Network Working Group
Request for Comments: 5552
Category: Standards Track
May 2009

D. Burke
Google
M. Scott
Genesys

SIP Interface to VoiceXML Media Services

Status of This Memo

Copyright Notice

Abstract

This document describes a SIP interface to VoiceXML media services. Commonly, Application Servers controlling Media Servers use this protocol for pure VoiceXML processing capabilities. This protocol is an adjunct to the full MEDIACTRL protocol and packages mechanism.

Table of Contents

1. Introduction

VoiceXML [VXML20], [VXML21] is a World Wide Web Consortium (W3C) standard for creating audio and video dialogs that feature synthesized speech, digitized audio, recognition of spoken and dual tone multi-frequency (DTMF) key input, recording of audio and video, telephony, and mixed-initiative conversations. VoiceXML allows Web-based development and content delivery paradigms to be used with interactive video and voice response applications.

This document describes a SIP [RFC3261] interface to VoiceXML media services. Commonly, Application Servers controlling media servers use this protocol for pure VoiceXML processing capabilities. SIP is responsible for initiating a media session to the VoiceXML media server and simultaneously triggering the execution of a specified VoiceXML application. This protocol is an adjunct to the full MEDIACTRL protocol and packages mechanism.

The interface described here leverages a mechanism for identifying dialog media services first described in [RFC4240]. The interface has been updated and extended to support the W3C Recommendation for VoiceXML 2.0 [VXML20] and VoiceXML 2.1 [VXML21]. A set of commonly implemented functions and extensions have been specified including VoiceXML dialog preparation, outbound calling, video media support, and transfers. VoiceXML session variable mappings have been defined for SIP with an extensible mechanism for passing application-specific values into the VoiceXML application. Mechanisms for returning data to the Application Server have also been added.

1.1. Use Cases

The VoiceXML media service user in this document is generically referred to as an Application Server. In practice, it is intended that the interface defined by this document be applicable across a wide range of use cases. Several intended use cases are described below.

1.1.1. IVR Services with Application Servers

SIP Application Servers provide services to users of the network. Typically, there may be several Application Servers in the same network, each specialized in providing a particular service. Throughout this specification and without loss of generality, we posit the presence of an Application Server specialized in providing Interactive Voice Response (IVR) services. A typical configuration for this use case is illustrated below.

```
                              +--------------+
                              |              |
                              |  Application |\
                              |    Server    | \
                              |              |  \ HTTP
                         SIP  +--------------+   \
                              /               \   \
             +-------------+ /             SIP \ +---------------+
             |             |/                   \|               |
             |     SIP     |                     |   VoiceXML    |
             | User Agent  |      RTP/SRTP       | Media Server  |
             |             |=====================|               |
             +-------------+                     +---------------+
```

Assuming the Application Server also supports HTTP, the VoiceXML application may be hosted on it and served up via HTTP [RFC2616]. Note, however, that the Web model allows the VoiceXML application to be hosted on a separate (HTTP) Application Server from the (SIP) Application Server that interacts with the VoiceXML Media Server via this specification. It is also possible for a static VoiceXML application to be stored locally on the VoiceXML Media Server, leveraging the VoiceXML 2.1 [VXML21] <data> mechanism to interact with a Web/Application Server when dynamic behavior is required. The viability of static VoiceXML applications is further enhanced by the mechanisms defined in Section 2.4, through which the Application Server can make session-specific information available within the VoiceXML session context.

The approach described in this document is sometimes termed the "delegation model" -- the Application Server is essentially delegating programmatic control of the human-machine interactions to one or more VoiceXML documents running on the VoiceXML Media Server. During the human-machine interactions, the Application Server remains in the signaling path and can respond to results returned from the VoiceXML Media Server or other external network events.

1.1.2. PSTN IVR Service Node

While this document is intended to enable enhanced use of VoiceXML as a component of larger systems and services, it is intended that devices that are completely unaware of this specification remain capable of invoking VoiceXML services offered by a VoiceXML Media Server compliant with this document. A typical configuration for this use case is as follows:

```
             +-------------+         SIP         +--------------+
             |             |---------------------|              |
             |   IP/PSTN   |                     |   VoiceXML   |
             |   Gateway   |      RTP/SRTP       | Media Server |
             |             |=====================|              |
             +-------------+                     +--------------+
```

Note also that beyond the invocation and termination of a VoiceXML dialog, the semantics defined for call transfers using REFER are intended to be compatible with standard, existing IP/PSTN (Public Switched Telephone Network) gateways.

1.1.3. 3GPP IMS Media Resource Function (MRF)

The 3rd Generation Partnership Project (3GPP) IP Multimedia Subsystem (IMS) [TS23002] defines a Media Resource Function (MRF) used to offer media processing services such as conferencing, transcoding, and prompt/collect. The capabilities offered by VoiceXML are ideal for offering richer media processing services in the context of the MRF. In this architecture, the interface defined here corresponds to the "Mr" interface to the MRFC (MRF Controller); the implementation of this interface might use separated MRFC and MRFP (MRF Processor) elements (as per the IMS architecture), or might be an integrated MRF (as is common practice).

```
             +----------+
             |   App    |
             |  Server  |
             +----------+
                  |
                  | SIP (ISC)
                  |
             +----------+   SIP (Mr)    +--------------+
             |  S-CSCF  |---------------|   VoiceXML   |
             |          |               |     MRF      |
             +----------+               +--------------+
                                               ||
                                               || RTP/SRTP (Mb)
                                               ||
```

The above diagram is highly simplified and shows a subset of nodes typically involved in MRF interactions. It should be noted that while the MRF will primarily be used by the Application Server via the Serving Call Session Control Function (S-CSCF), it is also possible for calls to be routed directly to the MRF without the involvement of an Application Server.

Although the above is described in terms of the 3GPP IMS architecture, it is intended that it is also applicable to 3GPP2, Next Generation Network (NGN), and PacketCable architectures that are converging with 3GPP IMS standards.

1.1.4. CCXML <-> VoiceXML Interaction

Call Control eXtensible Markup Language (CCXML) 1.0 [CCXML10] applications provide services mainly through controlling the interaction between Connections, Conferences, and Dialogs. Although CCXML is capable of supporting arbitrary dialog environments, VoiceXML is commonly used as a dialog environment in conjunction with CCXML applications; CCXML is specifically designed to effectively support the use of VoiceXML. CCXML 1.0 defines language elements that allow for Dialogs to be prepared, started, and terminated; it further allows for data to be returned by the dialog environment, for call transfers to be requested (by the dialog) and responded to by the CCXML application, and for arbitrary eventing between the CCXML application and running dialog application.

