VXI *

The VoiceXML Browser for Asterisk®

DEVELOPER GUIDE

Version: 5.2

i6net *

VXI* – VoiceXML Browser

DEVELOPER GUIDE

Company Name:

Address/City/State/Zip:

Country:

About I6NET

I6NET Solutions and Technologies Limited is a pan-European company specialized in the development of new applications and advanced communication solutions. I6NET creates new business solutions and opportunities with voice interactivity, helping phone and data networks convergence. Its innovative voice browsers systems and software components enable the creation of voice & video services in VoiceXML. You can contact us by email or call us by phone and leave us a message here.

Main/Sales office quarters:
C/ Magallanes 13 – 5º Izq 28015 Madrid (Spain)
VAT Number ES-B83388306 - See more information: www.i6net.com

Table of Contents

1 Purpose

This guide is intended for application developers who create VoiceXML applications on the Asterisk® platform with the I6NET VoiceXML browser, VXI*. It covers the VoiceXML 2.0 compliance and implementation specifications for VoiceXML on a browser. It also provides an introduction to VoiceXML concepts and describes how VoiceXML is used in the context of a Media Server.

2 Document Scope

This document is intended for individuals who develop VoiceXML pages for the I6NET browser. The intention of this document is not to override any standard specifications, such as the W3C VoiceXML 2.0 Specification. In case of errors or omissions within this document, the standard documents shall be assumed to be correct, unless explicitly stated otherwise.

It is recommended that developers have:

- Access to the standard documents
- Previously completed training in VoiceXML
- Advanced knowledge of the Asterisk PBX platform

Related Standard Documents

W3C VoiceXML	Voice Extensible Markup Language (VoiceXML) Version 2.0: http://www.w3.org/TR/voicexml20
W3C VoiceXML	French translation of VoiceXML Version 2.0: http://www.yoyodesign.org/doc/w3c/voicexml20/index.html
W3C Grammars	Speech Recognition Grammar Specification Version 1.0: http://www.w3.org/TR/speech-grammar
W3C SSML	Speech Synthesis Markup Language (SSML) Version 1.0: http://www.w3.org/TR/speech-synthesis
W3C Tutorial	VoiceXML Tutorial, "Getting started with VoiceXML 2.0": http://www.w3.org/Voice/Guide/
ECMAScript	Standard ECMA-262, ECMAScript Language Specification: http://www.ecma-international.org/publications/standards/Ecma-262.htm
ZVON VoiceXML	Online VoiceXML 2.0 Reference: http://www.zvon.org/xxl/VoiceXML2.0Reference/Output/index.html

Key Features

Feature	Description
Record	Record audio and video files.
Record echo	In video mode, you have a video echo during the record.
Play	Play different file formats.
Play VCR	Enable to play with VCR DTMF controls (forward, backward).
Overdial	Enable to generate DTMF events (inband or RFC).
Transfer	Enable to transfer a call with 2 modes: bridge (new session or reinvite, and blind with SIP refer.
Chat / Gtalk	With a specific modified chan_gtalk the user can interact with the chat interface.
Video silence	A video loop sequence can be automatically generated during the inputs waits.
Text-to-Speech TTS	Converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.
Text-to-Video TTV	Converts normal text into a video static image or with and avatar animation.
Automatic Speech Recognition ASR	Converts spoken words to machine-readable input.

Supported File Formats

Formats	Description
WAV	Wav format with codec PCM 16bits 8kHz
wav	Wav format with codec GSM V6.1
gsm	GSM format
alaw	Raw alaw format
ulaw	Raw µlaw format
3gp	174x144 10 fps codec h263 / AMRNB (hinted)
mp4	174x144 (for Video IP/3G)

Protocols and Standards

Protocols	RFC number
MIME	RFC 1341
RTP	RFC 1889, 1890, 2833, 2190, 2429/4229, 3984
SIP	RFC 2543, RFC 2976, “The SIP Info Method”
DSP	RFC 2327
URI	RFC 2396
URL	RFC 1738

3 VoiceXML Functions Specification

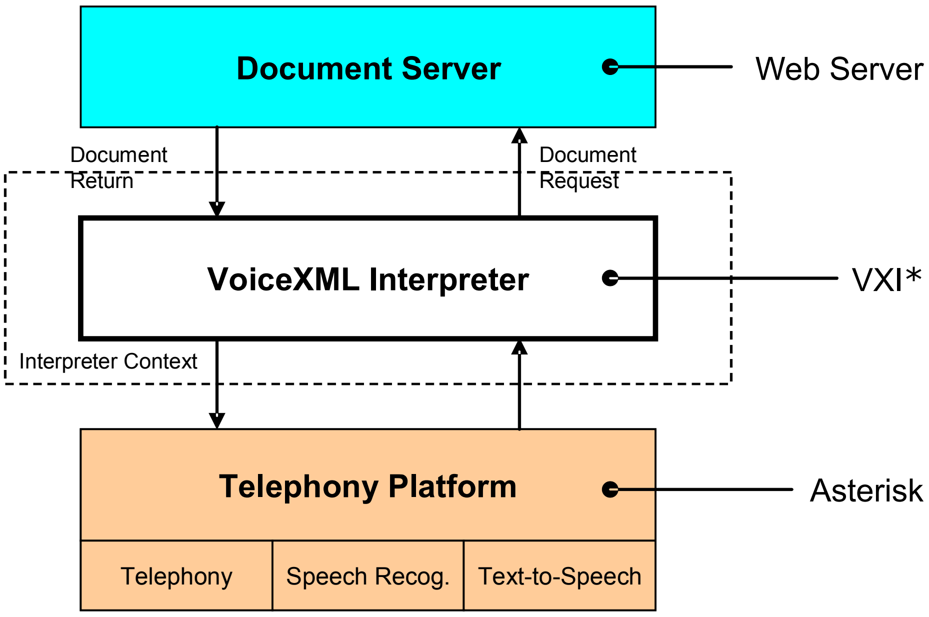
3.1 VoiceXML Overview

What is VoiceXML?

VoiceXML (or simply VXML) stands for Voice Extensible Markup Language. Much as HTML is a markup language for creating distributed textual/visual applications, VoiceXML is an XML based markup language for creating distributed voice applications. Using VoiceXML, one can develop web-based applications that users can access from a telephone, using their voice. Simply by "talking" to a VoiceXML application, a user can call into a web site and access the very same information and services, instead of having to be logged into the Internet.

Architectural Model

The distribuited application model of the Voice browser (VXI*) for Asterisk assumed by VoiceXML is a system based on three levels with the following architecture:



When a call is received, the implementation platform (telephony platform) sends an event to the VoiceXML interpreter, which in turn looks in its context for the URI of the initial document to fetch. A request is then sent to the document server for the initial document. The document server in turn sends the document to the VoiceXML interpreter that interprets the document and executes it appropriately using the services of the telephony platform.

The interpretation may result in a message to be prompted to the caller through the implementation platform or the VoiceXML interpreter making additional document requests to the document server. Under the document's control, the VoiceXML interpreter directs the implementation platform to:

- Send prompts, messages, or other audio/video material to the user
- Accept numeric input that the user enters by DTMF (Touch Tone©) signals)
- Accept voice input and recognize the words by receiving grammar data dynamically
- Simply record voice and video input
- Send the user's information to a Web site or other Internet server
- Receive information from the Internet, and pass it to the user

For instance, in an interactive voice response application, the VoiceXML interpreter context may be responsible for detecting an incoming call, acquiring the initial VoiceXML document, and answering the call, while the VoiceXML interpreter conducts the dialog after the answer.

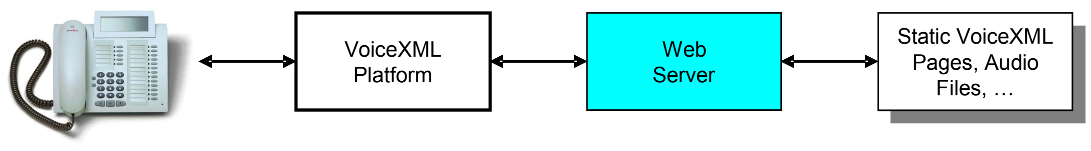
The implementation platform generates events in response to user actions (e.g. spoken or character input received, disconnect) and system events (e.g. timer expiration). Some of these events are acted upon the VoiceXML interpreter itself, as specified by the VoiceXML document, while others are acted upon the interpreter context.

VoiceXML applications are comprised of a collection of documents. You will recall that a document is the equivalent of an HTML page, and encapsulates one or more dialogs. Execution of a dialog typically involves presentation of information to the caller, along with collection of input from that caller. Transition from one dialog to another is controlled by the currently executing dialog. Transition occurs via the <goto> or <submit> tags.

Two application models exist to create a VoiceXML service. While simpler applications may use static VoiceXML pages, nearly all rely on dynamic VoiceXML page generation using an application server.

Static Applications

The simplest VoiceXML application is simply going to be a set of VoiceXML pages, returned by a web server to the VoiceXML platform. These pages are interpreted to control the interaction with the user.



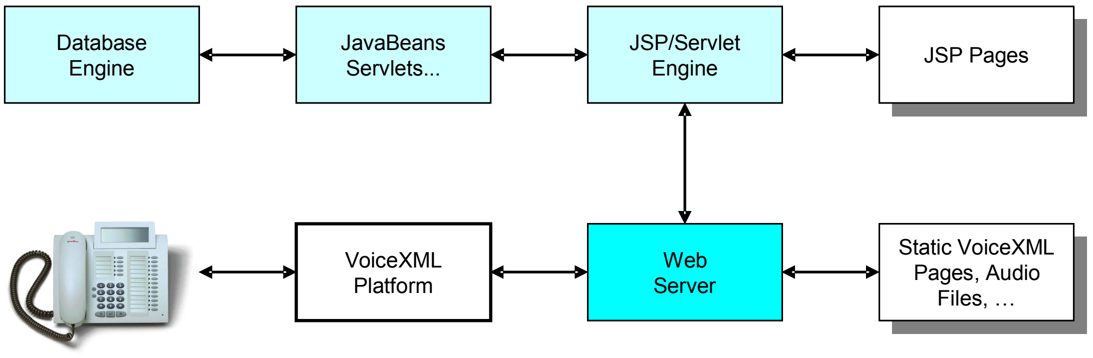
NOTE: The pages can be hosted on a Web Server or stored and referenced directly in the local disk.

Below is a Static VoiceXML page example:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <block>
      <prompt>
        Hello world!
      </prompt>
    </block>
  </form>
</vxml>
```

Dynamic Applications

Dynamic applications function in much the same way as Static ones, with the exception that some or all of the pages are generated dynamically, by a server-based technology such as JSP/JavaBeans, ASP/ActiveX, ColdFusion, or scripting languages such as perl. Typically, HTTP is used as the transport protocol for fetching VoiceXML pages. An overview of this model is shown below.



In a well-architected web application, the voice interface and the visual interface share the same back-end business logic. VoiceXML dynamic applications use same frameworks as Web applications. The content generated can be the result of several interactions with a database, a directory or an application interface (billing system, xml/http server…).

3.2 VoiceXML Concepts

Main concepts of VoiceXML language are:

Sessions

A session begins once the user begins to interact with a VoiceXML document.

Applications

An application is a collection of VoiceXML documents. All the documents in an application share the same application root document.

Forms

A form in a VoiceXML document presents information and gathers input from the user.

Menus

A menu gives the user a list of choices to select from and transitions to a different dialog or document based on the user's choice.

Links

A link specifies a transition that is common to all dialogs in the scope of the link. When a user input matches the link's grammar, control transfers to the link's destination.

Grammars

A grammar specifies a list of permissible vocabulary (words and phrases) for the user to select from in order to interact with the VoiceXML application. Each dialog has one or more speech and/or grammars associated with it.

Events

An event is thrown by the VoiceXML platform for a number of reasons, such as when a user does not respond to an input, does not respond correctly, or requests help, etc. The VoiceXML interpreter also throws events in the case that there are semantic errors in the VoiceXML document.

Applications

Below are a few examples in which VoiceXML applications can be used:

Voice Portals Services

Just like Web portals, voice portals can be used to provide personalized services to access information like stock quotes, weather, restaurant listings, news, etc.

Location-based Services

Location-based services allow you to receive targeted information specific to the location you are dialing from. Applications use the telephone number you are dialing from (the ANI) to deliver pertinent promotions or data.

Voice Alerts Services

VoiceXML can be used to send targeted alerts to a user. The user would sign up to receive special alerts informing him of upcoming events.

Commerce Services

VoiceXML can be used to implement applications that allow users to purchase over the phone. Because voice gives you less information than graphics, specific products that don't need a lot of description (such as tickets, CDs, office supplies, etc.) work well.

Following is an example of how to create a simple “Hello World!” VoiceXML application:

```
<?xml version="1.0"?>
 <vxml version="1.0">
  <form>
   <block>Hello world!</block>
 </form>
</vxml>
```

Forms

A form in a VoiceXML document presents information and gathers input from the user. A form is represented by the <form> tag and has an ID attribute associated with it. The ID attribute is the name of the form.

Following is an example of a form element:

```
<form id="hello" >
 <block>
  Hello world!
 </block>
</form>
```

NOTE:

In this example, the name of the form is "hello" and "Hello world!" is presented to the user.

Form Items

There are two types of form items, field items and control items. A field item prompts the user on what to say or key in, and then collects that information and fills in the field item variable. A field item also has grammars that define the allowed inputs, event handlers to process the resulting events, and a <filled> element that defines an action to be taken after the field item variable has been filled.

Following is a list of the types of field items:

<field>

The <field> form is the value of the field item obtained from the user via speech or DTMF.

<record>

The <record> form is the value of the field item which is an audio clip recorded by the user, such as a voice mail message, which can be collected by the <record> element.

<transfer>

the <transfer> form is used for transferring the user to another telephone number.

<object>

The <object> form invokes platform-specific objects with one or more properties.

<subdialog>

Like a function call, the <subdialog> form invokes a call to another dialog on the current page or another VoiceXML document.

The task of a control item is to help control the gathering of the form's fields.

Following are two types of control items:

<block>

The <block> control is a sequence of statements used for prompting and computation.

<initial>

The <initial> control is useful in mixed initiative dialogs that prompt the user for information

Variables and Conditions

A form item variable is associated with each form. The form item 'variable by default' is initially set to 'undefined' and contains a result (collected from the user) once a form item has been interpreted. You can define the name of a form item variable by using the name attribute. A guard condition exists for each form item. The guard condition tests whether the item's variable currently has a value. If a value exists, then the form item is skipped.

Menus

A menu gives the user a list of choices to select from and transitions to a different dialog or document based on the user's choice.

Following is an example of a menu:

```
<menu>
 <prompt>Say what sports news you are interested in:
  <enumerate/>
 </prompt>
 <choice dtmf="1" next="http://www.news.com/hockey.vxml">
  Hockey
 </choice>
 <choice dtmf="2" next="http://www.news.com/baseball.vxml">
  Baseball
 </choice>
 <choice dtmf="3"  next="http://www.news.com/football.vxml">
  Football
 </choice>
 <noinput>
  Please say what sports news you are interested in
  <enumerate/>
 </noinput>
</menu>
```

The menu element supports the following attributes:

id

The id attribute identifies the menu.

scope

The scope attribute identifies the scope of the menu's grammar.

dtmf

If choices in a menu don't have explicit DTMF elements, they are given implicit ones "1", "2", etc. This is only if DTMF is set to 'true'.

The choice element specifies the URL to go based on the choice selected from the menu. The enumerate element specifies a template that is applied to each choice in the order they appear in the menu. So, for the above code example, the menu's prompt would be, "Please say what sports news you are interested in listening: hockey, baseball, or football.”

Links

A <link> element specifies one or more grammars. When one of these grammars is matched, the link is activated and either transitions to the destination specified, or throws an event.

Following is an example of the usage of the link element:

```
<link next="http://www.news.com/hockey.vxml">
 <grammar type="application/x-jsgf"> 1 {red} | 2 {yellow} <grammar>
</link>
```

The link is activated when you say "red" or press "1". The next attribute of the link element specifies the appropriate destination.

Grammars

Speech Grammars – Grammar Elements

A speech grammar specifies a list of vocabulary for the user to select in order to interact with the VoiceXML application.

Following is an example of how you could define a speech grammar:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form id="yyy">
<field name="xxx">
<grammar>
<![CDATA[
[
[excellent]
[good]
[fair]
[poor]
[help]
]
]]>
</grammar>
</field>
</form>
</vxml>
```

DTMF Grammars – DTMF Elements

A DTMF defines a set of key presses for the user to supply when interacting with the VoiceXML application. Like speech grammars, the DTMF grammar can be either inline or external.

