



Developing X+V Applications Using the Multimodal Tools

IBM White Paper

May 2003

If you are a Web developer unfamiliar with VoiceXML, or a voice developer with no experience developing XHTML applications, the Multimodal Toolkit is an efficient method for you to quickly develop your multimodal applications.

Contents

Overview	3
The Multimodal Tools	3
What is XHTML?	4
What is VoiceXML?	4
What is XHTML+Voice?	4
Getting started.....	5
Getting up to speed quickly	5
Creating an X+V application	7
A sample application	7
Phase 1: Planning the application	8
Phase 2: Creating the multimodal project and file.....	9
Phase 3: Adding the visual component.....	9
Phase 4: Adding the voice component.....	9
Coding the easy way: Using Reusable Dialog Components	10
Coding VoiceXML from scratch	11
Phase 5: Creating the grammar	11
Phase 6: Linking the visual and voice components	12
Phase 7: Testing in the Multimodal Browser.....	12
Conclusion	13
References.....	13
Notices and trademarks	14
Trademarks	15

Overview

As computing devices become smaller and more pervasive, customers expect access to their data anytime, anywhere. With advances in the function, speed, and size of Personal Digital Assistants (PDAs) and cellular phones, coupled with an increasingly diverse set of users, the demands for flexible user interfaces from application developers have multiplied. Traditional visual interfaces, such as those provided by HTML pages, are no longer adequate to meet the public's rising expectations for convenience, performance, and usability. End-users of these embedded devices are no longer satisfied with low-resolution versions of desktop-based, Web applications and cumbersome methods of data entry. Consumers expect multiple methods, or modes, for interacting with a device. They want the ability to use the interaction method that most naturally fits the situation – to make the interface work for them, instead of being forced to work with an interface.

Traditionally, to create these “multimodal” applications, developers would have to master the development of both visual and voice software, resulting in a daunting learning curve. Many of these applications required extensive porting work to adapt to new platforms, and there was no way to leverage existing Web applications.

With IBM's 40-year commitment to voice technology, and emergent software and hardware making applications faster and more powerful, IBM has created a practical solution for application developers seeking to integrate both voice and visual technologies: the Multimodal Toolkit and Multimodal Browser.

This document provides an overview of the XHTML+Voice (X+V) language, an introduction to the development toolkit, and a general description of how to use the features in the toolkit to develop a multimodal application. For specific details on creating and implementing a multimodal application, refer to the documents that accompany the Multimodal Tools.

The Multimodal Tools

The Multimodal Tools release builds on the WebSphere® Studio framework to add the functionality you need to create, test, and run multimodal applications.

The **Multimodal Toolkit V4.1 for WebSphere Studio**, which adds extensions to a WebSphere Studio development product to provide multimodal functionality, introduces a user interface that can minimize both the skills and time needed to develop high-tech applications for PDAs and other handheld, wireless devices. IBM has used similar technology for years to facilitate the rapid development of server-based voice applications.

The Multimodal Toolkit provides an integrated development environment that lets you integrate visual and voice applications efficiently without requiring expertise in all the development languages. The toolkit provides multiple tools, editors, and views that are operated using standard menus, icons, toolbars, and basic XHTML and VoiceXML programming skills.

The toolkit's **Reusable Dialog Components** provide common functionality such as mailing address, credit card, and social security number form components using only a few button clicks, and each field provides the user with multiple methods of data entry.

The **WebSphere Everyplace® Multimodal Browser V1.0**, developed in a strategic relationship with Opera Software, provides a Web browser in which you can test voice-enabled Web applications. The browser is enhanced with extensions that include IBM's automatic speech recognition and text-to-speech technology, allowing you to view and interact with multimodal applications that you have built using XHTML+Voice. When you install the Multimodal Browser, the icon for the Opera Browser appears on your desktop, and you can use it to open the browser and run your multimodal applications.

The **Voice Server SDK V3.1.1** contains the programs that are needed to play and compose pronunciations in the Multimodal Toolkit.

Multimodal applications consist of visual (XHTML) and voice (VoiceXML) components.

What is XHTML?

The eXtensible HyperText Markup Language (XHTML) is an XML-based markup language for creating visual applications that users can access from their desktops or wireless devices. XHTML is the next generation of HTML 4.01 in XML.