The interface described in this document can be used by CCXML 1.0 implementations to control VoiceXML Media Servers. Note, however, that some CCXML language features require eventing facilities between CCXML and VoiceXML sessions that go beyond what is defined in this specification. For example, VoiceXML-controlled call transfers and mid-dialog, application-defined events cannot be fully realized using this specification alone. A SIP event package [RFC3265] MAY be used in addition to this specification to provide extended eventing.

1.1.5. Other Use Cases

In addition to the use cases described in some detail above, there are a number of other intended use cases that are not described in detail, such as:

1. Use of a VoiceXML Media Server as an adjunct to an IP-based Private Branch Exchange / Automatic Call Distributor (PBX/ACD), possibly to provide voicemail/messaging, automated attendant, or other capabilities.

2. Invocation and control of a VoiceXML session that provides the voice modality component in a multimodal system.

1.2. Terminology

Application Server: A SIP Application Server hosts and executes services, in particular by terminating SIP sessions on a media server. The Application Server MAY also act as an HTTP server [RFC2616] in interactions with media servers.

VoiceXML Media Server: A VoiceXML interpreter including a SIP-based interpreter context and the requisite media processing capabilities to support VoiceXML functionality.

VoiceXML Session: A VoiceXML Session is a multimedia session comprising of at least a SIP User Agent, a VoiceXML Media Server, the data streams between them, and an executing VoiceXML application.

VoiceXML Dialog: Equivalent to VoiceXML Session.

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

2. VoiceXML Session Establishment and Termination

This section describes how to establish a VoiceXML Session, with or without preparation, and how to terminate a session. This section also addresses how session information is made available to VoiceXML applications.

2.1. Service Identification

The SIP Request-URI is used to identify the VoiceXML media service. The user part of the SIP Request-URI is fixed to "dialog". This is done to ensure compatibility with [RFC4240], since this document extends the dialog interface defined in that specification and because this convention from [RFC4240] is widely adopted by existing media servers.

Standardizing the SIP Request-URI including the user part also improves interoperability between Application Servers and media servers, and reduces the provisioning overhead that would be required if use of a media server by an Application Server required an individually provisioned URI. In this respect, this document (and [RFC4240]) do not add semantics to the user part, but rather standardize the way that targets on media servers are provisioned. Further, since Application Servers -- and not human beings -- are generally the clients of media servers, issues such as interpretation and internationalization do not apply.

Exposing a VoiceXML media service with a well-known address may enhance the possibility of exploitation: the VoiceXML Media Server is RECOMMENDED to use standard SIP mechanisms to authenticate endpoints as discussed in Section 9.

The initial VoiceXML document is specified with the "voicexml" parameter. In addition, parameters are defined that control how the VoiceXML Media Server fetches the specified VoiceXML document. The list of parameters defined by this specification is as follows (note the parameter names are case-insensitive):

voicexml: URI of the initial VoiceXML document to fetch. This will typically contain an HTTP URI, but may use other URI schemes, for example, to refer to local, static VoiceXML documents. If the "voicexml" parameter is omitted, the VoiceXML Media Server may select the initial VoiceXML document by other means, such as by applying a default, or may reject the request.

maxage: Used to set the max-age value of the Cache-Control header in conjunction with VoiceXML documents fetched using HTTP, as per [RFC2616]. If omitted, the VoiceXML Media Server will use a default value.

maxstale: Used to set the max-stale value of the Cache-Control header in conjunction with VoiceXML documents fetched using HTTP, as per [RFC2616]. If omitted, the VoiceXML Media Server will use a default value.

method: Used to set the HTTP method applied in the fetch of the initial VoiceXML document. Allowed values are "get" or "post" (case-insensitive). Default is "get".

postbody: Used to set the application/x-www-form-urlencoded encoded [HTML4] HTTP body for "post" requests (or is otherwise ignored).

ccxml: Used to specify a "JSON value" [RFC4627] that is mapped to the session.connection.ccxml VoiceXML session variable -- see Section 2.4.

aai: Used to specify a "JSON value" [RFC4627] that is mapped to the session.connection.aai VoiceXML session variable -- see Section 2.4.

Other application-specific parameters may be added to the Request-URI and are exposed in VoiceXML session variables (see Section 2.4).

Formally, the Request-URI for the VoiceXML media service has a fixed user part "dialog". Seven URI parameters are defined (see the definition of uri-parameter in Section 25.1 of [RFC3261]).

```
  dialog-param      = "voicexml=" vxml-url ; vxml-url follows the URI
                                           ; syntax defined in [RFC3986]
  maxage-param      = "maxage=" 1*DIGIT
  maxstale-param    = "maxstale=" 1*DIGIT
  method-param      = "method=" ("get" / "post")
  postbody-param    = "postbody=" token
  ccxml-param       = "ccxml=" json-value
  aai-param         = "aai=" json-value
  json-value        =  false /
                       null /
                       true /
                       object /
                       array /
                       number /
                       string ; defined in [RFC4627]
```

Parameters of the Request-URI in subsequent re-INVITEs are ignored. One consequence of this is that the VoiceXML Media Server cannot be instructed by the Application Server to change the executing VoiceXML Application after a VoiceXML Session has been started.

Special characters contained in the dialog-param, postbody-param, ccxml-param, and aai-param values must be URL-encoded ("escaped") as required by the SIP URI syntax, for example, ’?’ (%3f), ’=’ (%3d), and ’;’ (%3b). The VoiceXML Media Server MUST therefore unescape these parameter values before making use of them or exposing them to running VoiceXML applications. It is important that the VoiceXML Media Server only unescape the parameter values once since the desired VoiceXML URI value could itself be URL encoded, for example.

Since some applications may choose to transfer confidential information, the VoiceXML Media Server MUST support the sips: scheme as discussed in Section 9.

Informative note: With respect to the postbody-param value, since the application/x-www-form-urlencoded content itself escapes non-alphanumeric characters by inserting %HH replacements, the escaping rules above will result in the ’%’ characters being further escaped in addition to the ’&’ and ’=’ name/value separators.

As an example, the following SIP Request-URI identifies the use of VoiceXML media services, with 'http://appserver.example.com/promptcollect.vxml' as the initial VoiceXML document, to be fetched with max-age/max-stale values of 3600s/0s, respectively:

```
    sip:dialog@mediaserver.example.com; \
       voicexml=http://appserver.example.com/promptcollect.vxml; \
       maxage=3600;maxstale=0
```

2.2. Initiating a VoiceXML Session

A VoiceXML Session is initiated via the Application Server using a SIP INVITE. Typically, the Application Server will be specialized in providing VoiceXML services. At a minimum, the Application Server may behave as a simple proxy by rewriting the Request-URI received from the User Agent to a Request-URI suitable for consumption by the VoiceXML Media Server (as specified in Section 2.1). For example, a User Agent might present a dialed number:

```
    tel:+1-201-555-0123
```

that the Application Server maps to a directory assistance application on the VoiceXML Media Server with a Request-URI of:

```
    sip:dialog@ms1.example.com; \
       voicexml=http://as1.example.com/da.vxml
```

Certain header values in the INVITE message to the VoiceXML Media Server are mapped into VoiceXML session variables and are specified in Section 2.4.

