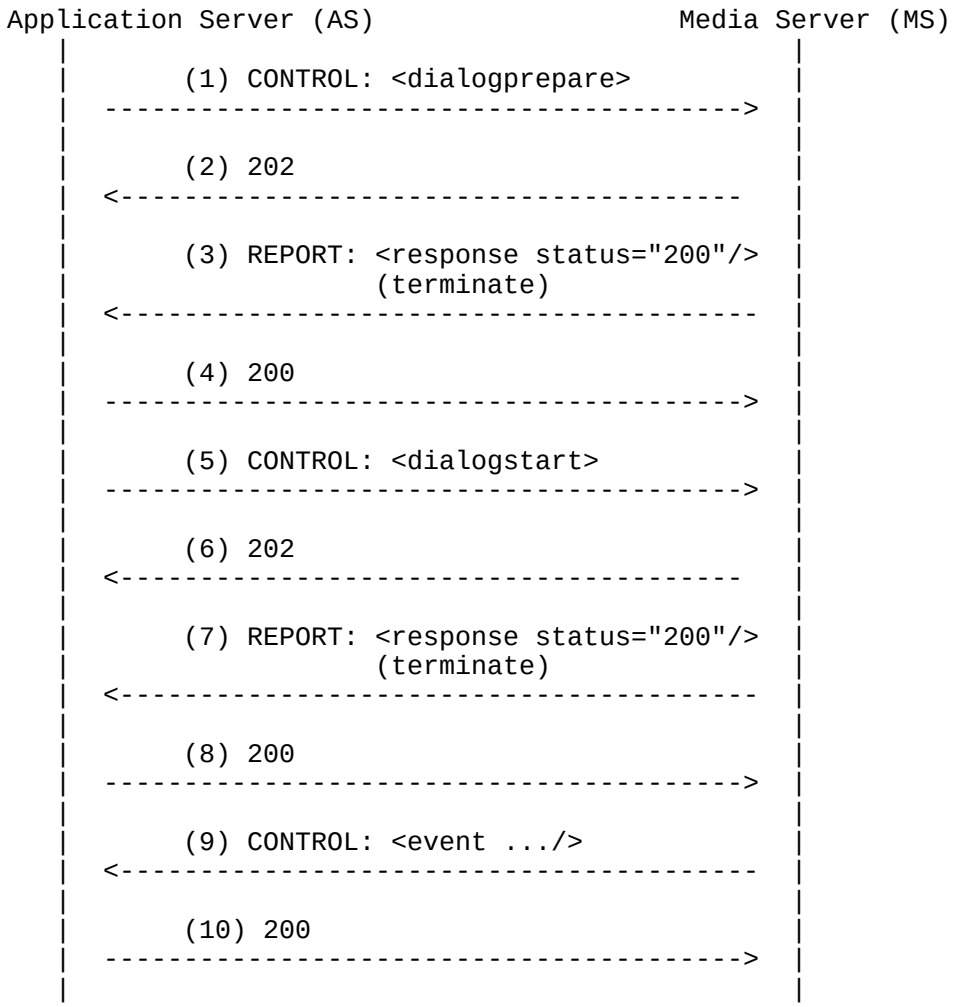
6.1.2. IVR Dialog Fails to Start

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



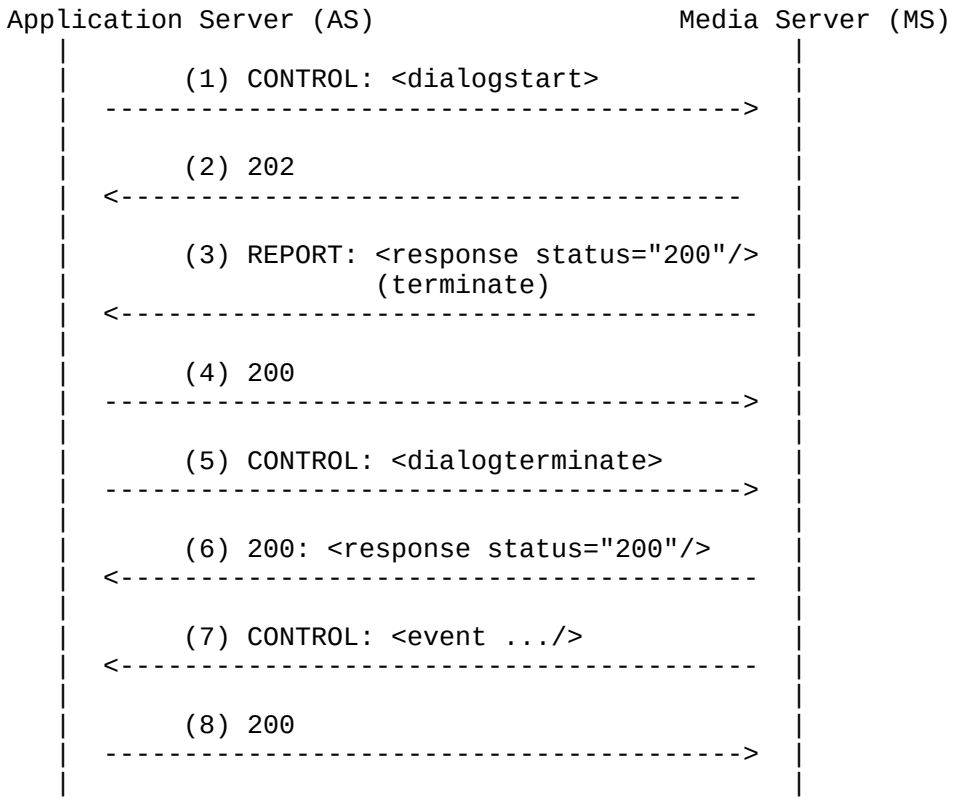
6.1.3. Preparing and Starting an IVR Dialog

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



6.1.4. Terminating a Dialog

An IVR dialog is started successfully, and then terminated by the AS. The dialogexit event is sent 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 where the dialog repeatCount is set to 2.

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
<dialogprepare>
   <dialog repeatCount="2">
    <prompt>
      <media loc="http://www.example.com/media/Number_09.wav"/>
      <media loc="http://www.example.com/media/Number_11.wav"/>
    </prompt>
   </dialog>
</dialogprepare>
</mscivr>
```

If the dialog is prepared successfully, a <response> is returned with status 200 and a dialog identifier assigned by the MS:

```
<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" duration="24000"/>
    </dialogexit>
 </event>
</mscivr>
```

6.2.2. Prompt and Collect

In this example, a prompt is played and then the MS waits for 30s 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 loc="http://www.example.com/prompt1.wav"/>
  </prompt>
  <collect timeout="30s" maxdigits="2"/>
 </dialog>
 </dialogstart>
</mscivr>
```

If no user input is collected within 30s, then the following notification event would be returned:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="vxi81">
    <dialogexit status="1" >
       <promptinfo termmode="completed" duration="4000"/>
       <collectinfo termmode="noinput"/>
    </dialogexit>
 </event>
</mscivr>
```

The collect operation can be specified without a prompt. Here the MS just waits for DTMF input from the user (the maxdigits attribute of <collect> defaults to 5):