If you have existing programs with HTML pages, you will have to make some simple structural changes to comply with XHTML conventions. XHTML has replaced HTML as the supported language by the World Wide Web Consortium® (W3C), so future-proofing your Web pages by using XHTML will not only help you with multimodal applications, but will ensure that users with all types of devices will be able to access your pages correctly.

For more information, refer to the XHTML 1.0 specification on the W3C Web site (see the [References](#) section at the end of this paper).

What is VoiceXML?

The Voice eXtensible Markup Language (VoiceXML) is an XML-based markup language for creating distributed voice applications, just as HTML is a language for distributed visual applications. VoiceXML was defined and promoted by an industry forum, the VoiceXML Forum™, founded by AT&T®, Lucent®, Motorola®, and IBM, and supported by approximately 500 member companies. Updates to VoiceXML are a product of the W3C voice working group. The language is designed to create audio dialogs that feature text-to-speech, pre-recorded audio, recognition of both spoken and DTMF key input, recording of spoken input, telephony, and mixed-initiative conversations. Its goal is to provide voice access and interactive voice response (such as by telephone, PDA, or desktop) to Web-based content and applications.

Users can interact with these Web-based voice applications by speaking or by pressing telephone keys rather than solely through a graphical user interface.

For more information, refer to the VoiceXML 2.0 specification on the W3C Web site (see the [References](#) section at the end of this paper).

What is XHTML+Voice?

XHTML+Voice, or X+V for short, is a markup language for multimodal Web pages. With X+V, Web developers can create Web pages that let end-users select voice input and output as well as traditional visual (GUI) interaction. X+V does this by providing a simple way to add voice markup to XHTML. Hence the name "XHTML plus Voice."

X+V fits into the Web environment by taking a normal visual Web user-interface and speech-enabling each part of it. That is, if you take a visual interface and break it up into its basic parts (such as an input field for a time of day, a check box for AM or PM, and so on), you can then simply enable the use of voice by adding voice markup to the visual markup. X+V consists of visual markup, a collection of snippets of voice markup for each element in the user interface, and a specification of which snippets to activate when. For visual markup, X+V uses the familiar XHTML standard. For voice markup, it uses a (simplified) subset of VoiceXML. For associating the snippets of VoiceXML and user-interface elements, X+V uses the XML Events standard. All of these are official standards for the Web as defined by the Internet Engineering Task Force (IETF) that governs web standards.

Motorola, Opera Software ASA, and IBM submitted the X+V specification to the W3C, which submitted it to the multimodal working group in January of 2002. For a Web site with the XHTML+Voice Profile 1.0 specification, see the [References](#) section at the end of this paper.

Note: The specific details of creating and implementing a multimodal application are beyond the scope of this white paper.

Getting started

The Multimodal Tools are Microsoft® Windows® 2000 programs requiring a minimum of an Intel® Pentium® 500 MHz processor or equivalent.

In addition, you must have an installed version of either:

WebSphere Studio Site Developer V5.0

(refer to <http://www.ibm.com/software/ad/studiositedev/>)

or

WebSphere Studio Application Developer V5.0

(refer to <http://www.ibm.com/software/ad/studioappdev/>).