On receipt of the INVITE, the VoiceXML Media Server issues a provisional response, 100 Trying, and commences the fetch of the initial VoiceXML document. The 200 OK response indicates that the VoiceXML document has been fetched and parsed correctly and is ready for execution. Application execution commences on receipt of the ACK (except if the dialog is being prepared as specified in Section 2.3). Note that the 100 Trying response will usually be sent on receipt of the INVITE in accordance with [RFC3261], since the VoiceXML Media Server cannot in general guarantee that the initial fetch will complete in less than 200 ms. However, certain implementations may be able to guarantee response times to the initial INVITE, and thus may not need to send a 100 Trying response.

As an optimization, prior to sending the 200 OK response, the VoiceXML Media Server MAY execute the application up to the point of the first VoiceXML waiting state or prompt flush.

A VoiceXML Media Server, like any SIP User Agent, may be unable to accept the INVITE request for a variety of reasons. For instance, a Session Description Protocol (SDP) offer contained in the INVITE might require the use of codecs that are not supported by the Media Server. In such cases, the Media Server should respond as defined by [RFC3261]. However, there are error conditions specific to VoiceXML, as follows:

1. If the Request-URI does not conform to this specification, a 400 Bad Request MUST be returned (unless it is used to select other services not defined by this specification).

2. If a URI parameter in the Request-URI is repeated, then the request MUST be rejected with a 400 Bad Request response.

3. If the Request-URI does not include a "voicexml" parameter, and the VoiceXML Media Server does not elect to use a default page, the VoiceXML Media Server MUST return a final response of 400 Bad Request, and it SHOULD include a Warning header with a 3-digit code of 399 and a human-readable error message.

4. If the VoiceXML document cannot be fetched or parsed, the VoiceXML Media Server MUST return a final response of 500 Server Internal Error and SHOULD include a Warning header with a 3-digit code of 399 and a human-readable error message.

Informative note: Certain applications may pass a significant amount of data to the VoiceXML dialog in the form of Request-URI parameters. This may cause the total size of the INVITE request to exceed the MTU of the underlying network. In such cases, applications/implementations must take care either to use a transport appropriate to these larger messages (such as TCP) or to use alternative means of passing the required information to the VoiceXML dialog (such as supplying a unique session identifier in the initial VoiceXML URI and later using that identifier as a key to retrieve data from the HTTP server).

2.3. Preparing a VoiceXML Session

In certain scenarios, it is beneficial to prepare a VoiceXML Session for execution prior to running it. A previously prepared VoiceXML Session is expected to execute with minimal delay when instructed to do so.

If a media-less SIP dialog is established with the initial INVITE to the VoiceXML Media Server, the VoiceXML application will not execute after receipt of the ACK. To run the VoiceXML application, the Application Server (AS) must issue a re-INVITE to establish a media session.

A media-less SIP dialog can be established by sending an SDP containing no media lines in the initial INVITE. Alternatively, if no SDP is sent in the initial INVITE, the VoiceXML Media Server will include an offer in the 200 OK message, which can be responded to with an answer in the ACK with the media port(s) set to 0.

Once a VoiceXML application is running, a re-INVITE that disables the media streams (i.e., sets the ports to 0) will not otherwise affect the executing application (except that recognition actions initiated while the media streams are disabled will result in noinput timeouts).

2.4. Session Variable Mappings

The standard VoiceXML session variables are assigned values according to:

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.redirect: This array is populated by information contained in the History-Info [RFC4244] header in the initial INVITE or is otherwise undefined. Each entry (hi-entry) in the History-Info header is mapped, in reverse order, 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.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 media server was accessed via SIP or SIPS procedures.

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

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.

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 converted to lowercase (note the case conversion does not apply to the parameter value). The corresponding array value is obtained by evaluating the URI parameter value as a "JSON value" [RFC4627] in the case of the ccxml-param and aai-param values and otherwise as a string. In addition, the array's toString() function returns the full SIP Request-URI. For example, assuming a Request-URI of sip:dialog@ example.com;voicexml=http://example.com;aai=%7b"x":1%2c"y":true%7d then session.connection.protocol.sip.requesturi["voicexml"] evaluates to "http://example.com", session.connection.protocol.sip.requesturi["aai"].x evaluates to 1 (type Number), session.connection.protocol.sip.requesturi["aai"].y evaluates to true (type Boolean), and session.connection.protocol.sip.requesturi evaluates to the complete Request-URI (type String) 'sip:dialog@ example.com;voicexml=http://example.com;aai={"x":1,"y":true}'.

session.connection.aai: Evaluates to session.connection.protocol.sip.requesturi["aai"].

session.connection.ccxml: Evaluates to session.connection.protocol.sip.requesturi["ccxml"].

session.connection.protocol.sip.media: This 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. The value is a string. It is strongly recommended that the following values are used for common types of media: "audio" for audio media, and "video" for video media.

* direction: - This required property indicates the directionality of the media relative to session.connection.originator. Defined values are sendrecv, sendonly, recvonly, and inactive.

* 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 (there is an array element for each payload type on the m-line). The object contains at least one property called "name" whose value is the MIME subtype of the media format (MIME subtypes are registered in [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, there is an array entry in session.connection.protocol.sip.media for each non-disabled m-line for the negotiated media session. Note that this session variable is updated if the media session characteristics for the VoiceXML Session change (i.e., due to a re-INVITE). For an example, consider a connection with bidirectional G.711 mu-law "audio" sampled at 8 kHz. In this case, session.connection.protocol.sip.media[0].type evaluates to "audio", session.connection.protocol.sip.media[0].direction to "sendrecv", session.connection.protocol.sip.media[0].format[0].name evaluates to "audio/PCMU", and session.connection.protocol.sip.media[0].format[0].rate evaluates to "8000".

Note that when accessing SIP headers and Request-URI parameters via the session.connection.protocol.sip.headers and session.connection.protocol.sip.requesturi associative arrays defined above, applications can choose between two semantically equivalent ways of referring to the array. For example, either of the following can be used to access a Request-URI parameter named "foo":

```
session.connection.protocol.sip.requesturi["foo"]
session.connection.protocol.sip.requesturi.foo
```

However, it is important to note that not all SIP header names or Request-URI parameter names are valid ECMAScript identifiers, and as such, can only be accessed using the first form (array notation). For example, the Call-ID header can only be accessed as session.connection.protocol.sip.headers["call-id"]; attempting to access the same value as session.connection.protocol.sip.headers.call-id would result in an error.