Following is an example of how you could define a DTMF grammar:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form id="yyy">
<field name="xxx">
<grammar>
<![CDATA[
[
[excellent dtmf-1]
[good dtmf-2]
[fair dtmf-3]
[poor dtmf-4]
[help dtmf-5]
]
]]>
</grammar>
</field>
</form>
</vxml>
```

Events

Events Types

An event is thrown by the VoiceXML platform for any type of reason, such as a user not responding to an input, not responding correctly, or requesting help, etc. An event is also thrown if there is a semantic error in the VoiceXML document, or when the <throw> element is encountered. The <throw> element generates an event (user-defined or system) and the <catch> element catches the event thrown by the VoiceXML document, dialog, or form item.

<throw>

The <throw> element generates predefined or user-defined events.

Following is an example of the use of the <throw> element:

```
<throw event="nomatch"/>
```

In this case, an event is generated when an input by the user is not recognized as part of the active grammar. The attribute for the <throw> element is event, which defines the event to be thrown.

<catch>

The <catch> element catches the event thrown by the VoiceXML document, dialog, or form item.

Folowing is an example of the use of the <catch> element:

```
<catch event="nomatch"/>
<throw event="event.password.invalid"/>
</catch>
```

Following is a set of available <catch> elements:

<error>

The <error> catch element catches events of type error.

<help>

The <help> catch element catches events if no help is available.

<noinput>

The <noinput> catch element catches events if there was no input by the user.

<nomatch>

The <nomatch> catches events if an input by the user is not recognized as part of the active grammar.

4 VoiceXML Reference Guide

VoiceXML language referenced language elements:

Events

List of standard events supported.

Errors

List of errors generated.

Variables

List of variables supported.

Properties

List of properties supported.

Elements

List of VoiceXML elements supported.

4.1 Events

Following is the list of events that are supported by the VoiceXML Browser:

exit

The user has asked to exit. This is available when the universals property includes exit.

help

The user has asked for help. This is available when the universals property includes help.

noinput

The user has not responded within the timeout interval.

nomatch

The user inputs something, but it was not recognized.

maxspeechtimeout

The user input was too long, exceeding the property value (not supported for speech recognition).

connection..disconnect.hangup

The caller hangs up.

connection.disconnect.transfer

The call was "blind transferred" to another line and will not return.

4.2 Errors

Following is the list of errors that are generated by the VoiceXML Browser:

error.badfetch

The interpreter context throws this event when a fetch of a resource has failed and the interpreter context has reached a place in the document interpretation where the fetch result is required.

error.badfetch

In case of a fetch failure, the interpreter context returns the specific HTTP response code, http.<response code>.

error.semantic

A run-time error was found in the VoiceXML document.

error.noauthorization

The user is not authorized to perform the requested operation.

error.unsupported.format

The requested resource has a format that is not supported by the platform, (not supported yet).

error.unsupported.language

The platform does not support the language for either speech synthesis or speech recognition, (for TTS/ASR, not supported yet).

error.telephone.noauthorization

The caller is not allowed to call the destination.

error.telephone.baddestination

Destination URI is malformed.

error.telephone.noroute

The platform is not able to place a call to the destination.

error.telephone.noresource

The platform cannot allocate a resource to place the call.

error.grammar

There is an error in the DTMF grammar.

error.application

Thrown when there is a miscellaneous application error.

error.script

There is an error with ECMAScript.

error.internal

Internal interpreter server error.

4.3 Variables

VoiceXML variables and ECMAScript variables share the same variable space. In other words, variables declared in a <script> element can be used in VoiceXML, and ECMAScript can use variables declared by a <var> element.

The variables can be declared in 3 different ways:

- By a <var> element.
- Within the <script> element in an ECMAScript block.
- Field item variables declared by one of the form items: <block>, <field>, <initial>, <object>, <record>, <subdialog>, <transfer>, <data>

Variables Scopes:

VoiceXML defines the following variable scopes:

“Session”

Session variables are available to all VoiceXML applications within a particular session. They are declared by the interpreter and are read-only. New session variables cannot be declared by VoiceXML documents.

“Application”

Application variables are available to the root document and its application leaf documents. They are declared by the <var> elements of application root document's <vxml> element.

“Document”

Document variables are available within the VoiceXML document. They are declared by the <var> elements that are children of the <vxml> element. Certain form items (<field>, <record>, <transfer>) may declare shadow variables in addition to the field item variable itself. Please refer to each tag reference for detailed information.

“Dialog”

Each dialog (<form> or <menu>) has a dialog scope that exists while the user is visiting the dialog. Variables are declared by <var> elements or by form items.

“Anonymous”

<block>, <filled>, and <catch> (includes <error>, <help>, <noinput>, and <nomatch>) elements define a new anonymous scope to contain variables declared in that element. Other predefined anonymous variables are available within the <catch> and <enumerate> elements. Please refer to the tag reference for detailed information.

Session Variables

The following are standard VoiceXML session variables:

session.connection.local.uri

This variable refers to the DNIS (Dialed Number Identification Service). It provides the telephone number dialed by the caller if the service is supported, (alias session.telephone.dnis).

session.connection.remote.uri

This variable refers to the ANI (Automatic Number Identification). It provides the telephone number of the caller if the service is supported, (alias session.telephone.ani).

The following are session variables added by the I6NET VoiceXML browser:

session.telephone.session

This variable indicates the session/channel ID.

session.telephone.id

This variable contents the call ID or a user parameter assigned by the Asterisk Extension script.

session.telephone.param

This variable contents a user paramer assigned by the Asterisk Extension script.

Application Variables

Application variables defined in the root document can be referenced by the application leaf documents as application.variable.The following are standard VoiceXML application variables:

application.lastresult$[i]

This read-only variable holds information about the last recognition to occur within this application. It is an array of elements, application.lastresult$[i], for the N-best recognition.

application.lastresult$[i].confidence

This variable is the confidence level for this interpretation from 0.0 (minimum confidence) to 1.0 (maximum confidence).

application.lastresult$[i].utterance

This variable indicates the raw string of words that were recognized for this interpretation.

application.lastresult$[i].inputmode

The input mode variable is the mode in which user input was provided: DTMF or Voice.

application.lastresult$[i].interpretation

The last result variable is the interpretation of this result. It will be the same as raw results for index 0 when no slot match was found, and for any index > 0.

4.4 Properties

Properties are used to set values that affect platform behavior. Properties apply to their parent element and all the descendants of the parent. A property at a lower level overrides a property at a higher level. Additional properties are supported with Call Control extensions.

DTMF Recognition Properties

The following are the DTMF recognition properties:

interdigittimeout

This property indicates the timeout period allowed between each digit when recognizing DTMF input.

The default is 3 seconds.

termtimeout

This property determines the terminating timeout to use when recognizing DTMF input. Allows entering the optional termchar. The default is 0 seconds.

termchar

This property is the terminating DTMF character for DTMF input recognition. The default is #.

Speech Recognition Properties

The following are the speech recognition properties:

confidencelevel

This property defines the threshold of the speech recognition confidence level. Values range from 0.0 (minimum confidence) to 1.0 (maximum confidence). User inputs are ignored if the confidence level is below this threshold. The default is 0.5.

sensitivity

The sensitivity property defines the sensitivity level to input. Values range from 0.0 (the least sensitive to background noise) to 1.0 (highly sensitive to noise input). The default is 0.5.

completetimeout

This property indicates the length of silence required following user speech, before the speech recognizer finalizes a result, either accepting it or throwing a nomatch event. The complete timeout is used when the speech result is a complete match of an active grammar. The default is 1 second.

incompletetimeout

The incompletetimeout property is used when the speech is an incomplete match to an active grammar. The default is 1 second.

maxspeechtimeout

The maxspeechtimeout property is the maximum duration of user speech. If the time designated has elapsed before the user stops speaking, the maxspeechtimeout event is thrown. The default is 0 and indicates no limit.

maxnbest

The maxnbest property is the maximum number of N-Best results returned by the recognizer. This also represents the maximum size of the application.lastresult$[i] array. The value is always 1 for the I6NET interpreter.

Prompt and Collect Properties

The following are the prompt and collect properties:

inputmodes

The input modes determines which input methods to use. The value is a space-separated list of input methods.

The default is dtmf voice:

* “dtmf” - allows DTMF sequences as input.

* “voice” - allows voice as input.

timeout

The timeout is the time after which the interpreter throws a noinput event. This property is not used in DTMF interactions. The default is 10 seconds.

universals

The universals property specifies universal commands. The value is a space-separated list of the following commands. You can designate none for nothing or all for all 3 commands.

This property is not supported yet.

* cancel - throws the cancel event.

* exit - throws the exit event.

* help - throws the help event.

bargein

Enables user input during prompts if set to true. No barge-in if set to false. The default is true.

bargeintype

The bargeintype property specifies the barge-in type listed below.

This property is not supported yet.

* energy - any noise can barge in the prompt.

* speech - any user utterance can barge in the prompt.

* recognition - only barge-in on the prompt when the user input matches a speech grammar.

Fetching Properties

The following are the fetching properties:

audiofetchhint

This property defines when audio files can be fetched:

* prefetch – the file can be fetched when the document is loaded.

* safe – the file is fetched only when needed, never before prefetch.

audiomaxage

This property defines the maximum acceptable age, in seconds, of cached audio resources.

audiomaxstale

Use this property to define the maximum staleness, in seconds, of expired cached audio resources.

promptmaxage

Promptmax age defines the maximum acceptable age, in seconds, of cached TTS audio resources.

promptmaxstale

This property defines the maximum staleness, in seconds, of expired cached TTS audio resources.

documentfetchhint

The documentfetchhint defines when a document can be fetched:

* prefetch – the file can be fetched when the document is loaded.

* safe – the file is fetched only when needed, never before prefetch.

documentmaxage

This property defines the maximum acceptable age, in seconds, of cached documents.

documentmaxstale

Documentmaxstale defines the maximum staleness, in seconds, of expired cached documents.

grammarfetchhint

This property defines when grammar files can be fetched.

This property is not supported yet.

* prefetch – the file can be fetched when the document is loaded.

* safe - the file is fetched only when needed, never before prefetch.

grammarmaxage

This defines the maximum acceptable age, in seconds, of cached grammar resources.

This property is not supported yet.

grammarmaxstale

Grammarmaxstyle defines maximum staleness, in seconds, of expired cached grammar resources.

This property is not supported yet.

objectfetchhint

This property defines when objects can be fetched:

* prefetch - the file can be fetched when the document is loaded.

* safe - the file is fetched only when needed, never before prefetch.

objectmaxage

Objectmax age defines the maximum acceptable age, in seconds, of cached object resources.

objectmaxstale

This property defines the maximum staleness, in seconds, of expired cached object resources.

scriptfetchhint

The scriptfetchhint property defines when script files can be fetched:

* prefetch - the file can be fetched when the document is loaded.

* safe - the file is fetched only when needed, never before prefetch.

scriptmaxage

This property defines the maximum acceptable age, in seconds, of cached script resources.

scriptmaxstale

This property defines the maximum staleness, in seconds, of expired cached script resources.

fetchaudio

Fetchaudiodefines the URI of audio to play while waiting for a document to be fetched.

fetchtimeout

This property defines the timeout for fetches.

The default is 7.

Platform Properties

The following properties are specific to the I6NET VoiceXML browser platform and they are useful for debugging purposes. This property is not yet available.

4.5 Elements

The following is the Elements Quick Reference Table:

Element	Description	Supported
<assign>	Assigns a value to a variable.	yes
<audio>	Plays an audio file to a user.	yes
<block>	Allows execution of code within this item.	yes
<catch>	Handles (or catches) events.	yes
<choice>	Provides definition of a menu.	yes
<clear>	Clears or resets form items (form fields).	yes
<data>	VoiceXML 2.1	no
<disconnect>	Terminates application and hang up.	yes
<else>	Conditional code, associated with <if>.	yes
<elseif>	Conditional code, associated with <if>.	yes
<enumerate>	Builds choice list.	yes
<error>	Catches error event.	yes
<exit>	Terminates application but keeps port alive.	yes*
<field>	Collects user input.	yes
<filled>	Defines code when user input is complete.	Yes
<foreach>	VoiceXML 2.1	no
<form>	Defines dialog for collecting user input.	yes
<goto>	Jumps to another dialog.	yes
<grammar>	Defines DTMF user input rules.	yes*
<help>	Catches help event.	yes
<if>	Defines conditional code.	yes
<initial>	Defines prompt for initial form level input.	no
<link>	Defines URL to go to when a choice is made.	yes*

Element	Description	Supported
<log>	Output debug message.	yes
<menu>	Dialog for selecting from fixed choices.	yes
<meta>	Defines page information.	Yes*
<metadata>	Defines page information.	no
<noinput>	Catches a user input timeout event.	yes
<nomatch>	Catches an invalid user input event.	yes
<object>	Executes an extension function.	yes
<option>	Defines a field selection.	yes
<param>	Defines parameter to subdialog or object.	yes
<prompt>	Defines output to be played to user.	yes
<property>	Allows internal setting to be configured.	yes
<record>	Makes user audio recording to the file.	yes
<reprompt>	Re-asks for user input after prompting.	yes
<return>	Return from a subdialog.	yes
<script>	Executes ECMAScript (javascript) code.	yes
<subdialog>	Invokes another dialog and then returns.	yes
<submit>	Sends application values and gets new doc.	yes
<throw>	Generates an event to be handled by catch.	yes
<value>	Inserts value of variable into text.	yes
<var>	Declares a variable.	yes
<vxml>	Defines root element of document.	yes

*See the specific tag below for limitation/restrictions.

<assign>

Description

The <assign> element assigns a value to a variable. It is illegal to make an assignment to a variable that has not been explicitly declared using a <var> element or a <var> statement within a <script>. Attempting to assign an undeclared variable causes an error.semantic event to be thrown.

Syntax

```
<assign
name="string"
expr="ECMAScript_Expression"/>
```

Attributes

name

The name of the variable being assigned to. This is required.

expr

The value to assign to the variable. This is required.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <var name="tree" expr="'an apple tree'"/>
 <form>
  <block>
   <prompt> The initial value is <value expr="tree"/> </prompt>
   <assign name="tree" expr="'apple orchards'"/>
   <prompt> The new value is <value expr="tree"/> </prompt>
  </block>
 </form>
</vxml>
```

<audio>

Description

The <audio> element plays an audio file. If the file cannot be played, the content of the <audio> element is played instead. The alternate content may be text, speech markup, or another <audio> element.

