VoiceXML Tutorial

Purpose

This tutorial teaches you how to develop Voice Extensible Markup Language (VoiceXML) based applications. VoiceXML is markup language that brings Web-based applications and content delivery to interactive voice response applications; input is in the form of audio (voice or DTMF keypad tones), as is output (text to speech voice or recorded audio files). This tutorial provides an introduction to basic VoiceXML development. It describes the tools you'll need to build VoiceXML-based applications, outlines some basic VoiceXML syntax, and concludes with a few sample programs. The tutorial is based on the version 2.0 specifications of VoiceXML.

Audience

The course is intented for developers and technical managers who want to get an overview and understandin of VoiceXML application development. To build and test VoiceXML applications, you'll need a ready to go Asterisk/VXI installation.

VoiceXML Overview

What is VoiceXML?

VoiceXML is a standard based on XML that allows Web applications and content to be accessed by a phone. You can develop speech-based telephony applications using VoiceXML. The standard was developed by the VoiceXML Forum, which was founded by AT&T, IBM, Lucent, and Motorola.

VoiceXML architecture model

This figure illustrates the components of the VoiceXML architecture model. The components include the following: * Document server: Processes requests received from the VoiceXML Interpreter and responds with VoiceXML documents. * VoiceXML Interpreter: Interprets the VoiceXML documents it receives from the document server. * Implementation platform: Controlled by the VoiceXML Interpreter context and VoiceXML Interpreter, the implementation platform generates events in response to user actions (for example, spoken or character input received) and system events (for example, timer expiration). The VoiceXML Interpreter context and VoiceXML Interpreter then handles the events.

VoiceXML gateway integration

VoiceXML relies on the network infrastucture to deliver and implement applications. Just like HTML is delivered to a Web browser, VoiceXML uses a voice browser for audio input and output. You use a regular phone to access a VoiceXML-based application. The voice browser runs on a voice gateway, which is connected to the public switched telephone network (PSTN, using Voip or) and to the network as illustrated in the figure.

VoiceXML applications

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

- Voice portals: 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: You can receive targeted information specific to the location you are dialing from. Applications use the telephone number you are dialing from.
- Voice alerts (such as for advertising): 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: 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.
- CTI: VoiceXML can identify and qualify a user before transfering it to a queue. Depending on the CTI integration method, the VoiceXML can push a context to the agent form associated to the user's interactions.
- Auto Attendent: VoiceXML can replace an operator to redirect calls. It can use the Automoatic Speech Recognition to recognize a user's name from a contact list and transfer the call to the associated extension.

VoiceXML explaining links

- Infographic Explaining VoiceXML
- What is VoiceXML PDF

VoiceXML concepts

Dialogs

The VoiceXML application structure model is a Finite State Machine. A user of a VoiceXML application is always in one conversational state or dialog at a given time. Each dialog is followed by another dialog, and if no dialog is specified next, then the execution of the VoiceXML application is terminated. There are two types of dialogs: forms and menus. Forms collect user inputs in the form of values, just like an HTML form. Menus present the user with a list of options to select from.

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 can share the same application root document. Root documents can be used to pass variables from one VoiceXML document to another. Any variables defined in the root document are available for all your documents.

Grammars

A grammar specifies a list of permissible vocabulary 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, doesn't respond correctly, requests help, etc. The VoiceXML interpreter also throws events in case there are any semantic errors in the VoiceXML document.

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

A basic VoiceXML application

Basic rules

Because VoiceXML is an extension of XML, you will have to follow the basic rules for XML, which include: * Well-formed XML documents: All elements must have a closing tag. * A root element: All documents must have a root element. The root element for a VoiceXML document is vxml. * Attributes: Attribute values must always be quoted. * Nesting: All elements must be properly nested.

<xml> version tag

It is recommended that the first line item in any VoiceXML document be the XML version tag:

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

<vxml> root tag

The rest of the VoiceXML document should be enclosed within the vxml tag:

```
<vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml">
```

Basic VoiceXML structure

Following is an outline of a basic VoiceXML application structure:

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

An example: Hello world!

In this example, you will create a simple Hello world! VoiceXML application. Copy and paste the

following code into the VoiceXML web server, configure an VoiceXML account on your VXI/asterisk installation and dial the number to test the application. To be able to run this example you need a TextToSpeech configured. You can have a look to the VoiceXML examples. There are other ways to have an Hello world! example using prerecorded audio files, without using the TextToSpeech :

The form and block elements are discussed in the next section.

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

Forms, menus, and links

What is a form?

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 the use of a form element:

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

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

Form items

Two types of form items exist: field items and control items. A field item prompts the user on what to say or key in and then collects the information from the user that is then filled into 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 types of field items:

- <field>: value of the field item is obtained from the user via speech or DTMF grammars
- <record>: value of the field item is an audio clip recorded by the user, such as a voice mail message, which can be collected by the <record> element
- <transfer>: used for transferring the user to another telephone number
- <object>: invokes platform-specific object with one or more properties
- <subdialog>: like a function call, invokes a call to another dialog on the current page or another VoiceXML document.