```
<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>
```

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 loc="http://www.example.com/prompt1.wav"/>
   </prompt>
   <collect cleardigitbuffer="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="400" dialogid="vxi82"
  reason="repeatCount attribute 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 loc="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 termmode="dtmf">
    <mediainfo type="audio/x-wav"
      loc="http://www.example.com/recording1.wav"/>
   </recordinfo>
  </dialogexit>
 </event>
</mscivr>
```

6.2.4. Runtime Controls

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

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="7HDY839:HJKSkyHS~HUwkuh7ns">
 <dialog>
  <prompt bargein="true">
   <media loc="http://www.example.com/prompt1.wav"/>
  </prompt>
  <control ffkey="5" rwkey="6" speedupkey="3"
           speeddnkey="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"/>
       <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>
       <collectinfo termmode="match" dtmf="14"/>
    </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 connectionid="7HDY839:HJKSkyHS">
  <dialog repeatCount="0">
   <collect maxdigits="2"/>
  </dialog>
  <subscribe>
   <dtmfsub matchmode="collect"/>
  </subscribe>
 </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 dialog would continue to loop until terminated (or an error occurred).

6.2.6. Dialog Repetition until DTMF Collection Complete

This example is a prompt and collect dialog to collect the PIN from the user. The repeatUntilComplete attribute in the <dialog> is set

to true in this case so that when the grammar collection is complete, the MS automatically terminates the dialog repeat cycle and reports the results in a <dialogexit> event.