Syntax

```
<audio
src="URI"
expr="ECMAScript_Expression"
fetchtimeout="time_interval"
fetchhint="prefetch" | "safe"
maxage="time_interval"
maxstale="time_interval"
>
alternate text
</audio>
```

Attributes

src

Src is the URI of the audio file. Either src or expr is required.

expr

Expr is an ECMAScript expression for the URI of the audio file. If the src attribute is specified, it takes precedence over this attribute. If <audio> contains this attribute that evaluates to null, this element will be skipped, including its alternate content. Either src or expr is required.

fetchtimeout

This attribute indicates the time interval to wait for an audio file to be fetched before playing the alternate content.

This is optional.

fetchhint

Fetchhint defines when the audio file should be retrieved. This is optional.

* prefetch - the file can be fetched when the document is loaded.

* safe - the file is fetched only when needed, never before prefetch.

* stream - allows audio streaming with HTTP.

maxage

This attribute indicates the maximum time in seconds that this document will use this audio file before fetching another copy.

This is optional.

maxstale

This attribute indicates the maximum time in seconds that this document will use an audio file.

Parents

<audio>, <block>, <catch>, <choice>, <enumerate>, <error>, <field>, <filled>, <if>, <initial>, <menu>, <noinput>, <nomatch>, <object>, <prompt>, <record>, <subdialog>, <transfer>

Children

<audio>, <value>, <enumerate>, Speech Markup (SSML)

Extensions

If "dial" is added as a URI prefix it will initiate DTMF generation.

Example:

<audio src="dial:123" /> or <audio expr="'dial:'+number" />

If the file is prefixed with “uri,” the Asterisk application will use this audio file base. Audio files are stored in /var/lib/asterisk/sounds, and the filename is passed without this extension.

If the file has an extension of h263 or h264, a video clip is generated in the background during an audio-only prompt. Note: the video will have no sound. (feature disabled for the moment)

Example:

<audio src="background.h263" />

If the file is a video clip with the 3gp or mp4 formats (see sip.fontventa.com), then the video clip will play on video phones.

Example:

<audio src="video.mp4" />

If the URI is prefixed also with “uri:rstp://…”, then the VXI* will plays the RTSP video stream.

Example:

<audio src="uri:rtsp://server.com/samples/video.mp4" />

If the property “control” is set to “true” (<property name=”control” value=”true”/>), the GSM and wav prompt can be controlled with VCR type controls like DTMF keys. (See the control keys configured in the configuration file). Start position can be set adding “:” and the value of the offset. After the prompt the position can be get with the object “offset” (see the tag <object>).

Example:

<audio src="samples/video.wav:1235" />

If the URI is prefixed with “cli:”, then the VXI* will execute the command in the src/expr attribute (remembre that this attribut don't support space, so replace them by underscore (“ “ → “_”). The VoiceXML browser disable the CLI execution by default. Enable it if needed by setting the parameter cli in the general section to “yes”.

Example:

<audio src="cli:stop_now" />

If the URI is prefixed with “exec:”, “application:” or “app:”, then the VXI* will execute the application and the parameters specified in the src/expr attribute.

Example:

<audio src="app:saynumber(123)" />

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <block>
   <prompt>
    Playing the recorded message
   </prompt>
   <audio src="thankyou.wav">
    Reading this if thankyou.wav is not found
   </audio>
  </block>
 </form>
</vxml>
```

<block>

Description

The <block> element specifies a block of directives to be executed in the order given.

Syntax

```
<block
name="String"
expr="ECMAScript_Expression"
cond="ECMAScript_Expression">
child elements
</block>
```

Attributes

name

This is the name of the block that can be referred within the form. This is optional and defaults to an inaccessible internal variable.

expr

This attribute indicates the initial value of the block. The block will be visited only if the expression evaluates to “undefined.” This is optional and defaults to undefined.

cond

This attribute is a Boolean condition that must evaluate to “true” in order for this block to be visited. This is optional and defaults to true.

Parents

<form>

Children

<assign>, <audio>, <clear>, <disconnect>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <block name="first">
   <prompt>This is a block</prompt>
   <goto nextitem="second"/>
  </block>
  <block name="second">
   <prompt>This is another block</prompt>
  </block>
 </form>
</vxml>
```

<catch>

Description

The <catch> element catches events thrown from the VoiceXML application or the platform. The <catch> element associates a catch with a document, dialog or form item and contains executable content. The <catch> element catches an event with the name that either matches exactly with the event attribute or a prefix match. A prefix match means the event attribute is a token prefix of the thrown event, where the dot is the token separator.

For example, <catch event="telephone.disconnect"> is a prefix match for event telephone.disconnect.transfer. Please refer to Event Handling for the list of events and errors.

Syntax

```
<catch
event="event1 event2 ..."
count="Integer"
cond="ECMAScript_Expression">
child elements
</catch>
```

Attributes

event

This attribute indicates the event or events to catch. A space-separated list of events may be specified to catch multiple events. The empty string matches all events. This attribute is required.

count

The count attribute allows you to handle different occurrences of the same event differently. Each <form>, <menu> and form item maintains a counter for each event that occurs while it is being visited. These counters are reset each time the <form> or <menu> is re-entered. When there is more than one <catch> element catching the same event, it will visit the element with a smallest count that is greater or equal to the current counter. This attribute is optional.

cond

The cond attribute is a Boolean condition that must evaluate to true in order for the <catch> element to catch the event. This attribute is optional.

Anonymous Variables

There are two anonymous variables that are available within the scope of the <catch> element:

* _event - the variable that contains the name of the event that was thrown.

* _message - the variable that contains the message string from the corresponding <throw> element, or a platform-defined value for events raised by the platform.

Parents

<field>, <form>, <initial>, <menu>, <object>, <record>, <subdialog>, <transfer>, <vxml>

Children

<assign>, <audio>, <clear>, <disconnect>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <grammar> goodbye </grammar>
  <catch event="goodbye">
   Thanks for using this script, goodbye.
   <disconnect/>
  </catch>
  <field name="password">
    <prompt> what is the code word </prompt>
    <grammar> apple </grammar>
    <help> It is the name of a fruit </help>
    <catch event="noinput"> I did not hear you. </catch>
    <catch event="nomatch" count="1"> Noop. Try again </catch>
    <catch event="nomatch" count="2"> Noop. give another try </catch>
    <catch event="nomatch" count="3">
     Sorry. You didn't get it for three times. Bye
     <disconnect/>
    </catch>
   <filled>
    <if cond="password=='goodbye'">
     <throw event="goodbye"/>
    <else/>
     This is correct.
    </if>
   </filled>
  </field>
 </form>
</vxml>
```

<choice>

Description

The <choice> element defines a menu item and serves several purposes:

* Specifies a speech grammar fragment and/or a DTMF grammar fragment that determines when that choice has been selected.

* Forms the <enumerate> prompt string with its contents.

* Specifies the URI to go to when the choice is selected.

Syntax

```
<choice
dtmf="DTMF sequence"
accept="exact" | "approximate"
next="URI"
event="event"
expr="ECMAScript_Expression"
fetchaudio="URI"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval">
child elements
</choice>
```

Attributes

dtmf

This attribute is the DTMF sequence for this choice.   This attribute is optional.

accept

The accept attribute overrides the setting for the accepted attribute in the <menu> element, with either exact or approximate values. This attribute is optional and defaults to exact.

* exact – use exact to define the exact phrase to be recognized.

* approximate – use approximate to define an approximate recognition phrase. A subset of the words in the phrase expression can be matched. For example, "Hello world" can be matched with "Hello world", "Hello", or "World".

next

The next attribute is the URI of the next dialog or document.

event

This attribute throws a specified event. Next and expr attributes have precedence over this attribute.

expr

The expr attribute is an ECMAScript Expression that defines the URI to transition. The next attribute has precedence over this attribute.

fetchaudio

The fetchaudio attribute is the URI of the audio to play while waiting for the next document to be fetched.

fetchtimeout

This attribute indicates the time interval to wait for an audio file to be fetched before playing the alternate content. This attribute is optional.

fetchhint

The fetchhint attribute defines when the audio file should be retrieved. This attribute is optional.

* prefetch - the file can be fetched when the document is loaded.

* safe - the file is fetched only when needed.

maxage

This attribute indicates the maximum time in seconds that this document will use this file before fetching another copy. This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use the file that exceeded the maxage time. This attribute is optional.

Parents

<menu>

Children

<audio>, <enumerate>, <grammar>, <value>

Extensions

None.

Limitations/Restrictions

The dtmf attribute must be specified.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <menu>
  <prompt> What do you want for drink, coffee or tea? </prompt>
  <choice dtmf="1" next="#getcoffee"> coffee </choice>
  <choice dtmf="2" next="#gettea"> tea </choice>
  <noinput> Please say coffee or tea </noinput>
  <nomatch> Please say coffee or tea </nomatch>
 </menu>
 <form id="getcoffee">
  <block>
```

```
   <prompt>Ok, here's your coffee</prompt>
  </block>
 </form>
 <form id="gettea">
  <block>
   <prompt>Ok, here's your tea</prompt>
  </block>
 </form>
</vxml>
```

<clear>

Description

The <clear> element resets one or more form items and will perform the following actions:

* Sets the form item variable to ECMAScript undefined.

* Reinitializes the prompt and event counters for the form item.

Syntax

```
<clear
namelist="item1 item2 item3 ..."/>
```

Attributes

namelist

The namelist attribute defines the names of the form items to be cleared. When not specified, all of the form items in the current form are cleared. This attribute is optional.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="password">
   <prompt> what is the code word </prompt>
   <help> It is the name of a fruit </help>
   <noinput> I did not hear you. </noinput>
   <nomatch count="1"> Noop. Try again </nomatch>
   <nomatch count="2"> Noop. give another try </nomatch>
   <nomatch count="3">
    Sorry. You didn't get it for three times. Bye
    <disconnect/>
   </nomatch>
   <filled>
    <if cond="password=='reset'">
     <clear namelist="password"/>
    <else/>
     This is correct.
    </if>
   </filled>
  </field>
 </form>
</vxml>
```

<disconnect>

Description

This element disconnects the user's phone call. As a result, the interpreter context will throw the telephone.disconnect.hangup event. A <disconnect> differs from an <exit> in that it forces the interpreter context to drop the call.

Syntax

```
<disconnect/>
```

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <block>
   <prompt>Hello and goodbye!</prompt>
   <disconnect />
  </block>
 </form>
</vxml>
```

<else>

Description

The <else> element is used within the conditional logic statement <if> and optional <elseif> element.

Syntax

```
<else/>
```

Parents

<if>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="password">
   <prompt> what is the code word </prompt>
   <help> It is the name of a fruit </help>
   <noinput> I did not hear you. </noinput>
   <nomatch count="1"> Nope. Try again </nomatch>
   <nomatch count="2"> Nope. Give it another try </nomatch>
   <nomatch count="3">
    Sorry. You didn't get it three times. Bye
    <disconnect/>
   </nomatch>
   <filled>
    <if cond="password=='reset'">
     <clear namelist="password"/>
    <else/>
     This is correct.
    </if>
   </filled>
  </field>
 </form>
</vxml>
```

<elseif>

Description

The <elseif> element is used within the conditional logic statement <if> and optional <else> element.

Syntax

```
<elseif
cond="ECMAScript_Expression"/>
```

Attributes

cond

The cond attribute is the Boolean expression for the conditional logic statement. This attribute is required.

Parents

<if>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <var name="card_type" expr="'amex'" />
 <form>
  <field name="card_num" type="digits">
   <prompt>What is your card number?</prompt>
   <filled>
    <if cond="card_type == 'amex' &amp;&amp; card_num.length != 15">
     American Express card numbers must have 15 digits.
     <clear namelist="card_num"/>
    <elseif cond="card_type != 'amex' &amp;&amp; card_num.length != 16"/>
     Mastercard and Visa card numbers have 16 digits.
     <clear namelist="card_num"/>
    </if>
   </filled>
  </field>
 </form>
</vxml>
```

<enumerate>

Description

The <enumerate> element is an automatically generated description of the choices available for the user. It specifies a template that is applied to each choice in the order they appear in the <menu> element, or in the <field> element that contains <option> elements.

Syntax

```
<enumerate>
child elements
</enumerate>
```

Anonymous Variables

If <enumerate> is used with no content, it lists all choices. You can customize the content using the two anonymous variables that are available within the scope of the <enumerate> element:

* _prompt – the prompt for the current choice.

* _dtmf – the assigned DTMF sequence for the current choice.

Parents

<audio>, <catch>, <error>, <field>, <filled>, <help>, <if>, <menu>, <noinput>, <nomatch>, <prompt>

Children

<audio>, <value>, Speech Markup (SSML)

Extensions

None.

Limitations/Restrictions

The dtmf attributes in the <choice> tags must be specified.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <menu dtmf="true">
  <prompt>
   Welcome home.
   <enumerate>
    For <value expr="_prompt"/>, press <value expr="_dtmf"/>.
   </enumerate>
  </prompt>
  <choice dtmf="1" next="http://www.example.com/vxml/sports.vxml">
   sports
  </choice>
  <choice dtmf="2" next="http://www.example.com/weather.vxml">
   weather
  </choice>
  <choice dtmf="3" next="http://www.example.com/news.vxml">
   news
  </choice>
 </menu>
</vxml>
```

<error>

Description

The <error> element catches an error event. This is a shorthand notation for <catch event="error"> that catches all events of type error.

Syntax

```
<error
count="Integer"
cond="ECMAScript_Expression">
child elements
</error>
```

Attributes

count

This attribute defines the event count, as in <catch> element. This attribute is optional.

cond

The cond attribute is a Boolean condition to test to see if the event is caught by the <error>. This attribute is optional.

Anonymous Variables

Two anonymous variables are available within the scope of the <catch> element:

* _event – the event contains the name of the event that was thrown.

* _message – the message contains the message string from the corresponding <throw> element, or a platform-defined value for events raised by the platform.

Parents

<field>, <form>, <initial>, <menu>, <object>, <record>, <subdialog>, <transfer>, <vxml>

Children

<assign>, <audio>, <clear>, <disconnect>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <error>
  <prompt>An error has occurred.</prompt>
  <exit/>
 </error>
 <form>
  <block>
   <audio expr="directory+'badaudio.wav'"/>
  </block>
 </form>
</vxml>
```

<exit>

Description

The <exit> element exits the interpreter session and returns control to the interpreter context, which determines what to do next. This element differs from the <return> element in that <exit> terminates all loaded elements while <return> returns from a <subdialog> invocation.