A control item's task is to help control the gathering of the form's fields. Following are two types of control items:

- <block>: sequence of statements used for prompting and computation

- <initial>: useful in mixed initiative dialogs that prompt the user for information

Form item variables and conditions

A form item variable is associated with each form. The form item 'variable by default' is set to 'undefined' initially and contains a result (collected from the user) once a form item has been intepreted. 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 next="http://www.news.com/hockey.vxml">
  Hockey
 </choice>
 <choice next="http://www.news.com/baseball.vxml">
  Baseball
 </choice>
 <choice 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: Identifies the menu
- scope: 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 to 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 example, the menu's prompt would be, “Say what sports news you are interested in: hockey; baseball; 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. Below is an example of the usage of the link element:

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

```
  <dtmf> 1 </dtmf>
 </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 element

A speech grammar specifies a list of vocabulary for the user to select from to interact with the VoiceXML application. Following is an example of how you could define a speech grammar:

```
<vxml version="2.0">
<form id="yyy">
<field name="xxx">
<grammar>
<![CDATA[
[
[excellent]
[good]
[fair]
[poor]
[help]
]
]]>
</grammar>
</field>
</form>
</vxml>
```

DTMF grammars – DTMF element

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:

```
<vxml version="1.0">
<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>
```

Event handling

Types of events

An event is thrown by the VoiceXML platform for any number of reasons, such as a user not responding to an input, not responding correctly, 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 element

The <throw> element generates predefined or user-defined events. An example of the use of the <throw> element is:

```
<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 element

The <catch> element catches the event thrown by the VoiceXML document, dialog, or form item. An example of the use of the <catch> element is:

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

The following is a set of catch elements available:

- <error>: catches events of type error
- <help>: catches events if no help is available
- <noinput>: catches events if there was no input by the user
- <nomatch>: catches events if an input by the user is not recognized as part of the active grammar

VoiceXML examples

Creating the welcome message

In this VoiceXML example, you will be creating an application that gives you a selection to choose

from. Once you make a selection, you are taken to the appropriate document or dialog. In this section, you will create the main greeting message of the application. In the code below, the user hears a “welcome” message and is then given a list of choices from the main menu. The <goto> element is used to skip to the menu section.

```
<?xml version="1.0"?>
<vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml">
 <!-- user hears welcome the first time -->
 <form id="intro">
  <block>
   <audio>Welcome</audio>
   <goto next="#make_choice"/>
  </block>
 </form>

 ...

</vxml>
```

Creating the menu

In this section you will create the menu for the VoiceXML application. The menu gives you four selections to choose from. DTMF is enabled, which allows for key presses by setting the dtmf attribute to 'true' in the <menu> element. The menu is created using the <choice> element and the next attribute specifies what document or anchor to go to. An event is caught, using the <catch> element, in case the user doesn't respond, says “help”, or the input by the user is not recognized.

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

 ...

 <menu id="make_choice" dtmf="true">
  <prompt>
   <audio>Say survey, weather, news, or quit.</audio>
  </prompt>
  <!-- next attribute takes you to the appropriate documents or anchor
within the current -->
  <choice next="http://www.hostname.com/survey.vxml">survey</choice>
  <choice next="http://www.hostname.com/weather.vxml">weather</choice>
  <choice next="#news_report">news</choice>
  <choice next="#quit_app">quit</choice>
  <catch event="nomatch noinput help">
   <reprompt />
  </catch>
 </menu>
</vxml>
```

Creating the anchors

In this section you will create the anchor sections that appear in the same document as the menu

section. The two choices, news and quit, go to the following sections:

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

 ...

 <!-- give the news -->
 <form id="news_report">
  <block>
   <audio>Sports news!
    Michael Jordan returned to the NBA hardcourt for the first time in
nearly 40 months.</audio>
   <goto next="#make_choice"/>
  </block>
 </form>

 <!-- quit the application -->
 <form id="quit_app">
  <block>
   <audio>Goodbye!</audio>
  </block>
 </form>
</vxml>
```

Creating the weather document

In this section you will create the document that gives the user weather information. Once the user says or selects weather from the menu, the following code is executed:

```
<?xml version="1.0"?>
<vxml version = "2.0" xmlns="http://www.w3.org/2001/vxml">
 <form id="weather_report">
  <block>
   <audio>
    It will be partly cloudy today.</audio>
   <goto next="http://www.hostname.com/main.vxml"/>
  </block>
 </form>
</vxml>
```

Creating the survey document

The survey document prompts the user to respond to a question. Once the user completes the survey, the user is taken back to the main menu. In the code below the user can respond to the survey question by selecting one of the choices from the active grammar. Below is the active grammar list for the survey document:

```
<grammar>
<![CDATA[
[
```

```
[one excellent dtmf-1] {<option "excellent">}
[two good dtmf-2] {<option "good">}
[three fair dtmf-3] {<option "fair">}
[four poor dtmf-4] {<option "poor">}
[five help dtmf-5] {<option "help">}
]
]]>
</grammar>
```

The user is prompted with a question – if the user does not say anything, or if there is no match with the active grammar, the user is asked the survey question again. If the user asks for help, the user hears the help information and is asked the question again.

After successfully responding to the survey question, the code within the <filled> element is executed and the user hears a confirmation message and is returned to the main menu.

```
  <!-- prompt the user what to do -->
  <prompt>
   Welcome to the survey! What did you think of your
   instructor? If you don't know what to do, say help
   or push 5.
  </prompt>
  <!-- if no match with active grammar list then user prompted again. -->
  <nomatch>
   What did you say?
   <reprompt/>
  </nomatch>
  <!-- executed if no input is provided by the user -->
  <noinput>
   Please input something!
   <reprompt/>
  </noinput>

  <!-- executed if help is requested -->
  <help>
   Please say what you think of your instructor.
   You can say excellent, good, fair, or poor.
   You can also push one for excellent, two for
   good, three for fair, and four for poor.
   <reprompt/>
  </help>

  <!-- on a successful input skip to the form confirmResponse -->
  <filled>
   <goto next = "#confirmResponse"/>
  </filled>
```

The entire VoiceXML application

- main.vxml
- weather.vxml

- survey.vxml