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="7HDY839:HJKSkyHS">
  <dialog repeatCount="3" repeatUntilComplete="true">
   <prompt bargein="true">
     <media loc="http://example.com/please_enter_your_pin.vox"/>
   </prompt>
   <collect maxdigits="4"/>
  </dialog>
 </dialogstart>
</mscivr>
```

If the user barges in on the prompt and <collect> receives DTMF input matching the grammar, the dialog cycle is considered complete and the MS returns the following:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <event dialogid="vxi81">
   <dialogexit status="1">
     <promptinfo duration="3654" termmode="bargein"/>
     <collectinfo dtmf="1234" termmode="match"/>
   </dialogexit>
 </event>
</mscivr>
```

If no user input was provided, or the input did not match the grammar, the dialog would loop for a maximum of 3 times.

6.3. Other Dialog Languages

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"
              connectionid="7HDY839:HJKSkyHS"
              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 language, then the response would have the status code 421 (Section 4.5). However, if it does support the VoiceXML dialog language, it would respond with a 200 status, activate the VoiceXML dialog, and make the <params> available to the VoiceXML script as described in Section 9.

When the VoiceXML dialog exits, exit namelist parameters are specified using <params> in 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>
```

6.4. Foreign Namespace Attributes and Elements

An MS can support attributes and elements from foreign namespaces within the <mscivr> element. For example, the MS could support a <listen> element (in a foreign namespace) for speech recognition by analogy to how <collect> supports DTMF collection.

In the following example, a prompt and collect request is extended with a <listen> element:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr"
 xmlns:ex="http://www.example.com/mediactrl/extensions/1">
 <dialogstart connectionid="7HDY839:HJKSkyHS~HUwkuh7ns">
 <dialog>
  <prompt>
   <media loc="http://www.example.com/prompt1.wav"/>
  </prompt>
  <collect timeout="30s" maxdigits="4"/>
  <ex:listen maxtimeout="30s" >
    <ex:grammar src="http://example.org/pin.grxml"/>
  </ex:listen>
 </dialog>
 </dialogstart>
</mscivr>
```

In the <mscivr> root element, the xmlns:ex attribute declares that "ex" is associated with the foreign namespace URI "http://www.example.com/mediactrl/extensions/1". The <ex:listen>,

its attributes, and child elements are associated with this namespace. This <listen> could be defined so that it activates an SRGS grammar and listens for user input matching the grammar in a similar manner to DTMF collection.

If an MS receives this request but does not support the <listen> element, then it would send a 431 response:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <response status="431" dialogid="d560"
  reason="unsupported foreign listen element"/>
</mscivr>
```

If the MS does support this foreign element, it would send a 200 response and start the dialog with speech recognition. When the dialog exits, it provides information about the <listen> execution within <dialogexit>, again using elements in a foreign namespace such as <listeninfo> below:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr"
 xmlns:ex="http://www.example.com/mediactrl/extensions/1">
 <event dialogid="d560">
   <dialogexit status="1">
    <ex:listeninfo speech="1 2 3 4" termmode="match"/>
   </dialogexit>
  </event>
</mscivr>
```

Note that in reply the AS sends a Control Framework 200 response even though the notification event contains an element in a foreign namespace that it might not understand.

7. Security Considerations

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

Implementations of this Control Package MUST address security, confidentiality, and integrity of messages transported over the Control Channel as described in Section 12 of "Media Control Channel Framework" [RFC6230], including Transport Level Protection, Control Channel Policy Management, and Session Establishment. In addition, implementations MUST address security, confidentiality, and integrity of User Agent sessions with the MS, both in terms of SIP signaling and associated RTP media flow; see [RFC6230] for further details on this topic. Finally, implementations MUST address security,

confidentiality, and integrity of sessions where, following a URI scheme, an MS uploads recordings or retrieves documents and resources (e.g., fetching a grammar document from a web server using HTTPS).

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

Denial of Service: An attacker could insert a request message into the transport stream causing specific dialogs on the MS to be terminated immediately. For example, <dialogterminate dialogid="XXXX" immediate="true">, where the value of "XXXX" could be guessed or discovered by auditing active dialogs on the MS using an <audit> request. Likewise, an attacker could impersonate the MS and insert error responses into the transport stream so denying the AS access to package capabilities.