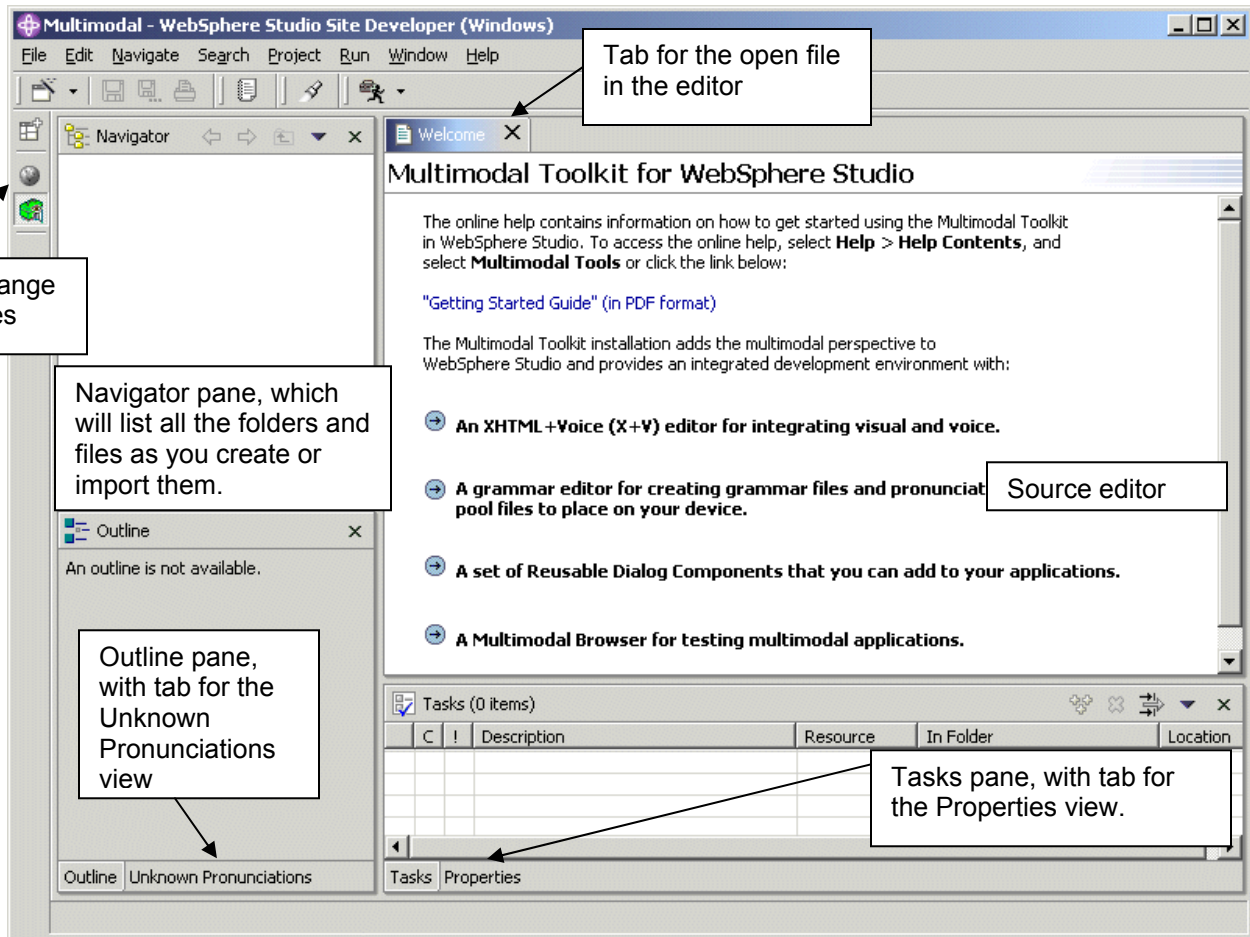
A single launcher panel simplifies the installation of the total package, including the **Voice Server SDK**, **Multimodal Browser**, and **Multimodal Toolkit**.

Getting up to speed quickly

The Multimodal Toolkit offers rich functionality and many resources to maximize productivity:

- **Getting Started Guide** (PDF format) with a basic scenario that you can re-create on your system to become familiar with the steps for creating a basic multimodal application.
- **Online help** (opened using Help > Help Contents > Multimodal Tools) with detailed information about creating multimodal projects, including X+V files, grammars, pronunciations, and audio files.
- **Content assist** with popup windows in the editors that list valid tags and elements, as well as descriptive information at the cursor position in the editor.
- **Related documents** including the *Reusable Dialog Components* guide and the Voice Server SDK *VoiceXML Programmer's Guide*, which provide detailed information, samples, and snippets that you can use in your applications.

Now take a look at the Multimodal Toolkit. To do this, open **WebSphere Studio**, and then change to the Multimodal "perspective" (or view) by selecting **Window > Open Perspective > Multimodal**. To see the Welcome screen, select **Help > Welcome > Multimodal Tools**. The figure below shows the Multimodal perspective.