Syntax

```
<exit
expr="ECMAScript_Expression"
namelist="item1 item2 item3..."/>
```

Attributes

expr

The expr attribute is the expression to be evaluated and returned. This attribute is optional and the value is sent to the call control.

namelist

The namelist attribute indicates variable names to be returned to interpreter context, and returns nothing by default. This attribute is optional.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <error>
  <prompt>An error has occurred.</prompt>
  <exit expr="'ERROR'"/>
 </error>
 <form>
  <block>
   <audio expr="directory+'badaudio.wav'"/>
  </block>
 </form>
</vxml>
```

<field>

Description

The <field> element declares an input field in a form and prompts the user for values that match a grammar.

Syntax

```
<field
name="String"
expr="ECMAScript_Expression"
cond="ECMAScript_Expression"
type="boolean"|"number"|"digit"
slot="String"
modal="true" | "false"
saveutterance="true" | "false">
child elements
</field>
```

Attributes

name

The name attribute defines the name of the field item variable that holds the matched user input. This attribute is required.

expr

The expr attribute is the initial value of the field item variable. This field will be visited only if the expression evaluates to undefined. This attribute is optional and defaults to undefined.

cond

The cond attribute is a Boolean condition that must evaluate to true in order for this field to be visited. This attribute is optional and defaults to true.

type

The type attribute specifies a built-in grammar. This attribute is optional and can be used as an alternative to the <grammar> element.

slot

The slot attribute is the name of the grammar slot used to populate the field item variable for mixed initiative dialog. This attribute is optional and defaults to the variable name.

modal

The modal attribute is set to true if only the field's grammars are enabled. Otherwise, all active grammars are enabled. This attribute is optional and defaults to false.

Shadow Variables

The <field> shadow variable (name$) has the following properties after the field is filled in by user input. These properties are available in object application.lastresult$ as well:

* name$.utterance – a raw string of words that were recognized.

* name$.inputmode – a user input type, either dtmf or voice.

* name$.interpretation – an interpretation for this result.

* name$.confidence – a recognition confidence level. Floating point value between 0 to 1.0.

DTMF Built-in types

The following built-in types are supported for the DTMF input mode:

boolean

The grammar for affirmative and negative phrases. It returns "true" for yes and "false" for no.

digits

The grammar for a string of digits from spoken or DTMF input. It returns the string of digits.

number

The grammar for numbers. It returns a string of digits from 0 to 9, and may optionally include a decimal point (".") and/or a plus or minus sign.

DTMF Built-in type parameters

The following built-in type parameters can be parameterized with this syntax:

typename?parameter1=value1;parameter2=value2

For example, we can assign a DTMF sequence for a Boolean type:

<field name="mychoice" type="boolean?y=5;n=6">

boolean

- y - the DTMF sequence for an affirmative answer.
- n - the DTMF sequence for a negative answer.

digits

- length - exact number of digits.

number

- length - exact number of numbers.

If there is a conflict among these parameters, an error.badfetch event is thrown.

ASR Built-in types

The built-in types supported depends on the ASR installed and configured. Refer to the ASR provider.

Parents

<form>

Children

<audio>, <catch>, <enumerate>, <filled>, <grammar>, <help>, <link>, <noinput>, <nomatch>, <option>, <prompt>, <property>, <return>, <value>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="testfield">
  <block>
   Welcome to the restaurant.
  </block>
<!-- field using built in grammar with digits type -->
  <field name="built_in_gram1" type="number">
   <prompt> How many people you have? </prompt>
   <filled>
    Here is a table for <value expr="built_in_gram1"/> people.
    Have a seat.
    <goto nextitem="built_in_gram2" />
</filled>
  </field>
<!-- filed using built in grammar with boolean type -->
  <field name="built_in_gram2" type="boolean">
   <prompt> Are you ready for ordering now ? </prompt>
   <filled>
    <if cond="built_in_gram2 == 'true'">
     <goto nextitem="optionlist" />
    <elseif cond="built_in_gram2 == 'false'"/>
     <goto nextitem="notready" />
    </if>
   </filled>
  </field>
<!-- filed using option list -->
  <field name="optionlist">
   <prompt>
    Which entree would you like, baked potato, french fries?
   </prompt>
   <option dtmf="1" value="baked potato">baked potato</option>
   <option dtmf="2" value="french fries">french fries</option>
   <filled>
    ok. <value expr="optionlist"/> as entree.
    <goto nextitem="explicit_gram1" />
   </filled>
  </field>
<!-- filled using explicit grammar -->
  <field name="explicit_gram1">
   <prompt>
    Here is the main dishes you could choose from,
    Vegetarain delight, Prime Rib, Baked clams
   </prompt>
   <grammar type="text/x-grammar-choice-dtmf" mode="dtmf">
    1 {Vegetarain delight} |
    2 {Prime Rib} |
    3 {Baked clams}
   </grammar>
   <filled>
    Good choice.
    <value expr="explicit_gram1"/>
    is my favorite as well.
    <goto nextitem="ordertaken" />
   </filled>
  </field>
  <block name="goodbye">
   good bye
   <disconnect/>
  </block>
  <block name="ordertaken">
   Please wait. Your meal will be ready shortly.
   <disconnect/>
  </block>
  <block name="notready">
   Ok. Take your time and call back later.
   <disconnect/>
  </block>
 </form>
</vxml>
```

<filled>

Description

The <filled> element specifies an action to perform when some combinations of fields are filled by user input. It may occur as a child of the element, or as a child of a field item.

Syntax

```
<filled
mode="all" | "any"
namelist="item1 item2 item3...">
child elements
</filled>
```

Attributes

mode

This attribute cannot be defined when it is in a field item. There are 2 filled modes, all any. This attribute is optional and the default is all.

all – the action is executed when all of the fields in namelist attribute are filled, and at least one has been filled by the last user input.

any – the action is executed when any of the specified fields are filled by the last user input.

namelist

This attribute cannot be defined when it is in a field item. The namelist defines the fields to trigger. This attribute is optional and the default is list of all form's field items.

Parents

<field>, <form>, <object>, <record>, <subdialog>, <transfer>

Children

<assign>, <audio>, <clear>, <disconnect>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Code Example

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <var name="card_type" expr="'amex'" />
 <form>
  <field name="card_num" type="digits">
   <prompt>What is your card number?</prompt>
   <filled>
    <if cond="card_type == 'amex' &amp;&amp; card_num.length != 15">
     American Express card numbers must have 15 digits.
     <clear namelist="card_num"/>
    <elseif cond="card_type != 'amex' &amp;&amp; card_num.length != 16"/>
     Mastercard and Visa card numbers have 16 digits.
     <clear namelist="card_num"/>
    </if>
   </filled>
  </field>
 </form>
</vxml>
```

<form>

Description

Forms are key components in VoiceXML applications. Forms collect user input and present information to the user. A form can define a field item (for collecting user input), or a control item (contains procedural items to help collect inputs).

Syntax

```
<form
id="String"
scope="dialog" | "document"
cleardtmf="true" | "false">
child elements
</form>
```

Attributes

id

The id attribute defines the name of the form. If specified, it can be referenced within the same document or from another document. e.g. <form id="hello"> can be referenced as <goto next="#hello"/>. This attribute is optional.

scope

The scope attribute defines the scope of the form's grammar. This attribute is optional and defaults to dialog.

dialog – the form's grammars are only active within the form.

document – the form's grammars are active throughout the document. If this is an application root document, then the form grammars are active throughout the application.

Parents

<vxml>

Children

<block>, <catch>, <error>, <field>, <filled>, <grammar>, <help>, <initial>, <link>, <noinput>, <nomatch>, <object>, <property>, <record>, <script>, <subdialog>, <transfer>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="first">
  <block>
   <prompt>Hi, this is the first form</prompt>
   <goto next="#second"/>
  </block>
 </form>
 <form id="second">
  <block>
   <prompt>Thank you for coming to the second form</prompt>
  </block>
 </form>
</vxml>
```

<goto>

Description

The <goto> attribute transitions to another item in the current form, another dialog in the same document, or to a different document. Transitioning to another dialog in the current document or to a different document will cause the old dialog's variables to be lost. Document variables are retained when transitioning to the same document with an empty URI reference.

Syntax

```
<goto
next="URI"
expr="ECMAScript_Expression"
nextitem="String"
expritem="ECMAScript_Expression"
fetchaudio="URI"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval"/>
```

Attributes

next

The next attribute is the URI in which to transition to. This attribute is required, and one of next, expr, nextitem, expritem is also required.

expr

The expr attribute is an ECMAScript Expression that yields the URI. This attribute is required, and one of next, expr, nextitem, expritem is also required.

nextitem

The nextitem attribute is the name of the next form item to visit in the current form. This attribute is required, and one of next, expr, nextitem, expritem is also required.

expritem

The expritem attribute is the ECMAScript Expression that yields the name of the next form item to visit. This attribute is required, and one of next, expr, nextitem, expritem is also required.

fetchaudio

The fetchaudio attribute is the URI of audio to play while waiting for the next document to be fetched.

fetchtimeout

The fetchtimeout attribute is the time interval to wait for an audio file to be fetched before playing the alternate content.

This attribute is optional.

fetchhint

The fetchhint attribute defines when the audio file should be retrieved. This attribute is optional.

* prefetch – the audio file may be downloaded when the page is loaded.

* safe – only loads the audio file when needed.

maxage

The maxage attribute indicates the maximum time in seconds that this document will use this file before fetching another copy.

This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use the file that exceeded the maxage time.

This attribute is optional.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="maindish">
  <field name="done" type="boolean">
   <prompt> Are you done here ? </prompt>
   <filled>
    <if cond="done== 'FALSE'">
     <goto nextitem="notdone" />
    </if>
   </filled>
  </field>
  <field name="whatnext">
   <prompt>
    Would like any desserts or you want me to bring your bill?
   </prompt>
   <grammar type="text/x-grammar-choice-dtmf">
    1 {bill} |
    2 {desserts}
   </grammar>
   <filled>
    <if cond="whatnext=='bill'">
     <goto nextitem="bill" />
    <elseif cond="whatnext=='desserts'" />
     <goto next="#desserts" />
    </if>
   </filled>
  </field>
  <block name="bill">
   Please wait. I am getting your bill for you.
   <disconnect/>
  </block>
  <block name="notdone">
   Oh, I am sorry. Enjoy.
   <disconnect/>
  </block>
 </form>
<form id="desserts">
  <field name="choice">
   <prompt>
<form id="desserts">
  <field name="choice">
   <prompt>
    What would you like for your desserts?
   </prompt>
  </field>
</form>
</vxml>
```

<grammar>

Description

The <grammar> element specifies a grammar (word or phrase) for speech recognition. When the grammar is recognized, the application may perform an action (such as a transition to another field item), or assign values to a field item variable. Multiple values can also be assigned with mixed initiative forms.

Note

Inline grammars that use the XML grammar format will be passed into the Speech Recognition Engine as an

XML document. The interpreter first decodes the VoiceXML document and then re-encodes the inline grammar into an XML document to pass to the Engine.

Syntax

```
<grammar>
xml:lang="language"
src="URI"
scope="dialog" | "document"
type="MIME_Type"
mode="dtmf" | "voice"
root="String"
version="version_number"
weight="positive floating_point"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval">
inline grammar
</grammar>
```

Attributes

xml:lang

The xml:lang attribute is the language and locale identifier of the grammar. Please refer to the multi-language support tutorial for detailed information.   This attribute is optional and defaults to en-US. The value choices are en-US, en-UK, fr-FR...

src:

The src attribute is the URI of the grammar specification. The URI can have one of the following formats:

* External grammar file – the URI of the grammar file.

* Built-in grammars – the builtin:grammar/type or builtin:dtmf/type, refer to <field> element for built-in types.

scope

The scope attribute defines the scope of the grammar. This attribute can be defined only if this <grammar> element is the child of a <form> or <menu> element. This attribute is optional and defaults to dialog.

* dialog – the grammar is only active within the form.

* document – the grammar is active throughout the document. If this is an application root document and the grammar is active throughout the application.

type

The type attribute defines the MIME type of grammar format. The following types are supported:

- application/x-jsgf for internal speech recognition.
- text/x-grammar-choice for voice speech recognition.
- text/x-grammar-choice-dtmf for DTMF recognition.
- application/srgs+xml for SRGS/XML grammar format.
- application/lumenvox-abnf for Lumenvox/ABNF format.
- application/isolated for Verbio/Isolated grammar format.
- application/verbio-abnf for Verbio/ABNF grammar format.

mode

The mode attribute defines the mode of the grammar. This attribute is required.

* voice – the voice input.

* dtmf – the DTMF input. This replaces the obsolete <dtmf> element in VoiceXML 1.0.

root

The root attribute specifies the root rule of the grammar when this is an inline XML grammar. This attribute is optional and defaults to default rule.

version

The version attribute defines the version of the grammar. This attribute is optional and defaults to 1.0.

weight

The weight attribute defines the weight of a grammar, which indicates the possible occurrence of the grammar which can potentially increase recognition accuracy. Grammar weights only affect grammar processing; they do not affect the processing of grammar results. Different speech recognition engines and VoiceXML platforms also treat weights differently. This attribute does not apply to DTMF grammars and implicit grammars (i.e. <grammar> elements as children of <option> or <choice> tags). The range of values depends on the speech recognition engine, but it is usually 0.0 to 1.0. This attribute is optional and defaults to 1.0.

fetchtimeout

The fetchtimeout attribute is the time interval to wait for an grammar file to be fetched. This attribute is optional.

fetchhint

The fetchhint attribute defines when the grammar file should be retrieved. This attribute is optional.

* prefetch – the audio file may be downloaded when the page is loaded.

* safe – the audio file only loads when needed.

maxage

The maxage attribute indicates the maximum time in seconds that this document will use the grammar file before fetching another copy.   This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use a grammar file that exceeded the maxage time.   This attribute is optional.

Parents

<choice>, <field>, <form>, <link>, <record>, <transfer>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="translate">
  <field name="number">
   <prompt>
    Enter a number
   </prompt>
   <grammar type="text/x-grammar-choice-dtmf">
    1 {uno} |
    2 {dos} |
    3 {tres} |
    4 {cuatro} |
    5 {cinco}
   </grammar>
   <filled>
    <prompt>
     <value expr="lastresult$.utterance" />
     in Spanish is
     <value expr="number" />
    </prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<help>

Description

The <help> element catches a help event. This is a shorthand notation for <catch event="help">.

Syntax

```
<help
count="Integer"
cond="ECMAScript_Expression">
child elements
</help>
```

Attributes

count

The count attribute is the event count just as in the <catch> element. This attribute is optional.

cond

The cond attribute is a Boolean condition to test to see if the event is caught by the <help>. This attribute is optional.

Parents

<field>, <form>, <initial>, <menu>, <object>, <record>, <subdialog>, <transfer>, <vxml>

Children

<assign>, <audio>, <clear>, <disconnect>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <link dtmf="*" event="help" />
 <form>
  <field name="hello">
   <option dtmf="0">0</option>
   <help>Just press 0.</help>
   <prompt>Say hello</prompt>
   <noinput>Say something</noinput>
   <nomatch>It is not 0.</nomatch>
   <filled>
    <prompt>Hello, world!</prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<if>

Description

The <if> element defines if-then-else conditional logic.

Syntax

```
<if
cond="ECMAScript_Expression">
child elements
</if>
```

Attributes

cond

The cond attribute is a Boolean expression for the conditional logic statement. This attribute is required.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

<assign>, <audio>, <clear>, <disconnect>, <else>, <elseif>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <var name="card_type" expr="'amex'" />
 <form>
  <field name="card_num" type="digits">
   <prompt>What is your card number?</prompt>
   <filled>
    <if cond="card_type == 'amex' &amp;&amp; card_num.length != 15">
     American Express card numbers must have 15 digits.
     <clear namelist="card_num"/>
    <elseif cond="card_type != 'amex' &amp;&amp; card_num.length != 16"/>
     Mastercard and Visa card numbers have 16 digits.
     <clear namelist="card_num"/>
    </if>
   </filled>
  </field>
 </form>
</vxml>
```

<initial>

Description

The <initial> element declares initial logic upon entry into a mixed-initiative form.

