Interactive Digital

11 Avalon Circle
Smithtown, New York 11787
Phone (631) 724-2323
Fax. (631) 724-2262
www.voicexl.com

VoiceXL™ For VoiceXML

WHITE PAPER

July 14th, 2003

1. Introduction

This document describes how the patented VoiceXL™ process is implemented on VoiceXML based IVR platforms. VoiceXML application developers can now quickly and effectively implement VoiceXL™ on their speech and touch-tone voice applications using open-standards VoiceXML and JavaScript.

VoiceXL™ for VXML is written in JavaScript 1.5 (ECMAScript) and will run on any IVR Platform with a VXML interpreter. Applications using VoiceXL™ can be run locally or via the web in client/server mode. Adding the VoiceXL.js file to your VXML applications exposes a comprehensive set of functions for automatic adjustment of voice playback. These functions allow the developer to optimize VoiceXL™ for each unique voice application. Caller response times for each level in the call script are first measured with the application running in normal mode, without VoiceXL™. This valuable and application dependant data is then used to calibrate VoiceXL™ for optimal use on your platform.

VoiceXL™ for VXML uses Alternate Speed Audio files generated off-line and selected automatically at run-time by the JavaScript process code. As few as 3 - 4 well selected script nodes can provide as much as 3 - 5 seconds savings in call duration and the process is implemented on a section of script at a time, starting with the nodes that will produce the best results first.

2. Technology Background

The primary objective of an IVR system is to allow your customers to self-direct their telephone calls. In order for this process to be worthwhile for the customer, they must be able to navigate though your application using their voice or touch-tone as an efficient and effective input device. To the extent this objective is not achieved, a direct proportion of callers will simply opt out of the system and wait to speak to an agent. Worse still, they may simply hang up.

IVR systems must also be designed to be easily navigable by first-time callers. In order for the first-time caller to have a successful interaction, prompts need to be comprehensive, outlining the full range of options to the caller. Complex IVR systems may have many layers of options and menus to navigate before the caller arrives at the information they need. Repeat or expert callers may become frustrated with the length of these option lists.

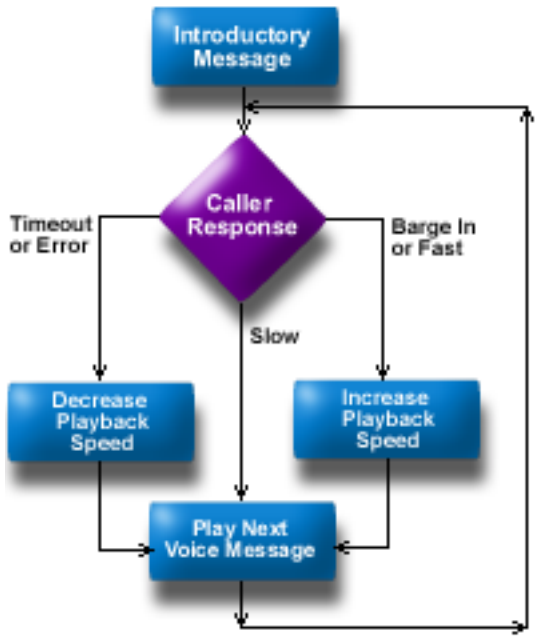
There are several ways of making IVR systems more navigable for callers. These include well-designed call scripts, allowing the caller to interrupt a prompt or menu with a selection and providing an option to bypass some of the prompts entirely. However, these options assume that the caller will always

remember the available choices further along in the script and that the caller is not distracted or in a noisy environment during the course of the call.

The optimal solution for experienced, novice and/or distracted callers is one that enables your IVR to automatically and continuously adjust the voice playback speed of your existing prompts to suit each type of caller on each call to the IVR system. This allows the experienced caller to move quickly to the desired information while still offering all of the guidance information necessary for a successful connection for a novice or distracted caller.

VoiceXL™ is an adaptive software process that continuously monitors the speed and accuracy with which callers are responding to your IVR prompts and menus. The process software then adjusts the voice playback speed of subsequent prompts accordingly, as shown in Figure 1. Under VoiceXL™, you can set a minimum, several intermediate, and a maximum playback speed for IVR playback, as well as the caller response times and events that will trigger an appropriate change in playback speed.