Resource Exhaustion: An attacker could insert into the Control Channel new request messages (or modify existing ones) with, for instance, <dialogprepare> elements with a very long fetchtimeout attribute and a bogus source URL. At some point, this will exhaust the number of connections that the MS is able to make.

Phishing: An attacker with access to the Control Channel could modify the "loc" attribute of the <media> element in a dialog to point to some other audio file that had different information from the original. This modified file could include a different phone number for people to call if they want more information or need to provide additional information (such as governmental, corporate, or financial information).

Data Theft: An attacker could modify a <record> element in the Control Channel so as to add a new recording location:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart>
  <dialog>
   <record>
      <media type="audio/x-wav" loc="(Good URI)"/>
      <media type="audio/x-wav" loc="(Attacker's URI)"/>
   </record>
  </dialog>
 </dialogstart>
</mscivr>
```

The recorded data would be uploaded to two locations indicated by the "{Good URI}" and the "{Attacker's URI}". This allows the attacker to steal the recorded audio (which could include sensitive or confidential information) without the originator of the request necessarily being aware of the theft.

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

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

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

Notifications: The MS MUST only send notification events for a dialog using the same Control Channel as it received the request creating the dialog.

Auditing: The MS MUST only provide audit information about dialogs that have been created on the same Control Channel as the one upon the <audit> request is sent.

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

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

There can also be valid reasons for 'channel handover' including high availability support or where one AS needs to take over management of dialogs after the AS that created them has failed. This could be achieved by the Control Channels using the same channel identifier, one after another. For example, assume a channel is created with the identifier 'cfw1234' and the channel is used to create dialogs on the MS. This channel (and associated SIP dialog) then terminates due to a failure on the AS. As permitted by the Control Framework, the channel identifier 'cfw1234' could then be reused so that another channel is created with the same identifier 'cfw1234', allowing it to 'take over' management of the dialogs on the MS. Again, the implementation needs to understand the full implications and carefully weight them before accepting these reasons as valid. If the reasons are not valid for their particular circumstances, the MS uses the appropriate SIP mechanisms to prevent session establishment when the same channel identifier is used in setting up another Control Channel (see Section 4 of [RFC6230]).

8. IANA Considerations

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

IANA has further created a new registry for IVR prompt variable types.

8.1. Control Package Registration

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

Package Name: msc-ivr/1.0
Published Specification(s): RFC 6231
Person & email address to contact for further information:
IETF MEDIACTRL working group (mediactrl@ietf.org),
Scott McGlashan (smcg.stds01@mcglashan.org).

8.2. URN Sub-Namespace Registration

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

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

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
   <title>Media Control Channel Framework IVR
          Package attributes</title>
  </head>
  <body>
   <h1>Namespace for Media Control Channel
       Framework IVR Package attributes</h1>
   <h2>urn:ietf:params:xml:ns:msc-ivr</h2>
     <p>See <a href="http://www.rfc-editor.org/rfc/rfc6231.txt">
     RFC 6231</a>.</p>
  </body>
 </html>
END
```

8.3. XML Schema Registration

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

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

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

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

Type name: application

Subtype name: msc-ivr+xml

Required parameters: (none)

Optional parameters: charset
Indicates the character encoding of enclosed XML. Default is UTF-8.

Encoding considerations: Uses XML, which can employ 8-bit characters, depending on the character encoding used. See RFC 3023 [RFC3023], Section 3.2.

Security considerations: No known security considerations outside of those provided by the Media Control Channel Framework IVR Package.

Interoperability considerations: This content type provides constructs for the Media Control Channel Framework IVR package.

Published specification: RFC 6231

Applications that use this media type: Implementations of the Media Control Channel Framework IVR package.

Additional information:
Magic number(s): (none)
File extension(s): (none)
Macintosh file type code(s): (none)

Person & email address to contact for further information:
Scott McGlashan <smcg.stds01@mcglashan.org>

Intended usage: LIMITED USE

Author/Change controller: The IETF

Other information: None.

8.5. IVR Prompt Variable Type Registration Information

This specification establishes an IVR Prompt Variable Type registry for Control Packages and initiates its population as follows. New entries in this registry must be published in an RFC (either as an IETF submission or RFC Editor submission), using the IANA policy [RFC5226] "RFC Required".