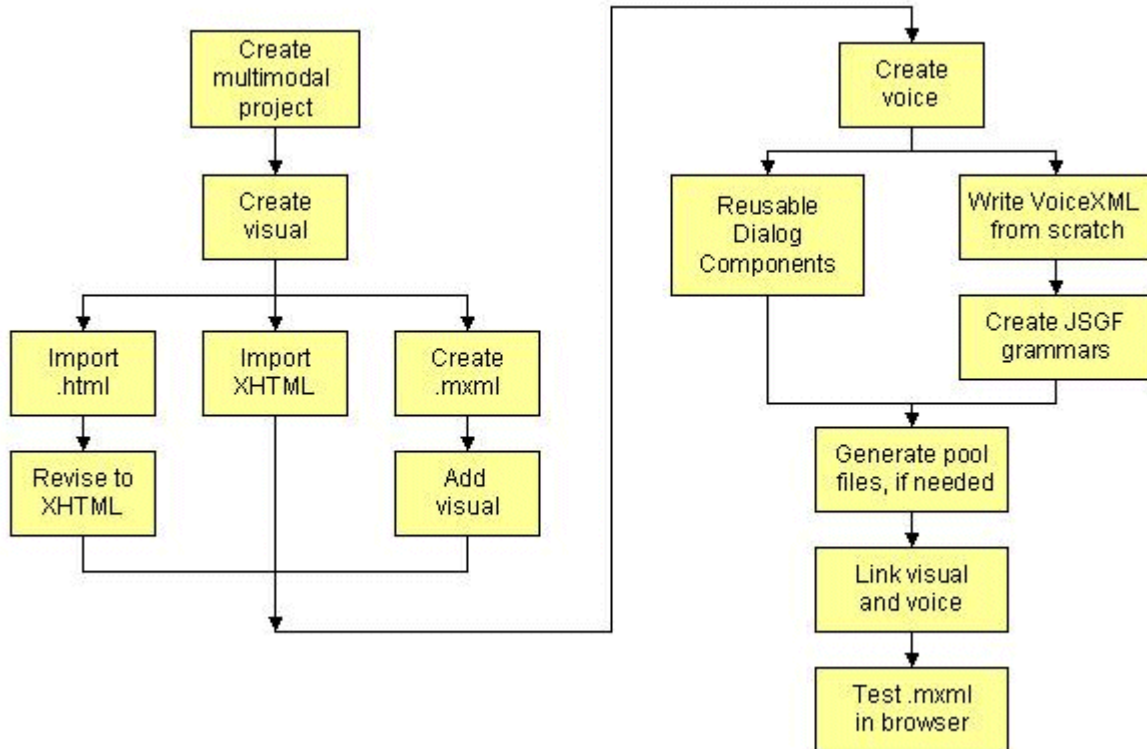


The Multimodal Toolkit perspective is composed of four panes. Looking at the illustration above, you can see how the Navigator, Outline, source editor (showing the Welcome message), and Tasks panes provide a coordinated layout of the information most useful for the application developer. The panes can each be resized, dragged to another location, or closed independently, letting you customize the view.

With a few clicks, you can create the initial folder and basic files needed to start an application. Experience with HTML Web pages and basic VoiceXML will increase the speed of development. The toolkit is rich with features that simplify each step. Each editor, including the X+V editor, has customized menus, right-click (contextual) menus, and toolbar buttons. A new tab appears above the source editor for each open file. And you can use the Import wizard to add your existing files into the toolkit.

Creating an X+V application

The following flowchart illustrates the basic process of completing a multimodal application using the Multimodal Toolkit.



The flowchart shows each milestone in the development process, which might also involve multiple substeps. The details in performing each of the substeps is beyond the scope and the objective of this paper; however, the Multimodal Tools comes with a *Getting Started Guide*, a how-to manual for developers, which includes guided practice in the substeps not described here.

A sample application

With the flowchart in mind, let's look at an example of an application that you might find on the Web today, shown in the figure below. In this example, you will see how to voice-enable fields in a typical transaction page that collects a customer's billing address and credit card information.