2.5. Terminating a VoiceXML Session

The Application Server can terminate a VoiceXML Session by issuing a BYE to the VoiceXML Media Server. Upon receipt of a BYE in the context of an existing VoiceXML Session, the VoiceXML Media Server MUST send a 200 OK response and MUST throw a 'connection.disconnect.hangup' event to the VoiceXML application. If the Reason header [RFC3326] is present on the BYE Request, then the value of the Reason header is provided verbatim via the '_message' variable within the catch element's anonymous variable scope.

The VoiceXML Media Server may also initiate termination of the session by issuing a BYE request. This will typically occur as a result of encountering a <disconnect> or <exit> in the VoiceXML application, due to the VoiceXML application running to completion, or due to unhandled errors within the VoiceXML application.

See Section 4 for mechanisms to return data to the Application Server.

2.6. Examples

2.6.1. Basic Session Establishment

This example illustrates an Application Server setting up a VoiceXML Session on behalf of a User Agent.

```
                         SIP               VoiceXML              HTTP
   User              Application            Media            Application
   Agent               Server               Server              Server
    |                    |                    |                    |
    |(1) INVITE [offer]  |                    |                    |
    |------------------->|(2) INVITE [offer]  |                    |
    |(3) 100 Trying      |------------------->|                    |
    |<-------------------|(4) 100 Trying      |                    |
    |                    |<-------------------|                    |
    |                    |                    |                    |
    |                    |                    |(5) GET             |
    |                    |                    |------------------->|
    |                    |                    |(6) 200 OK [VXML]   |
    |                    |                    |<-------------------|
    |                    |                    |                    |
    |                    |(7) 200 OK [answer] |                    |
    |(8) 200 OK [answer] |<-------------------|                    |
    |<-------------------|                    |                    |
    |(9) ACK             |                    |                    |
    |------------------->|(10) ACK            |                    |
    |                    |------------------->| (execute           |
    |(11) RTP/SRTP       |                    |  VoiceXML          |
    |.........................................|  application)      |
    |                    |                    |                    |
```

2.6.2. VoiceXML Session Preparation

This example demonstrates the preparation of a VoiceXML Session. In this example, the VoiceXML session is prepared prior to placing an outbound call to a User Agent, and is started as soon as the User Agent answers.

The [answer1:0] notation is used to indicate an SDP answer with the media ports set to 0.

```
                         SIP               VoiceXML              HTTP
   User              Application             Media            Application
   Agent               Server               Server               Server
    |                    |                     |                    |
    |                    |(1) INVITE           |                    |
    |                    |-------------------->|                    |
    |                    |(2) 100 Trying       |                    |
    |                    |<--------------------|                    |
    |                    |                     |                    |
    |                    |                     |(3) GET             |
    |                    |                     |------------------->|
    |                    |                     |(4) 200 OK [VXML]   |
    |                    |                     |<-------------------|
    |                    |                     |                    |
    |                    |(5) 200 OK [offer1]  |                    |
    |                    |<--------------------|                    |
    |                    |(6) ACK [answer1:0]  |                    |
    |(7) INVITE          |-------------------->|                    |
    |<-------------------|                     |                    |
    |(8) 200 OK [offer2] |                     |                    |
    |------------------->|(9) INVITE [offer2’] |                    |
    |                    |-------------------->|                    |
    |                    |(10) 100 Trying      |                    |
    |                    |<--------------------|                    |
    |                    |(11) 200 OK [answer2]|                    |
    |(12) ACK [answer2]  |<--------------------|                    |
    |<-------------------|(13) ACK             |                    |
    |                    |-------------------->| (execute           |
    |(14) RTP/SRTP                             |  VoiceXML          |
    |..........................................|  application)      |
    |                    |                     |                    |
```

Implementation detail: offer2’ is derived from offer2 -- it duplicates the m-lines and a-lines from offer2. However, offer2’ differs from offer2 since it must contain the same o-line as used in answer1:0 but with the version number incremented. Also, if offer1 has more m-lines than offer2, then offer2’ must be padded with extra (rejected) m-lines.

2.6.3. MRCP Establishment

Media Resource Control Protocol (MRCP) [MRCPv2] is a protocol that enables clients such as a VoiceXML Media Server to control media service resources such as speech synthesizers, recognizers, verifiers, and identifiers residing in servers on the network.

The example below illustrates how a VoiceXML Media Server may establish an MRCP session in response to an initial INVITE.

```
                       VoiceXML                                  HTTP
   User                Media                 MRCPv2          Application
   Agent               Server                Server             Server
    |                    |                      |                  |
    |(1) INVITE [offer1] |                      |                  |
    |------------------->|                      |                  |
    |(2) 100 Trying      |                      |                  |
    |<-------------------|(3) GET               |                  |
    |                    |---------------------------------------->|
    |                    |                      |                  |
    |                    |(4) 200 OK [VXML]     |                  |
    |                    |<----------------------------------------|
    |                    |                      |                  |
    |                    |(5) INVITE [offer2]   |                  |
    |                    |--------------------->|                  |
    |                    |                      |                  |
    |                    |(6) 200 OK [answer2]  |                  |
    |                    |<---------------------|                  |
    |                    |                      |                  |
    |                    |(7) ACK               |                  |
    |                    |--------------------->|                  |
    |                    |                      |                  |
    |                    |(8) MRCP connection   |                  |
    |                    |<-------------------->|                  |
    |(9) 200 OK [answer1]|                      |                  |
    |<-------------------|                      |                  |
    |                    |                      |                  |
    |(10) ACK            |                      |                  |
    |------------------->|                      |                  |
    |                    |                      |                  |
    |(11) RTP/SRTP       |                      |                  |
    |...........................................|                  |
    |                    |                      |                  |
```

In this example, the VoiceXML Media Server is responsible for establishing a session with the MRCPv2 Media Resource Server prior to sending the 200 OK response to the initial INVITE. The VoiceXML Media Server will perform the appropriate offer/answer with the MRCPv2 Media Resource Server based on the SDP capabilities of the Application Server and the MRCPv2 Media Resource Server. The VoiceXML Media Server will change the offer received from step 1 to establish an MRCPv2 session in step (5) and will re-write the SDP to include an m-line for each MRCPv2 resource to be used and other required SDP modifications as specified by MRCPv2. Once the VoiceXML Media Server performs the offer/answer with the MRCPv2 Media Resource Server, it will establish an MRCPv2 control channel in step (8). The MRCPv2 resource is deallocated when the VoiceXML Media Server receives or sends a BYE (not shown).

3. Media Support

This section describes the mandatory and optional media support required by this interface.