Figure 1
Simplified Process Flow for VoiceXL™



Today's voice applications are often required to provide information to callers under a wide variety of call specific circumstances. Each voice application is unique in its overall length and the complexity of each level in the call script. A simple voice mail/auto-attendant application might only ask the caller to select departments within the organization or to specify a name or extension. A more complex voice application could ask the caller to enter an account number, a PIN and then offer several choices to the caller via multiple menus.

In addition, each level in the application script has its own unique context and difficulty level for the caller. Most callers have little difficulty remembering the first five digits of their social security number, but many may not easily remember which PIN they used for a specific account or what the account number itself is. A five second audio file that asks the caller for a nine digit account number will generally require more time for an accurate response than a ten second audio file that simply asks the caller to select from one of three alternative input choices.

To further complicate matters, some of your callers will be using your voice application from the comfort and silence of their own home or office. Others will be calling from a noisy public phone or a cell phone with a poor connection. Add to this the fact that individual callers will respond to the same question at their own pace, comfort level, ability and knowledge and it becomes clear that this is indeed truly an individual centered, situation based process unique for each call.

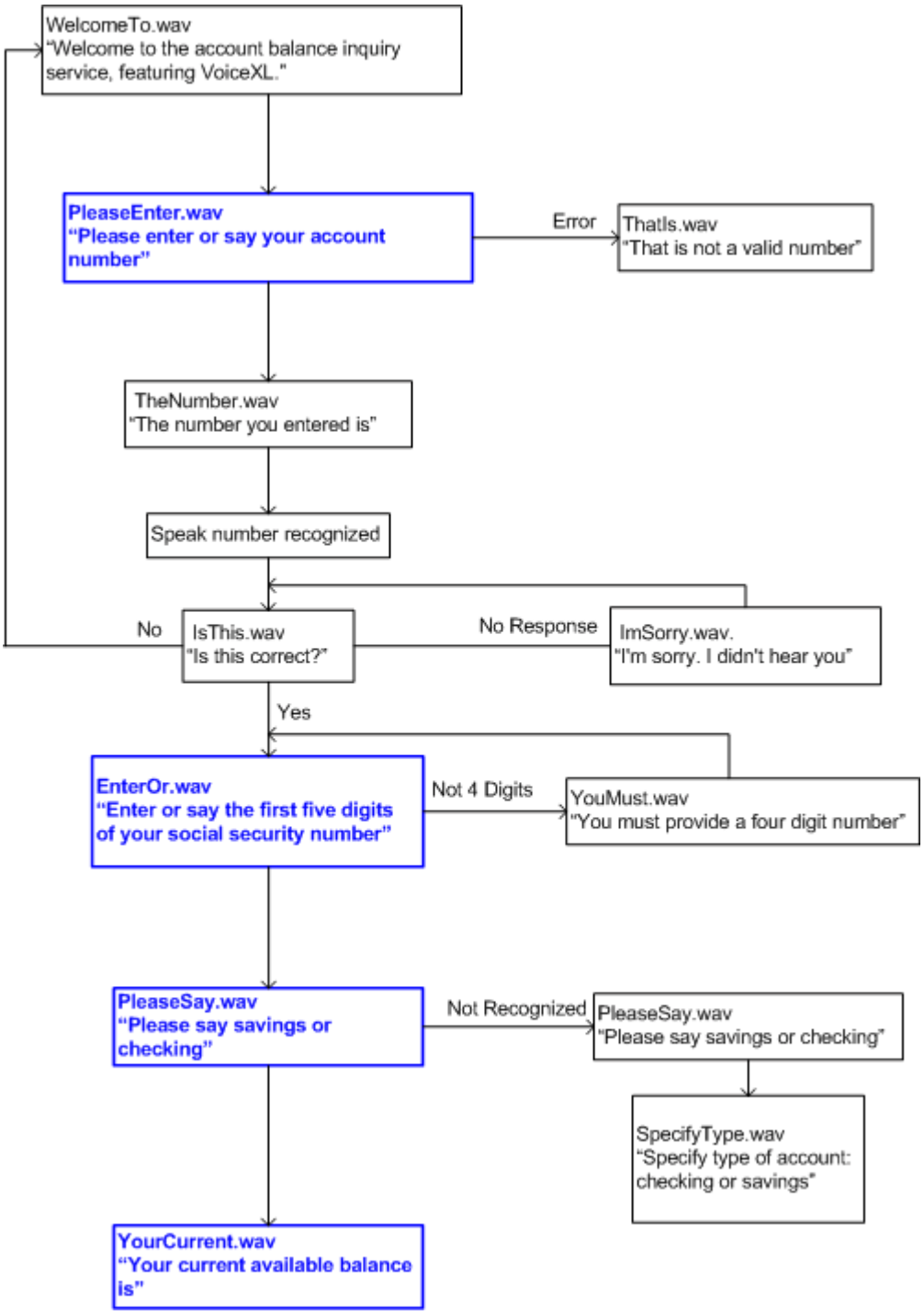
In order to account for this uniqueness among specific voice applications, we provide a calibration mode for VoiceXL™. This allows your application to run without VoiceXL™ in order to gather real-time feedback from your callers under live call circumstances. When your application runs with VoiceXL™ in calibration mode, there is no change in the playback speed of your audio and your callers hear the application as it sounds without VoiceXL™. However, VoiceXL™ tracks and logs how long it takes your callers to respond to each question/response pair (or VXML field) in which the process was implemented. This provides the valuable, real-time information inherent in your application to allow you to optimize VoiceXL™ during production.