Transaction example using X+V - Opera

File Edit View Navigation Bookmarks Mail Window Help

Opera OperaMail eBay Amazon Super search Amazon.com search Find in page search

Transaction example u...

file://localhost/C:/workspace/transaction/Web Content/trar Go Google search 100%

Enter your billing address and credit card information

Address:

City:

State:

ZIP Code:

Phone:

Credit Card:

Credit Card Number:

Expiration Date (mm/yyyy):

This sample page consists of multiple components: address, city, state, zip code, phone, credit card type, credit card number, and expiration date. A complete application would consist of even more. For the purposes of this document, our example will focus just on the **city field** and **credit-card type field**.

Tip: If you are starting a multimodal application from scratch, without any existing HTML, it is always a good idea to write and test the visual portion of the application first, keeping in mind that the next task will be to determine which portions on the visual page you want to voice-enable.

Phase 1: Planning the application

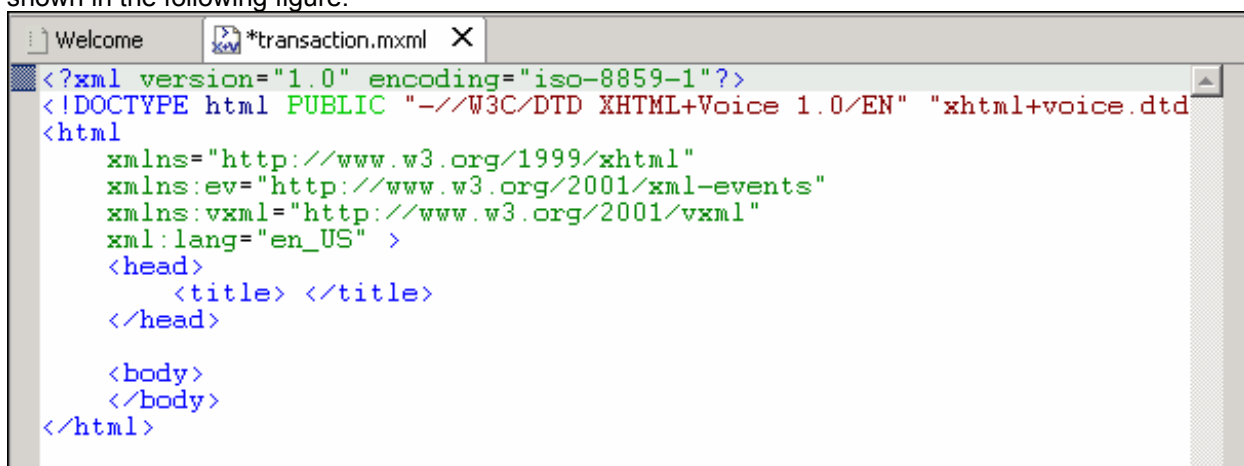
It is vital to plan the application in detail before the actual development begins. During this phase, you make high- and low-level design decisions that can assist developers in creating standardized, well-behaved multimodal applications, as well as reduce development time by taking some of the guesswork out of interface design. Planning will increase the usability of the interface, reduce the end-user's learning curve, and increase the effectiveness of the application in meeting your business needs. A thorough description of design decisions is included in the *VoiceXML Programmer's Guide*, packaged with the toolkit and the Voice Server SDK.

Just for starters, you will have to decide how and when end-users will add information so that you can match the visual and voice design. Keep in mind that all the words users could say must be added to a grammar file so that the speech recognition engine can match the spoken words to usable data. Effective designs use simple and consistent language for instructions, clear action verbs, such as "Select your credit card type," and consistent phrasing throughout the application.

Designers can also investigate and take advantage of the tools and features that are included in the toolkit. One of the key features is the built-in set of **Reusable Dialog Components**. When building an X+V application using the toolkit, you can use dialog components, which are ready-to-use sets of VoiceXML source code for common functions. This will enable you to quickly and easily add voice functions to your applications, thus saving time and reducing the amount of code you need to write. Dialog components can also be customized and reused within an application or in other applications.

Phase 2: Creating the multimodal project and file

With the design aspects in mind, you are now ready to start an X+V application. After opening the toolkit, you will use the New Project wizard to create a multimodal project in which you will store all the files needed for developing your application. Then you will use the New File wizard to create your first multimodal file, with file extension **.mxml**, the file format used by the Multimodal Toolkit for an X+V application. The X+V editor creates a new file in the toolkit with a pre-filled heading and basic tags as shown in the following figure.



```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML+Voice 1.0/EN" "xhtml+voice.dtd"
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ev="http://www.w3.org/2001/xml-events"
  xmlns:vxml="http://www.w3.org/2001/vxml"
  xml:lang="en_US" >
  <head>
    <title> </title>
  </head>
  <body>
  </body>
</html>
```

Phase 3: Adding the visual component

If you already have visual applications that you want to voice-enable, you can use an import wizard, drag & drop files into the Navigator, or cut & paste existing HTML into the multimodal project. You can also write XHTML code directly in the X+V editor.