3.1. Offer/Answer

The VoiceXML Media Server MUST support the standard offer/answer mechanism of [RFC3264]. In particular, if an SDP offer is not present in the INVITE, the VoiceXML Media Server will make an offer in the 200 OK response listing its supported codecs.

3.2. Early Media

The VoiceXML Media Server MAY support early establishment of media streams as described in [RFC3960]. This allows the Application Server to establish media streams between a User Agent and the VoiceXML Media Server in parallel with the initial VoiceXML document being processed (which may involve dynamic VoiceXML page generation and interaction with databases or other systems). This is useful primarily for minimizing the delay in starting a VoiceXML Session, particularly in cases where a session with the User Agent already exists but the media stream associated with that session needs to be redirected to a VoiceXML Media Server.

The following flow demonstrates the use of early media (using the Gateway model defined in [RFC3960]):

```
                         SIP              VoiceXML             HTTP
   User              Application           Media            Application
   Agent                Server             Server              Server
    |                      |                   |                   |
    |..(existing session)..|                   |                   |
    |                      |(1) INVITE         |                   |
    |                      |------------------>|                   |
    |                      |                   |(2) HTTP GET       |
    |                      |                   |------------------>|
    |                      |(3) 183 [offer]    |                   |
    |(4) re-INVITE [offer] |<------------------|                   |
    |<---------------------|                   |                   |
    |(5) 200 OK [answer]   |                   |                   |
    |--------------------->|                   |                   |
    |(6) ACK               |                   |                   |
    |<---------------------|                   |                   |
    |                      | (7) PRACK [answer]|                   |
    |                      |------------------>|                   |
    |                      | (8) PRACK 200 OK  |                   |
    |                      |<------------------|                   |
    |(9) RTP/SRTP          |                   |                   |
    |..........................................|                   |
    |                      |                   |(10) 200 OK [VXML] |
    |                      |                   |<------------------|
    |                      |                   |                   |
    |                      |(11) 200 OK        |                   |
    |                      |<------------------|                   |
    |                      |(12) ACK           |                   |
    |                      |------------------>| (execute          |
    |                      |                   |  VoiceXML         |
    |                      |                   |  application)     |
    |                      |                   |                   |
```

   Although [RFC3960] prefers the use of the Application Server model
   for early media over the Gateway model, the primary issue with the
   Gateway model -- forking -- is significantly less common when issuing
   requests to VoiceXML Media Servers. This is because VoiceXML Media
   Servers respond to all requests with 200 OK responses in the absence
   of unusual errors, and they typically do so within several hundred
   milliseconds. This makes them unlikely targets in forking scenarios,
   since alternative targets of the forking process would virtually
   never be able to respond more quickly than an automated system,
   unless they are themselves automated systems -- in which case, there
   is little point in setting up a response time race between two
   automated systems. Issues with ringing tone generation in the
   Gateway model are also mitigated, both by the typically quick 200 OK

response time, and because this specification mandates that no media packets are generated until the receipt of an ACK (thus eliminating the need for the User Agent to perform media packet analysis).

Note that the offer of early media by a VoiceXML Media Server does not imply that the referenced VoiceXML application can always be fetched and executed successfully. For instance, if the HTTP Application Server were to return a 4xx response in step 10 above, or if the provided VoiceXML content was not valid, the VoiceXML Media Server would still return a 500 response (as per Section 2.2). At this point, it would be the responsibility of the Application Server to tear down any media streams established with the media server.

3.3. Modifying the Media Session

The VoiceXML Media Server MUST allow the media session to be modified via a re-INVITE and SHOULD support the UPDATE method [RFC3311] for the same purpose. In particular, it MUST be possible to change streams between sendrecv, sendonly, and recvonly as specified in [RFC3264].

Unidirectional streams are useful for announcement- or listening-only (hotword). The preferred mechanism for putting the media session on hold is specified in [RFC3264], i.e., the UA modifies the stream to be sendonly and mutes its own stream. Modification of the media session does not affect VoiceXML application execution (except that recognition actions initiated while on hold will result in noinput timeouts).

3.4. Audio and Video Codecs

For the purposes of achieving a basic level of interoperability, this section specifies a minimal subset of codecs and RTP [RFC3550] payload formats that MUST be supported by the VoiceXML Media Server.

For audio-only applications, G.711 mu-law and A-law MUST be supported using the RTP payload type 0 and 8 [RFC3551]. Other codecs and payload formats MAY be supported.

Video telephony applications, which employ a video stream in addition to the audio stream, are possible in VoiceXML 2.0/2.1 through the use of multimedia file container formats such as the .3gp [TS26244] and .mp4 formats [IEC14496-14]. Video support is optional for this specification. If video is supported then:

1. H.263 Baseline [RFC4629] MUST be supported. For legacy reasons, the 1996 version of H.263 MAY be supported using the RTP payload format defined in [RFC2190] (payload type 34 [RFC3551]).

2. Adaptive Multi-Rate (AMR) narrow band audio [RFC4867] SHOULD be supported.

3. MPEG-4 video [RFC3016] SHOULD be supported.

4. MPEG-4 Advanced Audio Coding (AAC) audio [RFC3016] SHOULD be supported.

5. Other codecs and payload formats MAY be supported.

Video record operations carried out by the VoiceXML Media Server typically require receipt of an intra-frame before the recording can commence. The VoiceXML Media Server SHOULD use the mechanism described in [RFC4585] to request that a new intra-frame be sent.

Since some applications may choose to transfer confidential information, the VoiceXML Media Server MUST support Secure RTP (SRTP) [RFC3711] as discussed in Section 9.

3.5. DTMF

DTMF events [RFC4733] MUST be supported. When the User Agent does not indicate support for [RFC4733], the VoiceXML Media Server MAY perform DTMF detection using other means such as detecting DTMF tones in the audio stream. Implementation note: the reason only [RFC4733] telephone-events must be used when the User Agent indicates support of it is to avoid the risk of double detection of DTMF if detection on the audio stream was simultaneously applied.

4. Returning Data to the Application Server

This section discusses the mechanisms for returning data (e.g., collected utterance or digit information) from the VoiceXML Media Server to the Application Server.

4.1. HTTP Mechanism

At any time during the execution of the VoiceXML application, data can be returned to the Application Server via HTTP using standard VoiceXML elements such as <submit> or <subdialog>. Notably, the <data> element in VoiceXML 2.1 [VXML21] allows data to be sent to the Application Server efficiently without requiring a VoiceXML page transition and is ideal for short VoiceXML applications such as "prompt and collect".