Variable Type	Control Package	Reference
date	msc-ivr/1.0	[RFC6231]
time	msc-ivr/1.0	[RFC6231]
digits	msc-ivr/1.0	[RFC6231]

The following information MUST be provided in an RFC in order to register a new prompt variable type:

Variable Type: The value for the <variable> type attribute (Section 4.3.1.1.1). The RFC MUST specify permitted values (if any) for the format attribute of <variable> and how the value attribute is rendered for different values of the format attribute. The RFC MUST NOT weaken but MAY strengthen the valid values of <variable> attributes defined in Section 4.3.1.1.1 of this specification.

Reference: The RFC number in which the variable type is registered.

Control Package: The Control Package associated with the IVR variable type.

Person & address to contact for further information:

9. Using VoiceXML as a Dialog Language

The IVR Control Package allows, but does not require, the MS to support other dialog languages by referencing an external dialog document. This section provides MS implementations that support the VoiceXML dialog language ([VXML20], [VXML21], [VXML30]) with additional details about using these dialogs in this package. This section is normative for an MS that supports the VoiceXML dialog language.

This section covers preparing (Section 9.1), starting (Section 9.2), terminating (Section 9.3), and exiting (Section 9.4) VoiceXML dialogs as well as handling VoiceXML call transfer (Section 9.5).

9.1. Preparing a VoiceXML Dialog

A VoiceXML dialog is prepared by sending the MS a request containing a <dialogprepare> element (Section 4.2.1). The type attribute is set to "application/voicexml+xml" and the src attribute to the URI of the VoiceXML document that is to be prepared by the MS. For example:

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

The VoiceXML dialog environment uses the <dialogprepare> request as an opportunity to fetch and validate the initial document indicated by the src attribute along with any resources referenced in the VoiceXML document marked as prefetchable. The maxage and maxstale attributes, if specified, control how the initial VoiceXML document is fetched using HTTP (see [RFC2616]). Note that the fetchtimeout attribute is not defined in VoiceXML for an initial document, but the MS MUST support this attribute in its VoiceXML environment.

If a <params> child element of <dialogprepare> is specified, then the MS MUST map the parameter information into a VoiceXML session variable object as described in Section 9.2.3.

The success or failure of the VoiceXML document preparation is reported in the MS response. For example, if the VoiceXML document cannot be retrieved, then a 409 error response is returned. If the document is syntactically invalid according to VoiceXML, then a 400 response is returned. If successful, the response includes a dialogid attribute whose value the AS can use in <dialogstart> element to start the prepared dialog.