Unlike <field>, <initial> has no grammars and no <filled> action. <initial> can request user input and handle events. <initial> continues to be visited while its form item variable is undefined and its condition is true. When any of the form's fields are filled in by user input, then all <initial> form item variables are set to true, before any <filled> actions are executed. Then the form will visit those fields that are still unfilled to complete the form.

Syntax

```
<initial
name="String"
expr="ECMAScript_Expression"
cond="ECMAScript_Expression">
child elements
</initial>
```

Attributes

name

The name attribute represents the name of form item variable used. Use this variable if you want to explicitly control the <initial> execution. This attribute is optional and defaults to an inaccessible internal variable.

expr

The expr attribute is the initial value of the form item variable. <initial> will be visited only if the expression evaluates to undefined. This attribute is optional and defaults to undefined.

cond

The cond attribute is a Boolean condition that must evaluate to true in order for the <initial> element to be visited. This attribute is optional and defaults to true.

Parents

<form>

Children

<audio>, <catch>, <error>, <help>, <link>, <noinput>, <nomatch>, <prompt>, <property>, <value>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="product">
  <grammar type="text/x-grammar-choice-dtmf">
   1 {uno} |
   2 {dos} |
   3 {tres} |
   4 {cuatro} |
   5 {cinco}
  </grammar>
  <block>
   Welcome to the Product Information By Phone.
  </block>
  <initial name="id_product">
   <prompt>Enter the product ID?</prompt>
   <nomatch count="1">
    Please say something like this,
    "1 2 2 3 4".
   </nomatch>
   <nomatch count="2">
    I'm sorry, I still don't understand.
    I'll ask you for information one piece at a time.
    <assign name="id_product" expr="true"/>
    <reprompt/>
   </nomatch>
  </initial>
  <field name="color_product">
   <prompt>From which city are you leaving?</prompt>
  </field>
  <field name="form_product">
   <prompt>Which city are you going to?</prompt>
  </field>
  <block>
    <prompt>
      You said that you wanted <value expr="color_product"/>
      <value expr="color_product"/>.
    </prompt>
  </block>
 </form>
</vxml>
```

<link>

Description

A <link> element may have one or more grammars. The grammars are in the scope of the element containing this <link> element. When the user input matches one of the linked grammars, it activates the <link> to either throw an event, or transition to another document or dialog.

Syntax

```
<link
next="URI"
expr="ECMAScript_Expression"
event="event"
dtmf="DTMF Sequence"
fetchaudio="URI"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval">
child_elements
</link>
```

Attributes

next

The next attribute is the URI to transition when a match is recognized. This attribute is required. One and only one of the next, expr, event or eventexpr attributes are required.

expr

The expr attribute is an expression that yields the URI to transition when a match is recognized. This attribute is required. One and only one of the next, expr, event or eventexpr attributes are required.

event

The event attribute is the event to throw when the user input matches one of the linked grammars. This attribute is required. One and only one of the next, expr, event or eventexpr attributes are required.

eventexpr

An ECMAScript expression evaluating to the name of the event to throw when the user matches one of the link grammars.One and only one of the next, expr, event or eventexpr attributes are required.

dtmf

The dtmf attribute indicates the DTMF sequence for this link. It is equivalent to a simple DTMF <grammar>. This attribute can be used concurrently with other grammars, as the link is activated when user input matches a linked grammar or the DTMF sequence. This attribute is optional.

fetchaudio

The fetchaudio attribute is the URI of audio to play while waiting for the next document to be fetched. This attribute is optional.

fetchtimeout

The fetchtimeout attribute is the time interval to wait for an audio file to be fetched before playing the alternate content. This attribute is optional.

fetchhint

The fetchhint attribute defines when the audio file should be retrieved. This attribute is optional.

* prefetch – the audio file may be downloaded when the page is loaded.

* safe – the audio file loads only when needed.

maxage

The maxage attribute indicates the maximum time in seconds that this document will use this file before fetching another copy. This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use the file that exceeded the maxage time. This attribute is optional.

Parents

<field>, <form>, <initial>, <vxml>

Children

<grammar>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <link event="goodbye" dtmf="1"/>
 <catch event="goodbye">
  <prompt>Thank you for trying this script, goodbye.</prompt>
  <exit/>
 </catch>
 <form>
  <field name="hello" type="digits">
   <prompt>Say hello</prompt>
   <noinput>Say something</noinput>
   <filled>
    <prompt>Hello, world!</prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<log>

Description

The <log> element allows the application to generate debug messages. This element can contain a combination of text and <value> elements in which the resulting message will be the concatenation of the text and string value of the <value> element. The messages will be logged to the metrics file for viewing.

Syntax

```
<log
expr="ECMAScript_Expression"
label="String">
text
</log>
```

Attributes

expr

The expr attribute is a string expression that will be appended to the <log> element content and label. This attribute is optional.

label

The label attribute is a string label that will be appended to the <log> element content before expr. This attribute is optional.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

<value>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="hello">
   <grammar type="text/x-grammar-choice-dtmf">
    1 {hello} | 2 {world}
   </grammar>
   <prompt>Say hello</prompt>
   <noinput>Say something</noinput>
   <filled>
    <log>The user said <value expr="hello"/>.</log>
    <prompt>Hello, world!</prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<menu>

Description

The <menu> element presents a menu of choices to the user and transitions to another dialog based on user input.

Syntax

```
<menu
id="String"
```

```
scope="dialog" | "document"
dtmf="true" | "false"
accept="exact" | "approximate">
child elements
</menu>
```

Attributes

id

The id attribute defines the name of the form. If specified, it can be referenced within the same document or from another document. For example, <form id="hello"> can be referenced as <goto next="#hello"/>. This attribute is optional.

scope

The scope attribute defines the scope of the form's grammar. This attribute is optional and defaults to dialog.

* dialog – the form's grammars are only active within the form.

* document – the form's grammars are active throughout the document. If this is an application root document, then the form's grammars are active throughout the application.

dtmf

The dtmf attribute enables DTMF input for all menu choices. This attribute is optional and defaults to false.

* true – the interpreter assigns a DTMF sequence "1", "2", "3"... to the choices in order for all menu choices that do not explicitly define a DTMF sequence.

* false – the interpreter does not implicitly assign DTMF sequences.

accept

The accept attribute specifies the default grammar for each <choice> element. This attribute is optional and defaults to exact.

* exact – the text of the <choice> element that defines the exact phrase to be recognized.

* approximate – the text of the <choice> element that defines an approximate recognition.

phrase

The phrase attribute is a subset of the words in the phrase expression that can be matched. For example, "Hello world" can be matched with "Hello world", "Hello", or "World".

Parents

<vxml>

Children

<audio>, <catch>, <enumerate>, <error>, <help>, <noinput>, <nomatch>, <prompt>, <property>, <record>, <value>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
```

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <menu dtmf="true">
  <prompt>
   What information do you want?
   <enumerate>
    For <value expr="_prompt"/>, press <value expr="_dtmf"/>
   </enumerate>
  </prompt>
  <choice dtmf="1" next="#exit"> goodbye </choice>
  <choice dtmf="2" next="sports.vxml"> sports </choice>
  <choice dtmf="3" next="weather.vxml"> weather </choice>
  <choice dtmf="4" next="stock.vxml"> stock </choice>
 </menu>
 <form id="exit">
  <block>
   Good bye.
   <disconnect/>
  </block>
 </form>
</vxml>
```

<meta>

Description

The <meta> element specifies general information about the VoiceXML application.

Syntax

```
<meta
name="String"
content="String"
http-equiv="String"/>
```

Attributes

name

The name attribute is the name of the meta-data property. This attribute is required (specifically either name or http-equiv).

content

The content attribute is the value of the meta-data property. This attribute is required.

http-equiv

The http-equiv attribute is the name of an HTTP response header. This attribute is required (specifically either name or http-equiv).

application

The application attribute defines name of the application for billing purposes.

Meta-data Properties

application

The application meta-data property defines the name of the application for billing purposes.

callrequest

The call request meta-data property defines if the application will accept or decline incoming calls. If the value is decline, then the application will decline incoming calls. Otherwise, the application accepts incoming calls.

maintainer

The maintainer meta-data property is the email address that log files will be sent to.

Parents

<vxml>

Children

None.

Extensions

None.

Limitations/Restrictions

Not supported.

Example Code

Meta information:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <meta name="maintainer" content="jpdoe@anycompany.example.com"/>
 <form>
  <block>
   <prompt>Hello</prompt>
  </block>
 </form>
</vxml>
```

Meta specifies an HTTP response header:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <meta http-equiv="Expires" content="0"/>
 <meta http-equiv="Date" content="Thu, 12 Dec 2000 23:27:21 GMT"/>
 <form>
  <block>
   <prompt>Hello</prompt>
  </block>
 </form>
</vxml>
```

<noinput>

Description

The <noinput> element handles the event when the user does not give inputs in a field. This is a shorthand notation for <catch event="noinput"> that catches all events of type error.

Syntax

```
<noinput
```

```
count="Integer"
cond="ECMAScript_Expression">
child elements
</noinput>
```

Attributes

count

The count attribute indicates the event count as in the <catch> element.   This attribute is optional.

cond

The cond attribute is a Boolean condition to test to see if the event is caught by the <noinput> element. This attribute is optional.

Parents

<field>, <form>, <initial>, <menu>, <object>, <record>, <subdialog>, <transfer>, <vxml>

Children

<assign>, <audio>, <clear>, <disconnect>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="hello">
   <grammar type="text/x-grammar-choice-dtmf">
    1 {hello}
   </grammar>
   <prompt>Say hello</prompt>
   <noinput>Say something</noinput>
   <filled>
    <prompt>Hello, world!</prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<nomatch>

Description

The <nomatch> element handles the event when the user gives unrecognized input. This is a shorthand notation for <catch event="nomatch"> that catches all events of type error.

Syntax

```
<nomatch
count="Integer"
cond="ECMAScript_Expression">
child elements
</nomatch>
```

Attributes

count

The count attribute is the event count as in the <catch> element. This attribute is optional.

cond

The cond attribute is a Boolean condition to test to see if the event is caught by the <noinput> element. This attribute is optional.

Parents

<field>, <form>, <initial>, <menu>, <object>, <record>, <subdialog>, <transfer>, <vxml>

Children

<assign>, <audio>, <clear>, <disconnect>, <enumerate>, <exit>, <goto>, <if>, <prompt>, <reprompt>, <return>, <script>, <submit>, <throw>, <value>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="hello">
   <grammar type="text/x-grammar-choice-dtmf">
    1 {hello}
   </grammar>
   <prompt>Say hello</prompt>
   <nomatch>That's not hello</nomatch>
   <filled>
    <prompt>Hello, world!</prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<object>

Description

The <object> element interacts with custom extensions. A VoiceXML platform may have platform-specific functionality that an application wants to use, such as speaker verification, native components, additional telephony, etc. Such platform-specific objects are accessed using the <object> element, which is analogous to the HTML <OBJECT> element.

Syntax

```
<object
name="String"
expr="ECMAScript_Expression"
cond="ECMAScript_Expression"
classid="URI"
codebase="URI"
codetype=""
data="URI"
type=""
archive="URI"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval">
child elements
</object>
```

Attributes

name

The name attribute defines the name of the variable when the object is evaluated. This attribute is required.

expr

The expr attribute is the initial value of the variable. This attribute is optional and defaults to undefined.

cond

The cond attribute is a Boolean expression that must evaluate to true in order for this object to execute. This attribute is optional and defaults to true.

classid

The classid attribute is the URI specifying the location of the object's implementation. The platform can access external objects (dynamic installed libraries). To specify one, give the URI like, "extern:name", where 'name' generates the reference to VXIobject_name.so. I6NET can provide an SDK tool kit to develop external objects.

codebase

The codebase attribute is the base path used to resolve relative URIs specified by classid, data, and archive. This attribute is optional and defaults to the base URI of current document.

codetype

The codetype attribute is the content type of data expected when downloading the object specified by classid. This attribute is optional and defaults to the value of the type attribute.

data

The data attribute is the URI specifying the location of the object's data. If it is a relative URI, it is interpreted as relative to the codebase attribute.

type

The type attribute is the content type of the data specified by the data attribute.

archive

The archive attribute is a space-separated list of URIs for archives containing resources relevant to the object, which may include the resources specified by the classid and data attributes. URIs which are relative are interpreted as relative to the codebase attribute.

fetchtimeout

The fetchtimeout attribute is the time interval to wait for an audio file to be fetched before playing the alternate content.   This attribute is optional.

fetchhint

The fetchhint attribute defines when the audio file should be retrieved. This attribute is optional.

* prefetch – the audio file may be downloaded when the page is loaded.

* safe – the audio file is loaded only when needed.

maxage

The maxage attribute indicates the maximum time in seconds that this document will use this object before fetching another copy.   This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use an object that exceeded the maxage time. This attribute is optional.

Parents

<vxml>

Children

<audio>, <catch>, <error>, <filled>, <help>, <noinput>, <nomatch>, <param>, <prompt>, <property>, <value>

Extensions

Object , “offset” – can get the offset of the last audio prompted with the property control set to “yes.”

Object , “save” – can save a record locally.

Object , “pick” – can get a local file and use it to get and post local contents.

Object , “delete” - can delete a local file.

Limitations/Restrictions

The elements: codebase, codetype, data, type, and archive are not used.

No platform-specific objects are currently defined.

Example Code

```
<?xml version = "1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <object name="message" classid="extern:shell">
   <param name="command" expr="'ls -al'" />
  </object>
  <block>
   Parameter = ls -al.
   Result = <value expr="message.result" />.
  </block>
 </form>
</vxml>
```

<option>

Description

The <option> element specifies an option in a <field>. It is a convenient way to specify a list of choices in a field without specifying a grammar. The grammar is generated automatically from the text contained in each <option>.

Syntax

```
<option
dtmf="DTMF sequence"
value="String">
text
</option>
```

Attributes

dtmf

The dtmf attribute is the DTMF sequence for this option. This attribute is optional, yet is required if the menu has the dtmf attribute that is true.

value

The value attribute is the string to assign to the field item variable. This option is selected, whether by speech or DTMF. This attribute is optional and defaults to the dtmf attribute if specified. Otherwise, it defaults to the option text with leading and trailing white space removed.

Parents

<field>

Children

None.

Extensions

None.

Limitations/Restrictions

The attribute dtmf is required, if the menu has the dtmf attribute that is true.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="maincourse">
   <prompt>
     Please select an entree. Today, we're featuring:
    <enumerate/>
   </prompt>
   <option dtmf="1" value="fish"> swordfish </option>
   <option dtmf="2" value="beef"> roast beef </option>
   <option dtmf="3" value="frog"> frog legs </option>
   <filled>
    <prompt>
     <value expr="maincourse"/>, good choice.
     Please enjoy your meal.
    </prompt>
    <submit next="maincourse.cgi"
```

```
     method="post" namelist="maincourse"/>
   </filled>
  </field>
 </form>
</vxml>
```

<param>

Description

The <param> element specifies values that are passed into the <object> or in <subdialog>. When <param> is contained in a <subdialog> element, the values specified are used to initialize <var> declarations in the subdialog that is invoked. The initialization takes precedence over the expr attribute in <var>.