All visual markup must comply with XHTML conventions. This revision can be done before or after you add the HTML in the toolkit. XHTML documents must be well formed, with each element properly closed. All HTML elements must be nested within the `<html>` root element. See the **XHTML 1.0 specification** included in the [References](#) section.

For best results, test the XHTML file to make sure it is correctly revised before adding it into the project. A good resource for testing and development is <http://validator.w3.org>. You can use the Web site to check whether your page is valid XHTML.

Phase 4: Adding the voice component

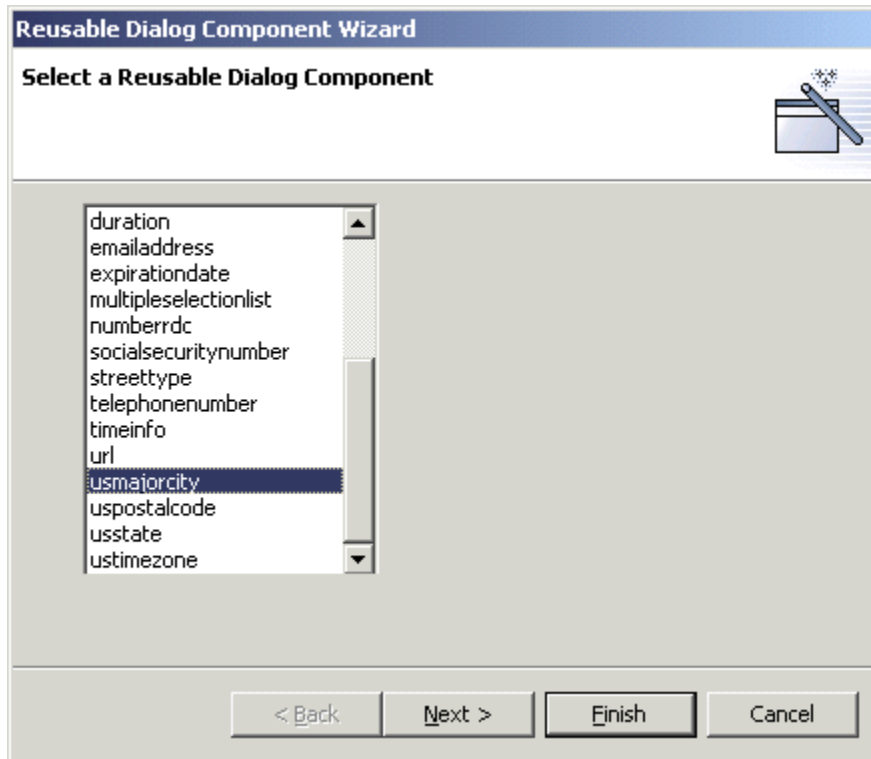
We will need to come back to the visual part later, but now you are ready to begin coding the VoiceXML, or voice portion. The VoiceXML code should be nested within the `<head>` element of the XHTML document, as you will see in the following example. You can write the voice portion using two methods: coding VoiceXML from scratch, or using the built-in Reusable Dialog Components to add the voice input.

To demonstrate the options, we will use a simple example of each to voice-enable the city field and the credit card field.