For most applications, it is necessary to correlate the information being passed over HTTP with a particular VoiceXML Session. One way this can be achieved is to include the SIP Call-ID (accessible in

VoiceXML via the session.connection.protocol.sip.headers array) within the HTTP POST fields. Alternatively, a unique "POST-back URI" can be specified as an application-specific URI parameter in the Request-URI of the initial INVITE (accessible in VoiceXML via the session.connection.protocol.sip.requesturi array).

Since some applications may choose to transfer confidential information, the VoiceXML Media Server MUST support the https: scheme as discussed in Section 9.

4.2. SIP Mechanism

Data can be returned to the Application Server via the expr or namelist attribute on <exit> or the namelist attribute on <disconnect>. A VoiceXML Media Server MUST support encoding of the expr/namelist data in the message body of a BYE request sent from the VoiceXML Media Server as a result of encountering the <exit> or <disconnect> element. A VoiceXML Media Server MAY support inclusion of the expr/namelist data in the message body of the 200 OK message in response to a received BYE request (i.e., when the VoiceXML application responds to the connection.disconnect.hangup event and subsequently executes an <exit> element with the expr or namelist attribute specified).

Note that sending expr/namelist data in the 200 OK response requires that the VoiceXML Media Server delay the final response to the received BYE request until the VoiceXML application’s post-disconnect final processing state terminates. This mechanism is subject to the constraint that the VoiceXML Media Server must respond before the User Agent Client’s (UAC’s) timer F expires (defaults to 32 seconds). Moreover, for unreliable transports, the UAC will retransmit the BYE request according to the rules of [RFC3261]. The VoiceXML Media Server SHOULD implement the recommendations of [RFC4320] regarding when to send the 100 Trying provisional response to the BYE request.

If a VoiceXML application executes a <disconnect> [VXML21] and then subsequently executes an <exit> with namelist information, the namelist information from the <exit> element is discarded.

Namelist variables are first converted to their "JSON value" equivalent [RFC4627] and encoded in the message body using the application/x-www-form-urlencoded format content type [HTML4]. The behavior resulting from specifying a recording variable in the namelist or an ECMAScript object with circular references is not defined. If the expr attribute is specified on the <exit> element instead of the namelist attribute, the reserved name __exit is used.

To allow the Application Server to differentiate between a BYE resulting from a <disconnect> from one resulting from an <exit>, the reserved name __reason is used, with a value of "disconnect" (without brackets) to reflect the use of VoiceXML’s <disconnect> element, and a value of "exit" (without brackets) to an explicit <exit> in the VoiceXML document. If the session terminates for other reasons (such as the media server encountering an error), this parameter may be omitted, or may take on platform-specific values prefixed with an underscore.

This specification extends the application/x-www-form-urlencoded by replacing non-ASCII characters with one or more octets of the UTF-8 representation of the character, with each octet in turn replaced by %HH, where HH represents the uppercase hexadecimal notation for the octet value and % is a literal character. As a consequence, the Content-Type header field in a BYE message containing expr/namelist data MUST be set to application/x-www-form-urlencoded;charset=utf-8.

The following table provides some examples of <exit> usage and the corresponding result content.

<exit> Usage	Result Content
<exit/>	__reason=exit
<exit expr="5"/>	__exit=5&__reason=exit
<exit expr="’done’"/>	__exit="done"&__reason=exit
<exit expr="userAuthorized"/>	__exit=true&__reason=exit
<exit namelist="pin errors"/>	pin=1234&errors=0&__reason=exit

assuming the following VoiceXML variables and values:

```
    userAuthorized = true
    pin = 1234
    errors = 0
```

For example, consider the VoiceXML snippet:

```
    ...
    <exit namelist="id pin"/>
    ...
```

If id equals 1234 and pin equals 9999, say, the BYE message would look similar to:

```
      BYE sip:user@pc33.example.com SIP/2.0
      Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
      Max-Forwards: 70
      From: sip:dialog@example.com;tag=a6c85cf
      To: sip:user@example.com;tag=1928301774
      Call-ID: a84b4c76e66710
      CSeq: 231 BYE
      Content-Type: application/x-www-form-urlencoded;charset=utf-8
      Content-Length: 30

      id=1234&pin=9999&__reason=exit
```

Since some applications may choose to transfer confidential information, the VoiceXML Media Server MUST support the S/MIME encoding of SIP message bodies as discussed in Section 9.

5. Outbound Calling

Outbound calls can be triggered via the Application Server using third-party call control [RFC3725].

Flow IV from [RFC3725] is recommended in conjunction with the VoiceXML Session preparation mechanism. This flow has several advantages over others, namely:

1. Selection of a VoiceXML Media Server and preparation of the VoiceXML application can occur before the call is placed to avoid the callee experiencing delays.

2. Avoidance of timing difficulties that could occur with other flows due to the time taken to fetch and parse the initial VoiceXML document.

3. The flow is IPv6 compatible.

An example flow for an Application-Server-initiated outbound call is provided in Section 2.6.2.

6. Call Transfer

While VoiceXML is at its core a dialog language, it also provides optional call transfer capability. VoiceXML's transfer capability is particularly suited to the PSTN IVR Service Node use case described in Section 1.1.2. It is NOT RECOMMENDED to use VoiceXML's call transfer capability in networks involving Application Servers. Rather, the Application Server itself can provide call routing functionality by taking signaling actions based on the data returned to it from the VoiceXML Media Server via HTTP or in the SIP BYE message.

If VoiceXML transfer is supported, the mechanism described in this section MUST be employed. The transfer flows specified here are selected on the basis that they provide the best interworking across a wide range of SIP devices. CCXML<->VoiceXML implementations, which require tight-coupling in the form of bidirectional eventing to support all transfer types defined in VoiceXML, may benefit from other approaches, such as the use of SIP event packages [RFC3265].

In what follows, the provisional responses have been omitted for clarity.

6.1. Blind

The blind-transfer sequence is initiated by the VoiceXML Media Server via a REFER message [RFC3515] on the original SIP dialog. The Refer-To header contains the URI for the called party, as specified via the dest or destexpr attributes on the VoiceXML <transfer> tag.

If the REFER request is accepted, in which case the VoiceXML Media Server will receive a 2xx response, the VoiceXML Media Server throws the connection.disconnect.transfer event and will terminate the VoiceXML Session with a BYE message. For blind transfers, implementations MAY use [RFC4488] to suppress the implicit subscription associated with the REFER message.

If the REFER request results in a non-2xx response, the <transfer>'s form item variable (or event raised) depends on the SIP response and is specified in the following table. Note that this indicates that the transfer request was rejected.