Syntax

```
<param
name="String"
value="String"
expr="ECMAScript_Expression"
valuetype="data" | "ref"
type="MIME_type"/>
```

Attributes

name

The name attribute is the name associated with the parameter.   This attribute is required.

value

The value attribute is the string value of the parameter. This attribute is required and either value or expr is required.

expr

The expr attribute is an expression that yields the parameter value. This attribute is required and either value or expr is required.

valuetype

The valuetype attribute indicates the reference type of the value. This attribute is not used for <subdialog>. This attribute is optional and defaults to data.

* data – the value is actual data.

* ref – the value referenced by a URI.

type

The type attribute is the media type of the result provided by a URI if the valuetype is ref. This is only relevant for the uses of <param> in the <object> element. This attribute is optional.

Parents

<object>, <subdialog>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

This represents a document that calls the subdialog:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <subdialog name="result" src="subdialog.vxml">
   <param name="birthday" expr="'2000-02-10'"/>
   <param name="age" value="100"/>
   <filled>
    <submit next="process.vxml"/>
   </filled>
  </subdialog>
 </form>
</vxml>
```

This represents a document containing the subdialog:

```
<?xml version="1.0"?>
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="getdriverslicense">
  <var name="birthday"/>
  <var name="age"/>
  <block>
   <prompt>
    Hello, your birthday is <value expr="birthday"/>
    and you are <value expr="age"/> years old.
   </prompt>
   <return/>
   </block>
 </form>
</vxml>
```

<prompt>

Description

The <prompt> element controls the output of synthesized speech and prerecorded audio.

Note

Any prompts that contain Speech Markup (SSML) will be passed into the TTS Engine as an XML document. The interpreter first decodes the VoiceXML document, and then re-encodes the prompt content into an XML document to pass to the TTS Engine.

Syntax

```
<prompt
bargein="true" | "false"
bargeintype="energy" | "speech" | "recognition"
```

```
cond="ECMAScript_Expression"
count="Integer"
timeout="time_interval">
child elements
</prompt>
```

Attributes

bargein

The bargein attribute enables user input during the prompt. This attribute is optional.

* true – the user input can barge-in the current prompt. Subsequent prompts in the queue will be treated as bargein=true.

* false – the user can't barge-in to the prompt until the prompt has finished playing.

bargeintype

The bargeintype attribute specifies the barge-in type:

* speech – any user utterance can barge-in during the prompt.

* hotword – only when user input matches a grammar can barge-in occur during the prompt.

cond

The cond attribute is a Boolean expression that must evaluate to true for this prompt to be played. This attribute is optional and defaults to true.

count

The field item maintains a counter of the number of times the item has been visited. The count attribute allows the application to play different prompts based on the counter. The prompt will be played when the counter reaches the count attribute. This attribute is optional and defaults to 1.

timeout

The timeout attribute is the time to wait before throwing a noinput event. This attribute is optional.

xml:lang

The xml:lang attribute specifies the language and locale information of the VoiceXML document. If omitted, it will inherit this value from the document hierarchy, or ultimately from the platform default, which equates to 'en-US'. If text-to-speech is configured, the VoiceXML browser will make an HTTP request to get a wav or gsm file generated. The audio files generated could be cached to greatly reduce the text-to-speech access.

Parents

<block>, <catch>, <choice>, <error>, <field>, <filled>, <if>, <initial>, <menu>, <noinput>, <nomatch>, object>, <prompt>, <record>, <subdialog>

Children

<audio>, <enumerate>, <value>, Speech Markup (SSML)

Extensions

If xml:lang is equal to "video", then the VoiceXML interpreter will use text-to-video to generate a static GIF file to the video silence. See Text-to-Video Option.

Limitations/Restrictions

The attribute timeout has no effect.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="maincourse">
   <noinput> <reprompt/> </noinput>
   <nomatch> <reprompt/> </nomatch>
   <prompt count="1">Please select an entree.</prompt>
   <prompt count="2">Today, we're featuring:<enumerate/></prompt>
   <option dtmf="1" value="fish"> swordfish </option>
   <option dtmf="2" value="beef"> roast beef </option>
   <option dtmf="3" value="frog"> frog legs </option>
   <filled>
    <prompt>
     <value expr="maincourse"/>, good choice.
     Please enjoy your meal.
    </prompt>
    <submit next="maincourse.cgi"
     method="post" namelist="maincourse"/>
   </filled>
  </field>
 </form>
</vxml>
```

<property>

Description

The <property> element controls implementation platform settings. Properties are used to set values that affect platform behavior, such as the recognition process, timeouts, caching policy, etc. Please refer to Properties for list of properties.

Syntax

```
<property
name="String"
value="String"/>
```

Attributes

name

The name attribute indicates the property name. This attribute is required.

value

The value attribute is the property value. This attribute is required.

Main properties list

bargein

bargeintype

inputmodes

interdigittimeout

termchar

termtimeout

timeout

control

Parents

<field>, <form>, <initial>, <menu>, <object>, <record>, <subdialog>, <transfer>, <vxml>

Children

None.

Extensions

None.

Limitations/Restrictions

The properties listed below are not supported for speech recognition:

confidencelevel

sensitivity

speedcaccuracy

completetimeout

incompletetimeout

maxspeechtimeout

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <property name="bargein" value="true"/>
 <form>
  <field name="hello">
   <grammar type="text/x-grammar-choice-dtmf">
    1 {hello}
   </grammar>
   <prompt>This prompt can be barged in. Please say hello</prompt>
   <noinput>Say something</noinput>
   <filled>
    <prompt>Hello, world!</prompt>
    <goto nextitem="helloagain"/>
   </filled>
  </field>
  <field name="helloagain">
   <property name="bargein" value="false"/>
   <grammar type="text/x-grammar-choice-dtmf">
    1 {hello}
   </grammar>
   <prompt>This prompt cannot be barged in.
   Please say hello</prompt>
   <noinput>Say something</noinput>
   <filled>
    <prompt>Hello, world!</prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<record>

Description

The <record> element collects a recording from the user. A reference to the recorded audio is stored in the field item variable which can be played back or submitted to a server.

Syntax

```
<record
name="String"
expr="ECMAScript_Expression"
cond="ECMAScript_Expression"
modal="true" | "false"
beep="true" | "false"
maxtime="time_interval"
finalsilence="time_interval"
dtmfterm="true" | "false"
type="MIME_type">
child elements
</record>
```

Attributes

name

The name attribute defines the field item variable that holds the recording. This attribute is required.

expr

The expr attribute indicates the initial value of the field item variable. The <record> element will not be executed if the value is not undefined. This attribute is optional and defaults to undefined.

cond

The cond attribute is a Boolean expression that must evaluate to true in order for <record> to execute. This attribute is optional and defaults to true.

modal

If the modal attribute is true, all higher-level speech and DTMF grammars are turned off while making the recording. If this attribute is false, speech and DTMF grammars scoped to the form, document, application, and calling documents are listened to.   This attribute is optional and defaults to true.

beep

The beep attribute, if set to true, is the tone emitted just prior to recording. This attribute is optional and defaults to false.

maxtime

The maxtime is the maximum duration to record. Minimum value is 100ms. This attribute is optional.

dtmfterm

The dtmfterm attribute, if set to true, will enable a DTMF key press to terminate a recording. The DTMF tone is not part of the recording. This attribute is optional and defaults to true.

type

The type attribute is the media format of the resulting recording. This attribute is optional and defaults to “audio/x-wav”. Supported values are : “audio/x-gsm”, “audio/x-wav”, “video/mp4” (Iso Files, not mpeg3 files) and “video/3gpp”.

Shadow Variables

The following shadow variables are available in the same scope of the field item variable called name$:

* name$.duration – the duration of the recording in milliseconds.

* name$.size – the size of the recording in bytes.

* name$.termchar – if the dtmfterm attribute is true, and the user terminates the recording by pressing a DTMF key, then this shadow variable is the key pressed (e.g. "#"). Otherwise it is null.

* name$.maxtime – true if the recording was terminated because the maxtime duration was reached.

Parents

<form>

Children

<audio>, <catch>, <enumerate>, <error>, <filled>, <help>, <nomatch>, <noinput>, <grammar>, <prompt>, <property>, <value>

Extensions

I6NET can provide video tools to manage the recorded files (ex : generate thumbnail, convert the video files to FLV/Flash, or make audio extractions).

Limitations/Restrictions

The attribute mintime is not supported.

The following shadow variables are not set: name$.size, name$.maxtime.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form>
 <record name="awaymessage" beep="true" maxtime="10s"
  finalsilence="4s" dtmfterm="true">
  <prompt>
   At the tone, please record your message
  </prompt>
 </record>
 <field name="confirm" type="boolean">
  <prompt>
   The message is <value expr="awaymessage"/>
  </prompt>
  <prompt>
   To keep it, please say yes. To discard it, say no
  </prompt>
  <filled>
   <if cond="comfirm">
    <submit next="save.php" method="post"
     namelist="awaymessage" />
   </if>
  </filled>
 </field>
</form>
</vxml>
```

<reprompt>

Description

The <reprompt> element replays a previously played prompt. Normally, the interpreter stops playing prompts on the next form item after executing a <catch> element. However, if a <reprompt> is executed in the catch, the interpreter will process a normal prompt for the next form item. This includes a selection of a prompt and an incremental addition to the prompt counter.

Syntax

```
<reprompt/>
```

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
 <field name="hello">
  <grammar type="text/x-grammar-choice-dtmf">
   1 {hello}
  </grammar>
  <prompt>Say hello</prompt>
  <noinput><reprompt/></noinput>
  <nomatch><reprompt/></nomatch>
  <filled>
   <prompt>Hello, world!</prompt>
  </filled>
 </field>
 </form>
</vxml>
```

<return>

Description

The <return> element completes the execution of <subdialog> and returns control and data to a calling dialog.

Syntax

```
<return
event="event"
```

```
namelist="variable1 variable2 ..."/>
```

Attributes

event

Return and throw this event. Optional

namelist

Space-separated list of variables to be returned to the calling dialog. Optional (Defaults to no variables)

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

The following shows the document that calls the subdialog:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form>
 <subdialog name="result" src="subdialog.vxml">
  <filled>
   <prompt>Your account number is <value expr="result.acctnum"/>
    Your phone number is <value expr="result.acctphone"/>
   </prompt>
  </filled>
 </subdialog>
</form>
</vxml>
```

The following shows the document containing the subdialog:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form id="basic">
 <field name="acctnum" type="digits">
  <prompt>
    What is your account number?
  </prompt>
 </field>
 <field name="acctphone" type="phone">
  <prompt>
    What is your home telephone number?
  </prompt>
  <filled>
   <return namelist="acctnum acctphone"/>
  </filled>
 </field>
</form>
</vxml>
```

<script>

Description

The <script> element includes a block of client-side script. Each <script> element is executed in the scope of its containing element; i.e., it does not have its own scope. Variables defined in the <script> element are equivalent to variables defined using <var> within the same scope.

Syntax

```
<script
src="URI"
charset="Encoding"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval">
Script Text
</script>
```

Attributes

src

The src attribute is the URI specifying the location of the external script. This attribute is optional.

charset

The charset attribute is character encoding if external script is used. This attribute is optional.

fetchtimeout

The fetchtimeout attribute is the time interval to wait for audio file to be fetched before playing the alternate content. This attribute is optional.

fetchhint

The fetchhint defines when the audio file should be retrieved. This attribute is optional.

* prefetch – the audio file may be downloaded when the page is loaded.

* safe – the audio file loads only when needed.

maxage

The maxage attribute indicates the maximum time in seconds that this document will use this script file before fetching another copy. This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use a script file that has exceeded the maxage time. This attribute is optional.