VoiceXL™ allows your customers to set their own pace for self-directed calls. This increases customer satisfaction and productivity during the call. It also minimizes the average time it takes for the caller to receive the same amount of information from your IVR. Automatic changes in playback speed enable your IVR system to adjust to the individual caller just as a human receptionist would, increasing efficiency for experienced callers while providing inexperienced callers with the support they need. Increasing the efficiency of your IVR system saves money in reduced toll-charges and higher call throughput via your IVR system. VoiceXL™ has been proven to reduce call duration by 5 - 15 percent and improve IVR call containment rates by 1 - 5 percent based on the type of voice application.

3. Design Stage

Figure 2 shows the first iteration of a VoiceXL™ implementation on a typical bank account balance inquiry application. To begin the implementation process, we first identify which script nodes in the application which will have the VoiceXL™ process. Not all nodes may require or justify VoiceXL™ implementation and this can be done on a section of script at a time, starting with the

Figure 2
Retail checking/savings account implementation
VoiceXL™ script nodes shown in **blue**



3 - 4 nodes that will produce the best results first. The primary factor to consider here is whether or not the call volume through the node itself is substantial enough to justify VoiceXL™ process implementation. As few as 3 - 4 well selected script nodes can provide as much as 3 - 5 seconds savings in call duration on a call script that averages 90 seconds. An additional 4 - 5 seconds can be saved by adding the process to other carefully selected script nodes.

4. VoiceXML Application Integration

VoiceXL™ for VXML is written in JavaScript 1.5 and will run on any IVR Platform with a VXML interpreter. Applications using VoiceXL™ can be run locally or via the web in client/server mode. In Addition, VoiceXL™ for VXML uses Alternate Speed Audio files generated off-line and selected automatically at run-time by the JavaScript process code.

4.1 Alternate Speed Audio (ASA) file generation

Once it is determined during the design stage which ASA files will be needed, off-line voice editing tools (such as CoolEdit® or Sound Forge®) are used to generate these files from your original recordings.

These off-line sound editing tools change the playback speed of previously recorded .wav and other audio files by reducing or eliminating unnecessary elements of the playback stream. There is no distortion in the audio output stream and the file plays just like the original recording only slightly faster or slower in terms of words per minute spoken. The tools also support batch processing. This means that all of the ASA files required for a given voice application can be generated and maintained simply by running a single audio batch file.