SIP Response	<transfer> variable / event
404 Not Found	error.connection.baddestination
405 Method Not Allowed	error.unsupported.transfer.blind
503 Service Unavailable	error.connection.noresource
(No response)	network_busy
(Other 3xx/4xx/5xx/6xx)	unknown

An example is illustrated below (provisional responses and NOTIFY messages corresponding to provisional responses have been omitted for clarity).

```
   User Agent 1        VoiceXML        User Agent 2
     (Caller)        Media Server        (Callee)
        |                  |                  |
        |(0) RTP/SRTP      |                  |
        |..................|                  |
        |                  |                  |
        |(1) REFER         | <transfer>       |
        |<-----------------|                  |
        |(2) 202 Accepted |                  |
        |----------------->|                  |
        |(3) BYE           |                  |
        |<-----------------|                  |
        |(4) 200 OK        |                  |
        |----------------->|                  |
        |                  | Stop RTP (0)     |
        |(5) INVITE                           |
        |------------------------------------>|
        |(6) 200 OK                           |
        |<------------------------------------|
        |(7) NOTIFY        |                  |
        |----------------->|                  |
        |(8) 200 OK        |                  |
        |<---------------- |                  |
        |(9) ACK                              |
        |------------------------------------>|
        |(10) RTP/SRTP                        |
        |.....................................|
        |                  |                  |
```

If the aai or aaiexpr attribute is present on <transfer>, it is appended to the Refer-To URI as a parameter named "aai" in the REFER method. Reserved characters are URL-encoded as required for SIP/SIPS URIs [RFC3261]. The mapping of values outside of the ASCII range is platform specific.

6.2. Bridge

The bridge transfer function results in the creation of a small multi-party session involving the Caller, the VoiceXML Media Server, and the Callee. The VoiceXML Media Server invites the Callee to the session and will eject the Callee if the transfer is terminated.

If the aai or aaiexpr attribute is present on <transfer>, it is appended to the Request-URI in the INVITE as a URI parameter named "aai". Reserved characters are URL-encoded as required for SIP/SIPS URIs [RFC3261]. The mapping of values outside of the ASCII range is platform specific.

During the transfer attempt, audio specified in the transferaudio attribute of <transfer> is streamed to User Agent 1. A VoiceXML Media Server MAY play early media received from the Callee to the Caller if the transferaudio attribute is omitted.

The bridge transfer sequence is illustrated below. The VoiceXML Media Server (acting as a UAC) makes a call to User Agent 2 with the same codecs used by User Agent 1. When the call setup is complete, RTP flows between User Agent 2 and the VoiceXML Media Server. This stream is mixed with User Agent 1’s.

```
User Agent 1         VoiceXML          User Agent 2
  (Caller)         Media Server          (Callee)
     |                  |                   |
     |(0)RTP/SRTP       |                   |
     |..................|                   |
     |                  |                   |
     |        <transfer>|(1)INVITE [offer]  |
     |                  |------------------>|
     |                  |(2) 200 OK [answer]|
     |                  |<------------------|
     |                  |(3) ACK            |
     |                  |------------------>|
     |                  |(4) RTP/SRTP       |
     |              mix |...................|
     |           (0)+(4)|                   |
```

If a final response is not received from User Agent 2 from the INVITE and the connecttimeout expires (specified as an attribute of <transfer>), the VoiceXML Media Server will issue a CANCEL to terminate the transaction and the <transfer>’s form item variable is set to noanswer.

If INVITE results in a non-2xx response, the <transfer>’s form item variable (or event raised) depends on the SIP response and is specified in the following table.

SIP Response	<transfer> variable / event
404 Not Found	error.connection.baddestination
405 Method Not Allowed	error.unsupported.transfer.bridge
408 Request Timeout	noanswer
486 Busy Here	busy
503 Service Unavailable	error.connection.noresource
(No response)	network_busy
(Other 3xx/4xx/5xx/6xx)	unknown

Once the transfer is established, the VoiceXML Media Server can "listen" to the media stream from User Agent 1 to perform speech or DTMF hotword, which when matched results in a near-end disconnect, i.e., the VoiceXML Media Server issues a BYE to User Agent 2 and the VoiceXML application continues with User Agent 1. A BYE will also be issued to User Agent 2 if the call duration exceeds the maximum duration specified in the maxtime attribute on <transfer>.

If User Agent 2 issues a BYE during the transfer, the transfer terminates and the VoiceXML <transfer>’s form item variable receives the value far_end_disconnect. If User Agent 1 issues a BYE during the transfer, the transfer terminates and the VoiceXML event connection.disconnect.transfer is thrown.

6.3. Consultation

The consultation transfer (also called attended transfer [RFC5359]) is similar to a blind transfer except that the outcome of the transfer call setup is known and the Caller is not dropped as a result of an unsuccessful transfer attempt.

Consultation transfer commences with the same flow as for bridge transfer except that the RTP streams are not mixed at step (4) and error.unsupported.transfer.consultation supplants error.unsupported.transfer.bridge. Assuming a new SIP dialog with User Agent 2 is created, the remainder of the sequence follows as illustrated below (provisional responses and NOTIFY messages corresponding to provisional responses have been omitted for clarity). Consultation transfer makes use of the Replaces: header [RFC3891] such that User Agent 1 calls User Agent 2 and replaces the latter’s SIP dialog with the VoiceXML Media Server with a new SIP dialog between the Caller and Callee.

```
   User Agent 1        VoiceXML       User Agent 2
     (Caller)       Media Server       (Callee)
        |                 |                 |
        |(0) RTP/SRTP     |                 |
        |.................|(4) RTP/SRTP     |
        |                 |.................|
        |(5) REFER        |                 |
        |<----------------|                 |
        |(6) 202 Accepted |                 |
        |---------------->|                 |
        |(7) INVITE Replaces:ms1.example.com|
        |---------------------------------->|
        |(8) 200 OK                         |
        |<----------------------------------|
        |(9) ACK                            |
        |---------------------------------->|
        |(10) RTP/SRTP                      |
        |...................................|
        |                 |(11) BYE         |
        |                 |<----------------|
        |                 |(12) 200 OK      |
        |                 |---------------->| Stop
        |(13) NOTIFY      |                 | RTP (4)
        |---------------->|                 |
        |(14) 200 OK      |                 |
        |<----------------|                 |
        |(15) BYE         |                 |
        |<----------------|                 |
        |(16) 200 OK      |                 |
        |---------------->| Stop            |
        |                 | RTP (0)         |
```

If a response other than 202 Accepted is received in response to the REFER request sent to User Agent 1, the transfer terminates and an error.unsupported.transfer.consultation event is raised. In addition, a BYE is sent to User Agent 2 to terminate the established outbound leg.

The VoiceXML Media Server uses receipt of a NOTIFY message with a sipfrag message of 200 OK to determine that the consultation transfer has succeeded. When this occurs, the connection.disconnect.transfer event will be thrown to the VoiceXML application, and a BYE is sent to User Agent 1 to terminate the session. A NOTIFY message with a non-2xx final response sipfrag message body will result in the transfer terminating and the associated VoiceXML input item variable being set to ’unknown’. Note that as a consequence of this mechanism, implementations MUST NOT use [RFC4488] to suppress the implicit subscription associated with the REFER message for consultation transfers.