Parents

<block>, <catch>, <error>, <filled>, <form>, <help>, <if>, <menu>, <noinput>, <nomatch>, <vxml>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <script>
 <![CDATA[
   function alwaysTrue() {
   // no need to escape the following less than sign
   return 1 < 2;
   }
 ]]>
 </script>
 <block>
  <prompt>
   Hello
  </prompt>
  <prompt cond="alwaysTrue">
   I am always here
  </prompt>
 </block>
</vxml>
```

NOTE: ü It is wise to put CDATA escapes around your scripts so you don't have to escape XML reserved characters (eg. <, >, &, etc).

<subdialog>

Description

The <subdialog> element invokes another dialog as a subdialog of the current one. The subdialog is a reusable dialog that allows values to be returned. The subdialog executes in a new execution context with all variables and execution states initialized. Values can be passed into the subdialog using <param> child elements; the subdialog must contain the <var> variable declaration for each parameter. The original dialog can continue execution only when the subdialog executes the <return> element. Returned values are available as properties of the <subdialog> field item variable.

Syntax

```
<subdialog
name="String"
expr="ECMAScript_Expression"
cond="ECMAScript_Expression"
namelist="variable1 variable2 ..."
src="URI"
method="get" | "post"
enctype="application/x-www-form-urlencoded" | "multipart/form-data"
```

```
fetchaudio="URI"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval">
Script Text
</script>
```

Attributes

name

The name attribute defines the name of the field item variable for the <subdialog> element. The returned results can be retrieved as a property of the variable, name.returnVariable. This attribute is required.

expr

The expr attribute is the initial value of the form item variable. The <subdialog> element will be visited only if the variable is undefined. This attribute is optional and defaults to undefined.

cond

The cond attribute is a Boolean expression that must evaluate to true for the <subdialog> to execute. This attribute is optional and defaults to true.

namelist

The namelist attribute is a space-separated list of variables to submit. This attribute is optional and defaults to nothing.

src

The src attribute is the URI of the <subdialog>. This attribute is required - specifically either src or srcexpr is required.

method

The method attribute specifies the query request method, get or post. This attribute is optional and defaults to get.

enctype

The enctype attribute defines the MIME encoding of the document. This attribute is optional and defaults to application/x-www-form-urlencoded. The following types are supported:

* application/x-www-form-urlencoded.

* multipart/form-data (to post recorded messages).

fetchaudio

The fetchaudio attribute is the URI of the audio to play while waiting for the next document to be fetched.

fetchtimeout

The fetchtimeout attribute is the time interval to wait for the audio file to be fetched before playing the alternate content. This attribute is optional.

fetchhint

The fetchhint attribute defines when the audio file should be retrieved. This attribute is optional.

* prefetch – the audio file may be downloaded when the page is loaded.

* safe – the audio file loads only when needed.

maxage

The maxage attribute indicates the maximum time in seconds that this document will use this subdialog file before fetching another copy. This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use a subdialog file that exceeded the maxage time. This attribute is optional.

Parents

<form>

Children

<audio>, <catch>, <enumerate>, <error>, <filled>, <help>, <noinput>, <nomatch>, <param>, <prompt>, <property>, <value>

Extensions

None.

Limitations/Restrictions

The variables submitted to the <subdialog> element in the namelist attribute may not be objects.

Example Code

The following is an example of a document that calls the <subdialog>:

```
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <subdialog name="result" src="subdialog.vxml">
   <filled>
    <prompt>
     Your account number is <value expr="result.acctnum"/>
     Your phone number is <value expr="result.acctphone"/>
    </prompt>
   </filled>
  </subdialog>
 </form>
</vxml>
```

The following is an example of a document containing the <subdialog>:

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="basic">
  <field name="acctnum" type="digits">
   <prompt> What is your account number? </prompt>
  </field>
  <field name="acctphone" type="phone">
   <prompt> What is your home telephone number? </prompt>
   <filled>   <prompt> What is your home telephone number? </prompt>
   <filled>
    <return namelist="acctnum acctphone"/>
   </filled>
  </field>
 </form>
</vxml>
```

<submit>

Description

The <submit> element submits values to document server. It is similar to <goto> in that it results in a new document being obtained. Unlike <goto>, it lets you submit a list of variables to the document server via HTTP GET or HTTP POST.

Syntax

```
<submit
next="URI"
expr="ECMAScript_Expression"
namelist="variable1 variable2 ..."
method="get" | "post"
enctype="MIME_type"
fetchaudio="URI"
fetchhint="prefetch" | "safe"
fetchtimeout="time_interval"
maxage="time_interval"
maxstale="time_interval">
Script Text
</script>
```

Attributes

next

The next attribute is the URI to which the query is submitted. This attribute is required - specifically either next or expr is required.

expr

The expr attribute is an expression that yields the URI. This attribute is required - specifically either next or expr is required.

namelist

The namelist attribute is a space-separated list of variables to submit. This attribute is optional and defaults to nothing.

method

The method attribute specifies the query request method, get or post. This attribute is optional and defaults to get.

enctype

The enctype attribute is MIME encoding of the document. This attribute is optional and defaults to application/x-www-form-urlencoded. The following types are supported:

* application/x-www-form-urlencoded.

* multipart/form-data.

fetchaudio

The fetchaudio attribute is the URI of the audio to play while waiting for the next document to be fetched.

fetchhint

The fetchhint defines when the audio file should be retrieved. This attribute is optional.

* prefetch – the audio file may be downloaded when the page is loaded.

* safe – the audio file is loaded only when needed.

maxage

The maxage attribute indicates the maximum time in seconds that this document will use this file before fetching another copy. This attribute is optional.

maxstale

The maxstale attribute indicates the maximum time in seconds that this document will use a file that exceeded the maxage time. This attribute is optional.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

The attribute fetchhint = prefetch is not supported, and is optional in VoiceXML.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="get_from_and_to_cities">
  <grammar src="from_to.grammar"/>
  <block>
   Welcome to the Driving Directions By Phone.
  </block>
  <initial name="bypass_init">
   <prompt>Where do you want to drive from and to?</prompt>
   <nomatch count="1">
    Please say something like this,
    "from Toronto Ontario to Ottawa Ontario".
   </nomatch>
   <nomatch count="2">
    I'm sorry, I still don't understand.
    I'll ask you for information one piece at a time.
    <assign name="bypass_init" expr="true"/>
    <reprompt/>
   </nomatch>
  </initial>
  <field name="from_city">
   <grammar src="city.grammar"/>
   <prompt>From which city are you leaving?</prompt>
   ...
  </field>
  <field name="to_city">
   <grammar src="city.grammar"/>
   <prompt>Which city are you going to?</prompt>
   ...
  </field>
  <block>
   <prompt>Retrieving directions</prompt>
   <submit next="direction.cgi" method="post"
   namelist="from_city to_city"/>
  </block>
 </form>
</vxml>
```

<throw>

Description

The <throw> element throws a pre-defined event or application-specific event.

Syntax

```
<throw
event="event"
eventexpr="ECMAScript_Expression"
message="String"
messageexpr="ECMAScript_Expression"/>
```

Attributes

event

The event attribute defines the event name to throw. The attribute is required – specifically the event or eventexpr is required.

eventexpr

The eventexpr is an expression that yields the event name. The attribute is required – specifically the event or eventexpr is required.

message

The message attribute indicates a message string providing additional context about the event being thrown. For the pre-defined events thrown by the platform, the value of the message is platform dependent. The message will be available as a variable within the scope of the catch element.   This attribute is optional.

messageexpr

The expression attribute defines the expression that yields the message. This attribute is optional.

Parents

<block>, <catch>, <error>, <filled>, <help>, <if>, <noinput>, <nomatch>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <catch event="goodbye">
  <prompt>goodbye.</prompt>
  <exit/>
 </catch>
 <form>
```

```
   <field name="hello">
    <grammar>hello | goodbye</grammar>
    <help>Just say hello</prompt>
    <prompt>Say hello</prompt>
    <noinput>Say something</noinput>
    <filled>
    <if cond="hello == 'goodbye'>
     <throw event="goodbye"/>
    </if>
    <prompt>Hello, world!</prompt>
    </filled>
   </field>
  </form>
</vxml>
```

<transfer>

Description

The <transfer> attribute transfers the caller to another phone number. There are 2 types of transfer:

* bridge transfer – the caller resumes the interpreter session after a call with the third party.

* blind transfer – as soon as the call with the third party connects, the platform throws a telephone.disconnect.transfer and terminates the interpreter session.

Syntax

```
<transfer
name="String"
expr="ECMAScript_Expression"
cond="ECMAScript_Expression"
dest="URI"
destexpr="ECMAScript_Expression"
bridge="true" | "false"
connecttimeout="time_interval"
maxtime="time_interval"
transferaudio="URI"
analysis="true" | "false"
connectwhen="analysis" | "answered" | "immediate"
uuidata="">
child elements
</transfer>
```

Attributes

Name

The name attribute is the name of the field item variable. It stores the result of the <transfer> and is required. The returned values are:

* busy – the endpoint refused the call.

* noanswer – there was no answer within the time specified by the connecttimeout attribute.

* network_busy – some intermediate network refused the call.

* near_end_disconnect – the call was completed and terminated by the caller.

* far_end_disconnect – the call was completed and terminated by the caller.

* network_disconnect – the call was completed and terminated by the network.

* maxtime_disconnect – the call duration exceeded the value of maxtime attribute and was terminated by the

platform.

* unknown – the outcome of transfer is unknown.

* not_allowed– not allowed to use the <transfer> element for this document.

* far_end_machine – detects an answering machine picking up the call when analysis attribute is set to true.

* far_end_fax – detects fax machine picking up the call when analysis attribute is set to true.

* invalid_phone_no – phone number in dest or destexpr attribute is not a valid phone number format.

* restricted_phone_no – phone number is restricted.

expr

The expr attribute is an expression that assigns the initial value of the field item variable. The <transfer> element will be executed only if the value is undefined. This attribute is optional and defaults to undefined.

cond

The cond attribute is a Boolean expression that must evaluate to true for this <transfer> element to execute. This attribute is optional and defaults to true.

dest

The dest attribute is a URI of the destination phone. The format is: tel://73643543. Some URI prefix have been added to extend the functionalities of the VoiceXML interpreter. This attribute is required – specifically dest or destexpr is required.

destexpr

The destexpr attribute is an expression that yields the destination. This attribute is required – specifically dest or destexpr is required.

bridge

The bridge attribute determines what to do after call is connected. Please refer to the description of this element for the 2 types of transfer. This attribute is optional and defaults to true.

* true – a bridge transfer.

* false –a blind transfer.

connecttimeout

The connecttimeout attribute is the time to wait while trying to connect the call before returning the noanswer condition. The minimum value allowed is 5s. This attribute is optional and defaults to 30s. (30 seconds)

maxtime

The maxtime attribute is the maximum time that the call is allowed to last. Minimum value is 30 seconds; maximum is one week, and 0 for no limit. This attribute only applies when bridge=true. This attribute is optional and defaults to 0.

transferaudio

The transferaudio attribute is the URI of the audio file to be played while connecting the call. This attribute is optional (not supported by VXI*).

Shadow Variables

The <transfer> shadow variable (name$) has the following property after the transfer completes:

name$.duration – the duration of a successful call in seconds. If the duration is 0, the call was terminated before it was answered.

Events Thrown

The following events can be thrown during the execution of the <transfer> element:

connection.disconnect.hangup – the caller hangs up.

connection.disconnect.transfer – the call was "blind transferred" to another line and will not return.

error.connection.noauthroization – the caller is not allowed to call the destination.

error.connection.baddestination – the destination URI is malformed.

error.connection.noroute – the platform is not able to place a call to the destination.

error.connection.noresource – the platform cannot allocate a resource to place the call.

Parents

<form>

Children

<audio>, <catch>, <enumerate>, <error>, <filled>, <help>, <nomatch>, <noinput>, <grammar>, <prompt>, <property>, <value>

Extensions

“exec:” or “tel:” (with an application name and ())

The <transfer> tag can be used to “execute” an Asterisk application.
Set the attribute dest with a URI like "exec:applicationname(parameters)".

Use “set” to set a variable:

<transfer name="setvar" bridge="true" dest="tel:set(VXML_NUMBER=10)">

Use “get” to get a variable value (value is in the value shadow variable):

<transfer name="getvar" bridge="true" dest="tel:get(URL)" />

<value expr="getvar$.value" />.

Use “function” to set a function:

<transfer name="maxduration" bridge="true" dest="tel:function(TIMEOUT(absolute)=10)">

Use '+=' to append values:

<transfer name="userfield" bridge="true" dest="tel:function(CDR(userfield))+=12" />

Use “function” to get a function value:

<transfer name="peerip" bridge="true" dest="tel:function(SIPPEER(user2:ip))">

<value expr="peerip$.value" />.

Use “conference :”

The <transfer> tag can be used to “push” the call into a conference. Set the attribute dest with a URI like "conference:x", where the value x is the id of the conference (you can add the conference parameters after a '/' too, or using the conferenceformat). The conference resources must be configured if more than two lines are set in call conferencing.

Use “celudan:”

The <transfer> tag can be used to “push” the call into a 3Gbuilder/Cedulan session. 3Gbuilder is a video resource to create and deliver dynamic web content in real time during the transfer. Set the attribute dest with a URI like "celudan:htpp://website", where the value website is the web content to play in real time (video and audio are supported).

Use “originate:” or “ori:”

The <transfer> tag can be use to generate an outgoing call and links it to an application or context. If the maxtime is set to 0, the transfer doesn’t wait the establishment. The variable content and events are set link a standard bridge transfer.

Example:

<transfer name="test" bridge="true" dest="ori:user1=vxml(account2)" connecttimeout="20s" maxtime="0s"/>

Limitations/Restrictions

The transfer now supports the bridge, with SIP re-inivite or a new session, and blind mode with SIP refer. The blind mode can be configured to use a specific asterisk application with the configuration parameter blindapplication=application_name (the parameter is the dest value). If you set the blindapplication with the empty value, the vxml Asterisk application will perform a Dial at the end of the blind session.

You should specify the channel used to generate the outgoing call. Use the application Dial syntax, for example, dest=”tel:SIP/600@localhost”.

You can use the parameter dialformat in the configuration file vxml.cfg to specify a string format, for example, dialformat=”SIP/%s@dialout”. You can use videodialformat for the video calls too. This parameters can be global in the general section or specific to an account.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
<form>
 <var name="calldur" expr="0"/>
 <block>
  <prompt>
   Welcome
  </prompt>
 </block>
  <transfer name="newcall" dest="0146120000"
   connecttimeout="10s" bridge="true">
   <filled>
    Your call lasted <value expr="newcall$.duration"/> seconds.
    <if cond="newcall == 'busy'">
     <prompt>
      All our customer care agents are currently busy.
      Please call back later.
     </prompt>
    </if>
   </filled>
  </transfer>
 </form>
</vxml>
```

<value>

Description

The <value> element inserts the value of an expression in a prompt.

Syntax

```
<value
expr="ECMAScript_Expression"/>
```

Attributes

expr

The expr attribute is an expression to return. This attribute is required.

Parents

<audio>, <block>, <catch>, <choice>, <enumerate>, <error>, <field>, <filled>, <help>, <if>, <initial>, <menu>, <noinput>, <nomatch>, <object>, <prompt>, <record>, <subdialog>, <transfer>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="maincourse">
   <noinput> <reprompt/> </noinput>
   <nomatch> <reprompt/> </nomatch>
   <prompt count="1">Please select an entree.</prompt>
   <prompt count="2">Today, we're featuring:<enumerate/></prompt>
   <option dtmf="1" value="fish">swordfish</option>
   <option dtmf="2" value="beef">roast beef</option>
   <option dtmf="3" value="frog">frog legs</option>
   <filled>
    <prompt>
     <value expr="maincourse"/>, good choice.
     Please enjoy your meal.
    </prompt>
    <submit next="maincourse.cgi" method="post"
namelist="maincourse"/>
   </filled>
  </field>
 </form>
</vxml>
```

<var>

Description

The <var> element declares a variable in the scope of its parent element. If the variable is already defined in the current scope, further declaration is treated as value assignments.

Syntax

```
<var
name="String"
expr="ECMAScript_Expression"/>
```

Attributes

name

The name attribute defines the name of the variable. This attribute is required.

expr

The expr attribute is the initial value of the variable. This attribute is optional and defaults to undefined.

Parents

<block>, <catch>, <error>, <filled>, <form>, <help>, <if>, <noinput>, <nomatch>, <vxml>

Children

None.