The ASA files are placed in the audio directory containing the applications original audio files and named as shown in Figure 3.

Figure 3
Alternate speed audio file naming convention

Original Audio	**Alternate Speed 1**	**Alternate Speed 2**	**Alternate Speed 3**
audiofileA.wav	audiofileA1.wav	audiofileA2.wav	audiofileA3.wav ….etc.
audiofileB.wav	audiofileB1.wav	audiofileB2.wav	audiofileB3.wav ….etc.
audiofileC.wav	audiofileC1.wav	audiofileC2.wav	audiofileC3.wav ….etc.
....			
etc.			

Based on empirical data gathered in production runs, Interactive Digital has determined which speed increment and decrement values will provide the best results for a given voice application. These values depend not only on the speed of the original recordings, but also on the number of script levels and the complexity of each level.

4.2 VoiceXML application code modifications

All VoiceXL™ functionality for your application is contained in the JavaScript file VoiceXL.js provided by Interactive Digital. Once added to your VXML code as a standard ECMAScript add-on, the functions within this module are called from your VXML code to control voice playback throughout the application. Figure 4 shows the VXML implementation for the account balance inquiry application discussed earlier. A step by step description of this implementation follows.

4.2.1 Add the JavaScript file VoiceXL.js to your VXML code

To make the full complement of VoiceXL™ playback speed control functions available to your application, first add the following line to the root document of your VXML code as shown in Figure 4:

```
<script src="http://www.voicexl.com/VoiceXL.js"/>
```

4.2.2 Call vl_init() to initialize VoiceXL™

Place this function call in the VXML code that starts each session or call answered by the application. The function is called as follows:

```
vl_init(id_increments, id_decrements)
```

The calling parameters are:

- id_increments is the number of positive speed adjustments
- id_decrements is the number of negative speed adjustments

In the example shown in Figure 4, VoiceXL™ is configured with three levels of speed increments and no speed decrements.

4.2.3 Update the application <audio> tags

In order to allow for dynamic selection of the appropriate audio file, replace all <audio> tags that will incorporate the VoiceXL™ process in the VXML application as follows:

```
<audio src="'http://www.voicexl.com/audio/filename.wav"/>
```

with:

```
<audio expr="'http://www.voicexl.com/audio/' + vl_play('filename.wav')"/>
```

Figure 4
VoiceXL™ Implementation in VoiceXML

```
<?xml version=”1.0" ?>
<vxml version=”2.0">
<!— This VoiceXML code prompts the caller for their nine digit Checking or Savings account
number. The code changes implemented to support VoiceXL are highlighted in blue.  —>

<script src=”http://www.voicexl.com/VoiceXL.js”/>
   <form id=”prompt_account_number”>
        <script> vl_init(3, 0) </script>
         <block>
                <audio expr=”’http://www.voicexl.com/audio/’ + vl_play(‘WelcomeTo.wav’)”/>
        </block>
        <field name=”account” type=”digits?minlength=9;maxlength=9">
        <prompt>
                <audio expr=”’http://www.voicexl.com/audio/’ + vl_play(‘ToContinue.wav’)”/>
        </prompt>
        <noinput>
                <script> vl_noinput() </script>
                <audio expr=”’http://www.voicexl.com/audio/’ + vl_play(‘ImSorryYou.wav’)”/>
                <reprompt />
        </noinput>
        <nomatch>
                <script> vl_nomatch() </script>
                <audio expr=”’http://www.voicexl.com/audio/’ + vl_play(‘ThatIsAccount.wav’)”/>
        <reprompt />
        </nomatch>
         <help>
                <script> vl_help() </script>
                <audio expr=”’http://www.voicexl.com/audio/’ +  vl_play(‘Help.wav’)”/>
        </help>
        <filled>
                <script> vl_filled(“account”, 10) </script>
        <goto next=”#prompt_ss_number” />
        </filled>
     </field>
   </form>
```

4.2.4 VoiceXL™ Event Handler Functions

VoiceXL™ provides functions to account for the effects your applications NOINPUT, NOMATCH, HELP and FILLED events will have on playback speed.