Coding the easy way: Using Reusable Dialog Components

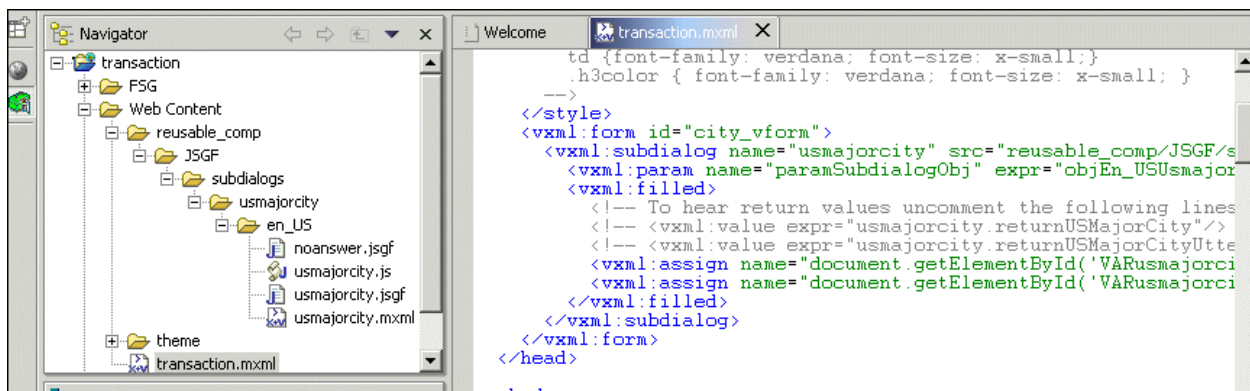
In place of, or in addition to, writing your own code, you can use the built-in Reusable Dialog Components to add common functions to your VoiceXML file. Each subdialog includes sample calling code (file extension **.mxml**) that you can copy and paste into your VoiceXML file.

In the Reusable Dialog Components wizard (shown below), you can select and customize a dialog component and then import it within a VoiceXML `<vxml:form>` element, with a unique id to reference in the HTML.



A quick look at the available dialog components will reveal the candidates for your application. For this example, we can use the toolkit's **usmajorcity** dialog component to voice-enable the "city" input field, thus eliminating the need to manually add all the major cities into a grammar file.

After you click the **Finish** button, the toolkit creates a **reusable_comps** folder in the Navigator that contains all necessary files imported with the newly created Reusable Dialog Component in the VoiceXML file. In the X+V editor (as shown in the example below), notice that the content inside the filled tag is commented out. This gives you the freedom to specify any actions you intend to take place after the field has been filled.



The usmajorcity dialog component is a two-part component: The “callee” subdialog prompts the user for a major US city and returns the user’s response to its caller dialog. The “caller” subdialog, as shown in the figure above, contains a `<vxml:filled>` element in which you can specify actions to be performed.

Coding VoiceXML from scratch

You will probably have to write some VoiceXML code from scratch. The example below shows how to code VoiceXML to voice-enable the credit-card type field.

```
<vxml:form id="creditcard_type_vform">
  <vxml:field name="cc_type_field">
    <vxml:grammar src="cc_type_gram.jsgf" />
    <vxml:prompt>
      Please select your credit card type
    </vxml:prompt>
    <vxml:filled>
      <vxml:assign name="document.getElementById('cc_type').value"
        expr="cc_type_field" />
    </vxml:filled>
  </vxml:field>
</vxml:form>
```

The VoiceXML code shown in the example above consists of a VoiceXML form element that contains other VoiceXML elements, such as the form id, field name, grammar file, voice prompt, and actions to be taken. The `<vxml:form>` element sits at the top of the VoiceXML element hierarchy and must be included in all XHTML+Voice applications.

In our example, the form corresponds to the credit-card type component in HTML and can contain the elements necessary to voice-enable the HTML counterpart. Let’s examine the elements contained in the form one by one.

- The `<vxml:field>` tag is used when you want to voice-enable a text field in HTML. In our example, the field element contains the other elements: `<vxml:grammar>`, `<vxml:prompt>`, and `<vxml:filled>`.
- The `<vxml:grammar>` element, or tag, imports an external grammar file, which contains all the words that can be spoken by the user. This type of grammar is called the "external" grammar. In the example, the grammar tag fetches a grammar file. Creating grammar files is an easy task when you use the toolkit and will be covered in the next phase.
- The `<vxml:prompt>` element contains a synthesized message (created using the Text-to-Speech engine) to be played to the user when the field element that contains it gets called. Thus, when the credit-card type component gets focus, the user will hear “Please select your credit card type.”
- The `<vxml:filled>` element contains all actions to be taken/executed when its parent field element has been filled. In other words, the users have input (using voice or touch) their credit card type. In our example, the action specified is to fill in the text field with what the user has said. The action is performed by the `<vxml:assign>` element, and one line of code is all it takes to accomplish it.