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <var name="world" expr="'world'"/>
 <form>
  <field name="hello">
   <grammar>hello | goodbye</grammar>
   <help>Just say hello</prompt>
   <prompt>Say hello</prompt>
   <noinput>Say something</noinput>
   <filled>
    <prompt>Hello,
     <value expr="world"/>!
    </prompt>
   </filled>
  </field>
 </form>
</vxml>
```

<vxml>

Description

The <vxml> element identifies a VoiceXML document (the xml root element).

Syntax

```
<vxml
version="Version"
base="URI"
xml:lang="Language"
application="URI" />
```

Attributes

version

The version attribute specifies the VoiceXML version number. The current version is 2.0. This attribute is required.

xmlns

The xmlns attribute is the namespace for VoiceXML, and must be http://www.w3.org/2001.vxml. This attribute is required.

base

The base attribute is the base URI of this document. Relative references in this document use this URI as the base URI. This attribute is optional.

xml:lang

The xml:lang attribute is the language and locale type for this document. This attribute is optional and defaults to en-US.

application

The application attribute is the URI of this document's application root document. If the root document cannot be found or the root document has a reference to another root document, an error.semantic event is thrown. This attribute is optional.

Parents

None.

Children

<catch>, <error>, <form>, <help>, <link>, <menu>, <meta>, <noinput>, <nomatch>, <property>, <script>, <var>

Extensions

None.

Limitations/Restrictions

None.

Example Code

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
 <form>
  <field name="hello">
```

```
   <grammar>hello | goodbye</grammar>
   <help>Just say hello</prompt>
   <prompt>Say hello</prompt>
   <noinput>Say something</noinput>
   <filled>
    <prompt>Hello, world!</prompt>
   </filled>
   </field>
</form>
```

5 Text-to-Speech (TTS)

The VXI* VoiceXML browser integrates an HTTP client interface to connect to an HTTP text-to-speech (TTS) server. This allows for dynamically generated audio content with a text-to-speech engine. Most VoiceXML browsers have an MRCP (Media Resource Control Protocol) interface to access text-to-speech features. The advantage however, of using the HTTP connector is that the “speech” generated is cached by the VoiceXML Browser, and re-used the next time. The text is posted via an HTTP request then the server responds with a standard wav file. Users can use the TTS to generate menus prompts, for example, without need to purchase a lot of TTS licenses. The integrated TTS packages use PHP, so you need to have an Apache/PHP already installed on your server (MySQL is optional, it allows to produce “CDR” like reporting information to evaluate the TTS using).

List of TTS Supported:

- Flite TTS (FREE)
- Loquendo TTS
- Ceptral TTS
- Acapela TTS
- Cepstral TTS
- Verbio TTS
- VoiceInteraction TTS
- Nuance TTS
- eSpeak (with/without Mbrola voices) TTS (FREE)

Installation Guide

For more information about the TTS engines installation procedures, please read:

I6NET-installation-guide-en-20XX-XX.pdf

NOTE:

To install and configure your TTS engine, please read our Installation Guide.

If you need any other TTS engine, please contact our team.

6 Text-to-Video (TTV)

The VXI* VoiceXML browser integrates an HTTP client interface to manage Text-to-Video contents. This allows to dynamically generating a video content from text. TTV includes audio using the Text-to-Speech interface to access text-to-speech features. The advantage however, of using the HTTP connector is that the “speech” generated is cached by the VoiceXML Browser, and re-used the next time. The text is posted via an HTTP request then the server responds with a standard h263/.h264 or mp4/3gp file. Users can use the TTS to generate menus prompts, for example.

Text-to-Video TTV package requires:

- VXI* VoiceXML Browser with a video key activation license
- Xtras* Video IP/3G package
- Any TTS engine (optional)

Installation Guide

For more information about the TTV installation procedures, please read:

I6NET-installation-guide-en-20XX-XXpdf

NOTE:

To use TTV you have to install first the video packages for VXI* (Xtras* Video IP/3G).

7 Automatic-Speech-Recognition (ASR)

The VXI* VoiceXML Browser uses the Digium/Asterisk speech connector bridge for Lumenvox, Vestec, VoiceInteraction and Verbio Speech engines. All these engines are made for Asterisk platforms. For MRCP compliant ASR engines you have to install our Xtras* uniMRCP package.

List of Supported ASR engines:

- Lumenvox ASR
- Verbio ASR
- Vestec ASR
- VoiceInteraction ASR
- Loquendo ASR (through MRCP)
- Nuance ASR (through MRCP)

Installation Guide

For more information about the ASR engines installation procedures, please read:

I6NET-installation-guide-en-20XX-XX.pdf

NOTE:

To install and configure your ASR engine, please read our Installation Guide.
If you need any other ASR engine, please contact our team.

8 Xtras* for VXI*

I6NET provides specific value added packages for VXI* VoiceXML browser to improve or extend your platform. These Xtras* suite packages are designed, tested and packaged to run only with VXI*. All these packages are only provided to VXI* customers with setup and our support. Please contact our sales team and tech team for any question or order.

List of Xtras* packages:

- Xtras* · Dialer
- Xtras* · Video IP/3G
- Xtras* · uniMRCP Client
- Xtras* · Outbound API (clic-to-call)
- Xtras* · Call RecorderMP4
- Xtras* · MPEG4IP Video Tools
- Xtras* · RTMP/Flash Server Channel

NOTE:

All these Xtras* require VXI* to run over Asterisk.

8.1 Xtras* · Dialer

Synopsis

Dialer is an addon for VXI* platforms to provide automatic outbound dialing over Asterisk®

Description

Dialer is a module to manage outbound IVR campaing with your VXI* platform. This application can be managed from the Linux Shell or from its simple and customizable web interface.

Package

dialer-2010-12-12.tar.gz

User Guide

I6NET-dialer-en-20XX-XX.pdf

8.2 Xtras* · Video IP/3G

Synopsis

Video module to run all video over IP and 3G functions for VXI* platforms.

Description

Video IP/3G is a special addon to convert your VXI* platform into a Video IVR or IVVR. I6NET has packaged the “app_h324m”, “app_mp4” and “app_rtsp” applications. With the application functions you should be able to receive and make h324m / h323 / SIP video calls and handle them in the Asterisk dial plan / VoiceXML as you would do with any other call from any standard channel.

Package

video-2010-12-12.tar.gz

User Guide

I6NET-video-IP-3G-en-20XX-XX.pdf

8.3 *Xtras* · uniMRCP Client*

Synopsis

MRCP Client for VXI* to connect MRCP compliant ASR and TTS engines.

Description

The Asterisk UniMRCP solution for VXI* consists of a speech resource module as well as an Asterisk® application. The res_speech_unimrcp use the standard Speech Asterisk API.

Package

unimrcp-2010-12-12.tar.gz

User Guide

I6NET-unimrcp-en-20XX-XX.pdf

8.4 *Xtras* · Outbound API*

Synopsis

Outbound API programs to make clic-to-call (HTTP-PHP) for VXI*

Description

This HTTP-API for VXI* is a buit in PHP to launch voice or video calls to a VoiceXML service from any external application like a CTI, CRM or any else business web servise.

Package

outbound-api--2010-12-12.tar.gz

User Guide

I6NET-outbound-api-en-20XX-XX.pdf

8.5 Xtras · Call RecorderMP4*

Synopsis

Call Recorder module (support both voice and video calls)

Description

Used to make a video recording of a channel.
The channel's input and output voice/video packets are logged to files until the channel hangs.

Package

recorder-2010-12-12.tar.gz

User Guide

I6NET-recordermp4-en-20XX-XX.pdf

8.6 Xtras · MPEG4IP Video Tools*

Synopsis

FFmpeg/Mpeg4ip Video Tools to generate and convert video contents for VXI*.

Description

The MPEG4IP video tools are a collection of application to convert and manage the video content compatible with Asterisk and the VideoAsterisk applications (mp4play/mp4save).

Package

mpeg4ip-2010-12-12.tar.gz

User Guide

I6NET-Mpeg4ip-en-20XX-XX.pdf

8.7 Xtras · RTMP/Flash Server Channel*

Synopsis

Special Flash (Adobe) Channel for VXI* over RTMP to make voice or video call over the web using any web browser with Flash Player 9 or upper.

Description

Used to make a voice or video call over Flash.
The server channel's input and output voice/video call using RTMP from Adobe®.

Package

flash-2010-12-12.tar.gz

User Guide

I6NET-flash-rtmp-en-20XX-XX.pdf

9 Complementary Software

I6NET recommends a suite specific software tools for VXI* VoiceXML browser platforms. All these packages are specific complementary products to create or edit contents or to improve your system. Please contact our sales team and tech team for any question.

List of Tools:

- WavePad Editor
- FF* Video Converter
- ImageMagick
- CDR Analyser (Stats)
- Inkscape Editor
- Apple QuickTime Pro
- Helix Mobile Producer
- Xenon Offline Encoder
- MP3player (mpg123)
- Zingaya MediaServer
- Arno Firewall

Complementary Software Guide

For more information about all the Complementary Software available for VXI*, please read:

I6NET-complementary-software-guide-en-20XX-XXpdf

NOTE:

Please read the products’ license conditions.
Most of them are Open Source Projects and Commercial Products from specific Vendors.

Reference Documents

Books

The VoiceXML Handbook

Do you know telephony but need to learn about the Web? Do you know the Web but need to learn telecom? Are you a Webmaster who needs to telephony-enable your site? If you answered "Yes" to any of these questions, then this book is for you. Bob Edgar explains everything you need to understand and use VoiceXML, the "HTML for telephony" which is revolutionizing the industry.

Not a telephony expert? Not a Web expert? No problem, Bob covers all the background you'll need: how the Web works, how telecom and computer telephony work, XML, voice browsers and more. Once you've understood the background, Bob starts with a "Hello, World" application--a VoiceXML page which answers the phone and speaks to you--then leads you step by step through all the features of VoiceXML, including VoiceXML 2.0.

Author: Bob C. Edgar

Format: Paperback, 1st ed., 481pp.

Publisher: C M P Books

Asterisk, The Future of Telephony

This bestselling book is now the standard guide to building phone systems with Asterisk, the open source IP PBX that has traditional telephony providers running scared! Revised for the 1.4 release of the software, the new edition of Asterisk: The Future of Telephony reveals how you can save money on equipment and support, and finally be in control of your telephone system.

If you've worked with telephony in the past, you're familiar with the problem: expensive and inflexible systems that are tuned to the vendor's needs, not yours. Asterisk isn't just a candle in the darkness, it's a whole fireworks show. Because Asterisk is so powerful, configuring it can seem tricky and difficult. This book steps you through the process of installing, configuring, and integrating Asterisk with your existing phone system.

Authors: Jim Van Meggelen, Leif Madsen, Jared Smith

Format: Paperback, PDF, 602 pages

Published: O'Reilly, 2007

Links

W3C

"Voice Browser" Voice Activity article: http://www.w3.org/Voice/

This article goes over applying Web technology to enable users to access services from their telephone via a combination of speech and DTMF. Voice Extensible Markup Language (VoiceXML) 2.1 (3/04). Fully backwards-compatible with VoiceXML 2.0, the draft's purpose is to standardize eight additional features implemented by VoiceXML platforms.

More information:

http://www.w3.org/TR/2004/WD-voicexml21-20040323/

VoiceXML Forum

The VoiceXML Forum is an organization of over 500 member companies worldwide. Its mission is to develop the industry-recognized certification programs that businesses need to demonstrate their product's interoperability to clients and business partners. It also takes a leadership role in training to expedite the implementation of VoiceXML. In addition, the VoiceXML Forum spreads the word about VoiceXML's future -- to encourage the development of even more applications built around it and expand the market for voice product and service offerings. While VoiceXML version 1.0 was developed within the VoiceXML Forum, the trademark for the specification was recently released to the public domain. The VoiceXML Forum is a program of the IEEE-Industry Standards and Technology Organization.

More information:

http://www.voicexml.org

The VoiceXML Review

News articles regarding VoiceXML.

More information:

http://www.voicexmlreview.org

The World of VoiceXML

Ken Rehor's World of VoiceXML includes news, events, reports and other useful information.

More information:

http://www.kenrehor.com/voicexml/

Asterisk, The Open Source PBX

Asterisk is a complete PBX in software. It runs on Linux, BSD, Windows and OS X and provides all of the features you would expect from a PBX and more. Asterisk does Voice over IP in four protocols, and can interoperate with almost all standards-based telephony equipment using relatively inexpensive hardware.

More information:

http://www.asterisk.org

VoIP-info.org

This is a very good resource for everything about Asterisk.

More information:

http://www.voip-info.org/wiki/index.php?page=Asterisk

OpenVXI

OpenVXI is a portable open source library that interprets the VoiceXML dialog markup language (www.voicexml.org). It closely follows the VoiceXML 2.0 draft specification, avoiding proprietary extensions of any kind.

More information:

http://www.speech.cs.cmu.edu/openvxi/

Glossary

Automatic Number Identification (ANI)

ANI is a service that provides the receiver of a telephone call with the number of the calling phone. The method of providing this information is determined by the service provider and can be useful for accounting and billing purposes, or to direct to a specific contact center group.

Confidence Score

The probability that the result returned by the speech engine matches what a speaker said. Speech engines generally return confidence scores that reflect the probability; the higher the score, the more likely the engine's result is correct.

Direct Inward Dialing (DID)

A telephony acronym for Direct Inward Dialing is a feature allowing callers to directly reach a PBX extension without an operator’s assistance.

Dialed Number Identification Service (DNIS)

DNIS data identifies which telephone number was dialed. A PBX often receives calls on the same port that were dialed to different 800 or 900 numbers, and the DNIS data contains the dialed number so that the PBX can track the call.

Dual-tone Multi-frequency (DTMF)

The tones produced by pressing keys on a telephone. DTMF, also called Touch-Tone, is often used as a way of sending data to IVRs. DTMF assigns a specific sound frequency, or tone, to each key so that it can easily be identified by a monitoring microprocessor. That frequency is then translated into a usable analog or digital signal.

Grammar

A grammar (in the context of speech recognition) is a file that contains a list of words and phrases to be recognized by a speech application. Grammars may also contain bits of programming logic to aid the application. All of the active grammar words make up the vocabulary.

Interactive Voice Response (IVR)

IVR is an automated system that allows callers to interact with a computer, using a telephone (or VOIP). An IVR may use speech recognition, DTMF, or a combination of the two.

Public Switched Telephone Network (PSTN)

This is the term which usually is the “cloud.” It represents a generic term for any non-dedicated data service such as dial-up analog service.

Speech Platform

A Speech Platform is a piece of software that runs the speech application. It follows the logic of the application, collects spoken audio, passes the audio to the speech engine, and passes the recognition results back to the application.

Speech Recognition (ASR)

The process by which a computer speech engine recognizes human speech

Text-to-Speech (TTS)

Text-to-Speech software converts text into voice output using speech synthesis engines.

Vocabulary

The total list of words a speech engine will be comparing an utterance against. The vocabulary is made up of all the words in all active grammars. The utterance is the word or phrase spoken by the caller.

VoiceXML

Voice Extensible Markup Language (VXML) is a mark-up language designed to code voice applications with many of the same architectural components as HTML. VoiceXML platforms connect to a combination of speech recognition engines, text-to-speech synthesis, video, telephony interfaces and VoiceXML Interpreter software to process the call.

VoIP

VoIP is an acronym for Voice over IP.   This technology enables voice delivered using the Internet Protocol. In general, this means sending voice information in digital form in discrete packets rather than in the traditional circuit committed protocols of the public switched telephone network.