4.2.4.1 Typically, you will want the application to temporarily and incrementally slow down playback when either a NOINPUT or NOMATCH event is received from the caller. In order to do this, place calls to the **vl_noinput()** and **vl_nomatch()** functions at the corresponding handlers for these events as shown in Figure 4.

4.2.4.2 When a HELP event is triggered by the caller, playback is set to normal playback (the original recorded rate) for the remainder of the session as this caller is assumed to be a novice if they are asking for help with the application. In order to do this, place a call to the **vl_help()** function at the corresponding handler for this event as shown in Figure 4.

4.2.4.3 When the caller triggers a FILLED event in the application, the primary function of VoiceXL™ is invoked causing a change in voice playback speed based on the relative speed of the callers response. In order to do this, place a call to the **vl_filled()** function at the handler for this event as shown in Figure 4. The calling parameters here are the <field> name (“account”) for this node and 10, indicating that the average response time as determined during the Calibration phase described below is 10 seconds for this field.

5. Run the Application with VoiceXL™ in Calibration Mode

VoiceXL™ also provides a Calibration Mode of operation.

Running VoiceXL™ in calibration mode allows you to gather the vital caller/application specific information inherent in each specific voice application. This information tells you exactly how long for example, your callers are taking to correctly enter their nine digit account number. It will be against these measurements that VoiceXL™ will determine later whether to adjust voice playback for a specific caller at a particular stage in the call script when running in production mode. When running in Calibration Mode, VoiceXL™, though present in your application, does not alter voice playback speed. Your application functions as though the feature were not present.

In order to run your application in calibration mode, simply run your application with the call to the vl_init() function shown in Figure 3 set as follows:

vl_init(0,0)

Let the application run for as long as it takes to complete a statistical sample of calls, then observe the Calibration Log File as shown in Figure 5 to see what the response times at each script level are. Make note of these response times and compute the average for use in your application production run.

Figure 5
Sample VoiceXL™ Calibration Log File

```
[Date Time] DEBUG Caller Response Times:
[Date Time] DEBUG Field Name/Response Time: account,8
[Date Time] DEBUG Field Name/Response Time: ssn,5
[Date Time] DEBUG Field Name/Response Time: checking_savings,3
[Date Time] DEBUG Field Name/Response Time: balance,12
```

6. Run the Application with VoiceXL™ in Production Mode

Upon completion of the calibration mode run, you will have enough data specific to your application to allow you to fine tune the VoiceXL™ response parameters for optimal performance of your application. Use this data to call each vl_filled() JavaScript function as follows:

vl_filled(FieldName, FieldAverage)

Where FieldName is the name given to this node in the script and FieldAverage is the average response time for this field as computed in Calibration Mode. With these parameters in place for each call to the vl_filled() function, you can now run the application in Production Mode with VoiceXL™ tuned for optimal performance.

7. Auto-calibration mode and Server Side Data Storage

When the pilot phase for VoiceXL™ has been completed and it is time to make VoiceXL™ part of your overall IVR deployment strategy, an additional feature is available to automate the calibration process.

Auto-calibration mode allows your voice applications to be automatically tuned for optimal performance based on data collected for each VXML field/response while the application is running in production mode.

This caller response information is stored on the application server and used to determine when to adjust voice playback during calls to the vl_filled() function. This means the VoiceXL™ calibration parameters can be automatically updated based on the most recent data available on the server. Once this mechanism has been implemented for your Server/IVR Platform it is available for use in all of your voice applications deployed in that environment. This feature is useful in providing automated updating of expected response times when your voice applications change.

About Interactive Digital, Inc.

Interactive Digital, Inc. specializes in the in the design and development of software for the voice response industry. Our mission is to provide innovative, timely and cost-effective solutions that bring enhanced productivity to our targeted markets.

Our commitment to customer satisfaction drives our excellence in development, production, deployment and customer support. Interactive Digital has partnerships and development agreements with Intel Dialogic, NMS Communications, Nortel Networks and Interactive Northwest, Inc. Our solutions target a wide variety of industries including healthcare, finance, travel and government.