Phase 5: Creating the grammar

All the words that you want the speech recognition engine to recognize from an end-user's input must be included in a grammar. A grammar in a voice application uses a particular syntax, or set of rules, to define the words and phrases that can be recognized by the engine. A grammar can be as simple as a list of

words, or it can be designed with more flexibility and variability so that a more natural language capability is achieved. The design of grammars is important to achieving accuracy and customer satisfaction.

The speech recognition engine provides default pronunciations for words in the grammars. If you want to customize or create additional pronunciations, you can generate a pronunciation pool file for the grammar. When your multimodal application executes, the speech recognition engine uses the information in the grammars and pool files to match the phrase spoken by the user to one of the valid utterances specified in the grammar.

The Multimodal Toolkit simplifies the development of rule grammars and aids in testing these grammars to enhance robustness and performance. The toolkit supports Java™ Speech Grammar Format (JSGF) development and provides an environment in which you can write, syntax check, compile, and test source code.

If you use a dialog component for your VoiceXML code, the .jsgf file is automatically imported into the project.

If you write VoiceXML from scratch, you can open the JSGF editor in the toolkit and add the words into the grammar editor. The example below shows an external .jsgf grammar file for the credit-card type field.

```
#JSGF V1.0 iso-8859-1;

grammar cc_type_gram;

public <cc_type_gram> = Master [Card]
    | Visa [Card]
    | Discover [Card]
    | Novus [Card]
    | ((American Express) | Amex) [Card]
    | Diners Club [International] [Card];
```

Phase 6: Linking the visual and voice components

Next you will add the links in the visual portion at the points where you want to access the voice portion using an event handler. An event handler specifies an action to be performed when a particular event (such as a mouse click) takes place. In XHTML+Voice, event handlers enable interaction between XHTML and VoiceXML markup, as shown in the example below:

```
<td align="left">
<input type="text" id="cc_type" name="cc_type" value="VISA" size="20"
    ev:event="focus"
    ev:handler="#creditcard_type_vform" />
</td>
```

In the example, whenever the user clicks on the <input> element, a "focus" event is generated. This causes the VoiceXML form with ID "creditcard_type_form" to be activated.

Phase 7: Testing in the Multimodal Browser

Once you finish the application, and it contains a completed .mxml file with accompanying grammars, you can test the application on the Multimodal Browser to iron out any remaining problems before you deploy the application. You can launch the browser from the X+V editor, or open the file from the browser, and then put the application through its paces, just like an end-user.

Conclusion

The Multimodal Toolkit is a straightforward, robust tool for application developers who want a single interface in which to develop multimodal applications. It seamlessly combines the built-in features of the toolkit with the emerging X+V language. This frees developers from tedious or repetitious tasks, and lets them focus on the content and quality of their applications. IBM recognizes the importance of getting your multimodal applications running quickly, efficiently, and in a way that provides added value to your customers. The Multimodal Toolkit and Browser make a quick path to take advantage of the technological leaps in voice recognition software and to integrate these advances with your new or existing Web content.

References

Visit the **IBM Multimodal Web site** for Frequently Asked Questions (FAQs), white papers, and other product information: <http://www.ibm.com/pvc/multimodal>

This release of the Multimodal Tools is based on the following versions of specifications:

- XHTML+Voice Profile 1.0 specification:
<http://www.w3.org/TR/2001/NOTE-xhtml+voice-20011221>
- XHTML 1.0 specification:
<http://www.w3.org/TR/xhtml1/>
- VoiceXML 2.0 specification
<http://www.w3.org/TR/2003/CR-voicexml20-20030220/>
- XML Events specification:
<http://www.w3.org/TR/xml-events/>
- Java Speech Grammar Format specification:
<http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/>
- Semantic Interpretation for Speech Recognition (SISR) specification:
<http://www.w3.org/TR/2003/WD-semantic-interpretation-20030401/>

Notices and trademarks

This publication was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1784
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department T01B
3039 Cornwallis Road
Research Triangle Park, NC 27709-2195
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks or registered trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- Everyplace
- IBM
- WebSphere

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.