9.2. Starting a VoiceXML Dialog

A VoiceXML dialog is started by sending the MS a request containing a <dialogstart> element (Section 4.2.2). If a VoiceXML dialog has already been prepared using <dialogprepare>, then the MS starts the dialog indicated by the prepareddialogid attribute. Otherwise, a new VoiceXML dialog can be started by setting the type attribute to "application/voicexml+xml" and the src attribute to the URI of the VoiceXML document. For example:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="ssd3r3:sds345b"
              type="application/voicexml+xml"
              src="http://www.example.com/mydialog.vxml"
              fetchtimeout="15s"/>
</mscivr>
```

The maxage and maxstale attributes, if specified, control how the initial VoiceXML document is fetched using HTTP (see [RFC2616]). Note that the fetchtimeout attribute is not defined in VoiceXML for an initial document, but the MS MUST support this attribute in its

VoiceXML environment. Note also that support for <dtmfsub> subscriptions (Section 4.2.2.1.1) and their associated dialog notification events is not defined in VoiceXML. If such a subscription is specified in a <dialogstart> request, then the MS sends a 439 error response (see Section 4.5).

The success or failure of starting a VoiceXML dialog is reported in the MS response as described in Section 4.2.2.

When the MS starts a VoiceXML dialog, the MS MUST map session information into a VoiceXML session variable object. There are 3 types of session information: protocol information (Section 9.2.1), media stream information (Section 9.2.2), and parameter information (Section 9.2.3).

9.2.1. Session Protocol Information

If the connectionid attribute is specified, the MS assigns protocol information from the SIP dialog associated with the connection to the following session variables in VoiceXML:

session.connection.local.uri Evaluates to the SIP URI specified in the 'To:' header of the initial INVITE.

session.connection.remote.uri Evaluates to the SIP URI specified in the 'From:' header of the initial INVITE.

session.connection.originator Evaluates to the value of session.connection.remote (MS receives inbound connections but does not create outbound connections).

session.connection.protocol.name Evaluates to "sip". Note that this is intended to reflect the use of SIP in general, and does not distinguish between whether the connection accesses the MS via SIP or SIP Secure (SIPS) procedures.

session.connection.protocol.version Evaluates to "2.0".

session.connection.redirect This array is populated by information contained in the 'History-Info' header [RFC4244] in the initial INVITE or is otherwise undefined. Each entry (hi-entry) in the 'History-Info' header is mapped, in the order it appeared in the 'History-Info' header, into an element of the session.connection.redirect array. Properties of each element of the array are determined as follows:

uri Set to the hi-targeted-to-uri value of the History-Info entry

pi Set to 'true' if hi-targeted-to-uri contains a 'Privacy=history' parameter, or if the INVITE 'Privacy' header includes 'history'; 'false' otherwise

si Set to the value of the 'si' parameter if it exists; undefined otherwise

reason Set verbatim to the value of the 'Reason' parameter of hi-targeted-to-uri

session.connection.aai Evaluates to the value of a SIP header with the name "aai" if present; undefined otherwise.

session.connection.protocol.sip.requesturi This is an associative array where the array keys and values are formed from the URI parameters on the SIP Request-URI of the initial INVITE. The array key is the URI parameter name. The corresponding array value is obtained by evaluating the URI parameter value as a string. In addition, the array's toString() function returns the full SIP Request-URI.

session.connection.protocol.sip.headers This is an associative array where each key in the array is the non-compact name of a SIP header in the initial INVITE converted to lowercase (note the case conversion does not apply to the header value). If multiple header fields of the same field name are present, the values are combined into a single comma-separated value. Implementations MUST at a minimum include the 'Call-ID' header and MAY include other headers. For example, session.connection.protocol.sip.headers["call-id"] evaluates to the Call-ID of the SIP dialog.

If a conferenceid attribute is specified, then the MS populates the following session variables in VoiceXML:

session.conference.name Evaluates to the value of the conferenceid attribute.

9.2.2. Session Media Stream Information

The media streams of the connection or conference to use for the dialog are described in Section 4.2.2, including use of <stream> elements (Section 4.2.2.2) if specified. The MS maps media stream information into the VoiceXML session variable session.connection.protocol.sip.media for a connection, and session.conference.media for a conference. In both variables, the value of the variable is an array where each array element is an object with the following properties:

type This required property indicates the type of the media associated with the stream (see Section 4.2.2.2 <stream> type attribute definition).

direction This required property indicates the directionality of the media relative to the endpoint of the dialog (see Section 4.2.2.2 <stream> direction attribute definition).

format This property is optional. If defined, the value of the property is an array. Each array element is an object that specifies information about one format of the media stream. The object contains at least one property called name whose value is the subtype name of the media format [RFC4855]. Other properties may be defined with string values; these correspond to required and, if defined, optional parameters of the format.

As a consequence of this definition, when a connectionid is specified there is an array entry in session.connection.protocol.sip.media for each media stream used by the VoiceXML dialog. For an example, consider a connection with bidirectional G.711 mu-law audio sampled at 8kHz where the dialog is started with

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="ssd3r3:sds345b"
              type="application/voicexml+xml"
              src="http://www.example.com/mydialog.vxml"
              fetchtimeout="15s">
  <stream media="audio" direction="recvonly"/>
 </dialogstart>
</mscivr>
```

In this case, session.connection.protocol.sip.media[0].type evaluates to "audio", session.connection.protocol.sip.media[0].direction evaluates to "recvonly" (i.e., the endpoint only receives media from the dialog -- the endpoint does not send media to the dialog), session.connection.protocol.sip.media[0].format[0].name evaluates to "PCMU", and session.connection.protocol.sip.media[0].format[0].rate evaluates to "8000".

Note that the session variable is updated if the connection or conference media session characteristics for the VoiceXML dialog change (e.g., due to a SIP re-INVITE).