7. Contributors

The bulk of the early work for this effort was carried out on weekly teleconferences and over email. The authors would particularly like to recognize the contributions of R. J. Auburn (Voxeo), Jeff Haynie (Hakano), and Scott McGlashan (Hewlett-Packard).

8. Acknowledgements

This document owes its genesis to, "A SIP Interface to VoiceXML Dialog Servers", authored by J. Rosenberg, P. Mataga, and D. Ladd. The following people had input to the current document:

R. J. Auburn (Voxeo)

Hans Bjurstrom (Hewlett-Packard)

Emily Candell (Comverse)

Peter Danielsen (Lucent)

Brian Frasca (Tellme)

Jeff Haynie (Hakano)

Scott McGlashan (Hewlett-Packard)

Matt Oshry (Tellme)

Rao Surapaneni (Tellme)

The authors would like to acknowledge the support of Cullen Jennings and the Mediactrl chairs, Eric Burger and Spencer Dawkins.

9. Security Considerations

Exposing a VoiceXML media service with a well-known address may enhance the possibility of exploitation (for example, an invoked network service may trigger a billing event). The VoiceXML Media Server is RECOMMENDED to use standard SIP mechanisms [RFC3261] to authenticate requesting endpoints and authorize per local policy.

Some applications may choose to transfer confidential information to or from the VoiceXML Media Server. To provide data confidentiality, the VoiceXML Media Server MUST implement the sips: and https: schemes in addition to S/MIME message body encoding as described in [RFC3261].

The VoiceXML Media Server MUST support Secure RTP (SRTP) [RFC3711] to provide confidentiality, authentication, and replay protection for RTP media streams (including RTCP control traffic).

To mitigate the possibility of denial-of-service attacks, the VoiceXML Media Server is RECOMMENDED (in addition to authenticating and authorizing endpoints described above) to provide mechanisms for implementing local policies such as the time-limiting of VoiceXML application execution.

10. IANA Considerations

IANA has registered the following parameters in the SIP/SIPS URI Parameters registry, following the Specification Required policy of [RFC3969]:

Parameter Name	Predefined Values	Reference
maxage	No	RFC 5552
maxstale	No	RFC 5552
method	"get" / "post"	RFC 5552
postbody	No	RFC 5552
ccxml	No	RFC 5552
aai	No	RFC 5552

11. References

11.1. Normative References

[HTML4] Raggett, D., Le Hors, A., and I. Jacobs, "HTML 4.01 Specification", W3C Recommendation, Dec 1999.

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

[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.

[RFC3016]   Kikuchi, Y., Nomura, T., Fukunaga, S., Matsui, Y., and H. Kimata, "RTP Payload Format for MPEG-4 Audio/Visual Streams", RFC 3016, November 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.

[RFC3264]   Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3265]   Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[RFC3311]   Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[RFC3326]   Schulzrinne, H., Oran, D., and G. Camarillo, "The Reason Header Field for the Session Initiation Protocol (SIP)", RFC 3326, December 2002.

[RFC3515]   Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[RFC3550]   Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3551]   Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

[RFC3711]   Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

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

[RFC3891]   Mahy, R., Biggs, B., and R. Dean, "The Session Initiation Protocol (SIP) "Replaces" Header", RFC 3891, September 2004.

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

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

   [RFC4320]      Sparks, R., "Actions Addressing Identified Issues with
                  the Session Initiation Protocol's (SIP) Non-INVITE
                  Transaction", RFC 4320, January 2006.

   [RFC4488]      Levin, O., "Suppression of Session Initiation Protocol
                  (SIP) REFER Method Implicit Subscription", RFC 4488,
                  May 2006.

   [RFC4585]      Ott, J., Wenger, S., Sato, N., Burmeister, C., and J.
                  Rey, "Extended RTP Profile for Real-time Transport
                  Control Protocol (RTCP)-Based Feedback (RTP/AVPF)",
                  RFC 4585, July 2006.

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

   [RFC4629]      Ott, H., Bormann, C., Sullivan, G., Wenger, S., and R.
                  Even, "RTP Payload Format for ITU-T Rec", RFC 4629,
                  January 2007.

   [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.

   [RFC4867]      Sjoberg, J., Westerlund, M., Lakaniemi, A., and Q.
                  Xie, "RTP Payload Format and File Storage Format for
                  the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate
                  Wideband (AMR-WB) Audio Codecs", RFC 4867, April 2007.

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

[VXML21]   Oshry, M., Auburn, R J., Baggia, P., Bodell, M., Burke, D., Burnett, D., Candell, E., Kilic, H., McGlashan, S., Lee, A., Porter, B., and K. Rehor, "Voice Extensible Markup Language (VoiceXML) Version 2.1", W3C Candidate Recommendation, June 2005.

11.2. Informative References

[CCXML10]   Auburn, R J., "Voice Browser Call Control: CCXML Version 1.0", W3C Working Draft, June 2005.

[IEC14496-14] "Information technology. Coding of audio-visual objects. MP4 file format", ISO/IEC ISO/IEC 14496-14:2003, October 2003.

[MRCPv2]   Shanmugham, S. and D. Burnett, "Media Resource Control Protocol Version 2 (MRCPv2)", Work in Progress, November 2008.

[RFC2190]   Zhu, C., "RTP Payload Format for H.263 Video Streams", RFC 2190, September 1997.

[RFC3960]   Camarillo, G. and H. Schulzrinne, "Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP)", RFC 3960, December 2004.

[RFC3969]   Camarillo, G., "The Internet Assigned Number Authority (IANA) Uniform Resource Identifier (URI) Parameter Registry for the Session Initiation Protocol (SIP)", BCP 99, RFC 3969, December 2004.

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

[RFC5359]   Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples", BCP 144, RFC 5359, October 2008.

[TS23002]   "3rd Generation Partnership Project: Network architecture (Release 6)", 3GPP TS 23.002 v6.6.0, December 2004.

[TS26244]   "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)", 3GPP TS 26.244 v6.4.0, December 2004.

Appendix A. Notes on Normative References

We make a "downref" normative reference to [RFC4627] -- an Informational document describing a proprietary (but extremely popular) format.

Authors’ Addresses

Dave Burke
Google
Belgrave House, 76 Buckingham Palace Road
London SW1W 9TQ
United Kingdom

EMail: daveburke@google.com

Mark Scott
Genesys
1120 Finch Avenue West, 8th floor
Toronto, Ontario M3J 3H7
Canada

EMail: Mark.Scott@genesyslab.com