9.2.3. Session Parameter Information

Parameter information is specified in the <params> child element of <dialogprepare> and <dialogstart> elements, where each parameter is specified using a <param> element. The MS maps parameter information into VoiceXML session variables as follows:

session.values This is an associative array mapped to the <params> element. It is undefined if no <params> element is specified. If a <params> element is specified in both <dialogprepare> and <dialogstart> elements for the same dialog, then the array is first initialized with the <params> specified in the <dialogprepare> element and then updated with the <params> specified in the <dialogstart> element; in cases of conflict, the <dialogstart> parameter value take priority. Array keys and values are formed from <param> children of the <params> element. Each array key is the value of the name attribute of a <param> element. If the same name is used in more than one <param> element, then the array key is associated with the last <param> in document order. The corresponding value for each key is an object with two required properties: a "type" property evaluating to the value of the type attribute, and a "content" property evaluating to the content of the <param>. In addition, this object's toString() function returns the value of the "content" property as a string.

For example, a VoiceXML dialog started with one parameter:

```
<mscivr version="1.0" xmlns="urn:ietf:params:xml:ns:msc-ivr">
 <dialogstart connectionid="ssd3r3:sds345b"
              type="application/voicexml+xml"
              src="http://www.example.com/mydialog.vxml"
              fetchtimeout="15s">
  <params>
   <param name="mode">playannouncement</param>
  </params>
 </dialogstart>
</mscivr>
```

In this case, session.values would be defined with one item in the array where session.values['mode'].type evaluates to "text/plain" (the default value), session.values['mode'].content evaluates to "playannouncement", and session.values['mode'].toString() also evaluates to "playannouncement".

The MS sends an error response (see Section 4.2.2) if a <param> is not supported by the MS (e.g., the parameter type is not supported).

9.3. Terminating a VoiceXML Dialog

When the MS receives a request with a <dialogterminate> element (Section 4.2.3), the MS throws a 'connection.disconnect.hangup' event into the specified VoiceXML dialog. Note that if the immediate attribute has the value true, then the MS MUST NOT return <params> information when the VoiceXML dialog exits (even if the VoiceXML dialog provides such information) -- see Section 9.4.

If the connection or conference associated with the VoiceXML dialog terminates, then the MS throws a 'connection.disconnect.hangup' event into the specified VoiceXML dialog.

9.4. Exiting a VoiceXML Dialog

The MS sends a <dialogexit> notification event (Section 4.2.5.1) when the VoiceXML dialog is complete, has been terminated, or exits due to an error. The <dialogexit> status attribute specifies the status of the VoiceXML dialog when it exits and its <params> child element specifies information, if any, returned from the VoiceXML dialog.

A VoiceXML dialog exits when it processes a <disconnect> element, an <exit> element, or an implicit exit according to the VoiceXML form interpretation algorithm (FIA). If the VoiceXML dialog executes a <disconnect> and then subsequently executes an <exit> with namelist information, the namelist information from the <exit> element is discarded.

The MS reports namelist variables in the <params> element of the <dialogexit>. Each <param> reports on a namelist variable. The MS set the <param> name attribute to the name of the VoiceXML variable. The MS sets the <param> type attribute according to the type of the VoiceXML variable. The MS sets the <param> type to 'text/plain' when the VoiceXML variable is a simple ECMAScript value. If the VoiceXML variable is a recording, the MS sets the <param> type to the MIME media type of the recording and encodes the recorded content as CDATA in the <param> (see Section 4.2.6.1 for an example). If the VoiceXML variable is a complex ECMAScript value (e.g., object, array, etc.), the MS sets the <param> type to 'application/json' and converts the variable value to its JSON value equivalent [RFC4627]. The behavior resulting from specifying an ECMAScript object with circular references is not defined.

If the expr attribute is specified on the VoiceXML <exit> element instead of the namelist attribute, the MS creates a <param> element with the reserved name '__exit'. If the value is an ECMAScript literal, the <param> type is 'text/plain' and the content is the literal value. If the value is a variable, the <param> type and

content are set in the same way as a namelist variable; for example, an expr attribute referencing a variable with a simple ECMAScript value has the type 'text/plain' and the content is set to the ECMAScript value. To allow the AS to differentiate between a <dialogexit> notification event resulting from a VoiceXML <disconnect> from one resulting from an <exit>, the MS creates a <param> with the reserved name '__reason', the type 'text/plain', and a value of "disconnect" (without brackets) to reflect the use of VoiceXML's <disconnect> element, and the value of "exit" (without brackets) to an explicit <exit> in the VoiceXML dialog. If the VoiceXML session terminates for other reasons (such as encountering an error), this parameter MAY be omitted or take on platform-specific values prefixed with an underscore.

Table 2 provides some examples of VoiceXML <exit> usage and the corresponding <params> element in the <dialogexit> notification event. It assumes the following VoiceXML variable names and values: userAuthorized=true, pin=1234, and errors=0. The <param> type attributes ('text/plain') are omitted for clarity.

<exit> Usage	<params> Result
<exit>	<params> <param name="__reason">exit</param> </params>
<exit expr="5">	<params> <param name="__reason">exit</param> <param name="__exit">5</param> </params>
<exit expr="'done'">	<params> <param name="__reason">exit</param> <param name="__exit">'done'</param> </params>
<exit expr="userAuthorized">	<params> <param name="__reason">exit</param> <param name="__exit">true</param> </params>
<exit namelist="pin errors">	<params> <param name="__reason">exit</param> <param name="pin">1234</param> <param name="errors">0</param> </params>

Table 2: VoiceXML <exit> Mapping Examples

9.5. Call Transfer

While VoiceXML is at its core a dialog language, it also provides optional call transfer capability. It is NOT RECOMMENDED to use VoiceXML's call transfer capability in networks involving application servers. Rather, the AS itself can provide call routing

functionality by taking signaling actions based on the data returned to it, either through VoiceXML's own data submission mechanisms or through the mechanism described in Section 9.4. If the MS encounters a VoiceXML dialog requesting call transfer capability, the MS SHOULD raise an error event in the VoiceXML dialog execution context: an error.unsupported.transfer.blind event if blind transfer is requested, error.unsupported.transfer.bridge if bridge transfer is requested, or error.unsupported.transfer.consultation if consultation transfer is requested.

10. Contributors

Asher Shiratzky 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, Diego Besprosvan, Mary Barnes, and Steve Buko, who provided valuable feedback, input, and text to this document.

11. Acknowledgments

The authors would like to thank Adnan Saleem, Gene Shtirmer, Dave Burke, Dan York, Steve Buko, Jean-Francois Bertrand, Henry Lum, and Lorenzo Miniero for expert reviews of this work.

Ben Campbell carried out the RAI expert review on this specification and provided a great deal of invaluable input. Donald Eastlake carried out a thorough security review.

12. References

12.1. Normative References

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

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

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

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

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

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

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

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

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

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

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

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

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

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

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

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

[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] Jansen, J., Layaida, N., Michel, T., Grassel, G., Koivisto, A., Bulterman, D., Mullender, S., and D. Zucker, "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.

12.2. Informative References

[CCXML10] Auburn, R J., "Voice Browser Call Control: CCXML Version 1.0", W3C Candidate Recommendation (work in progress), April 2010.

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

[IANA] IANA, "RTP Payload Types", available from http://www.iana.org.

[MIME.mediatypes] IANA, "MIME Media Types", available from http://www.iana.org.

[MIXER-CP] McGlashan, S., Melanchuk, T., and C. Boulton, "A Mixer Control Package for the Media Control Channel Framework", Work in Progress, January 2011.

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

[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.

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

[RFC4244] Barnes, M., "An Extension to the Session Initiation Protocol (SIP) for Request History Information", RFC 4244, November 2005.

[RFC4267] Froumentin, M., "The W3C Speech Interface Framework Media Types: application/voicexml+xml, application/ssml+xml, application/srgs, application/srgs+xml, application/ccxml+xml, and application/pls+xml", RFC 4267, November 2005.

[RFC4281] Gellens, R., Singer, D., and P. Frojdh, "The Codecs Parameter for "Bucket" Media Types", RFC 4281, November 2005.

[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.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.

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

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

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

[VXML30] McGlashan, S., Burnett, D., Akolkar, R., Auburn, RJ., Baggia, P., Barnett, J., Bodell, M., Carter, J., Oshry, M., Rehor, K., Young, M., and R. Hosn, "Voice Extensible Markup Language (VoiceXML) Version 3.0", W3C Working Draft, August 2010.

[XCON-DATA-MODEL]
Novo, O., Camarillo, G., Morgan, D., and J. Urpalainen, "Conference Information Data Model for Centralized Conferencing (XCON)", Work in Progress, April 2011.

Authors' Addresses

Scott McGlashan
Hewlett-Packard

EMail: smcg.stds01@mcglashan.org

Tim Melanchuk
Rainwillow

EMail: timm@rainwillow.com

Chris Boulton
NS-Technologies

EMail: chris@ns-technologies.com