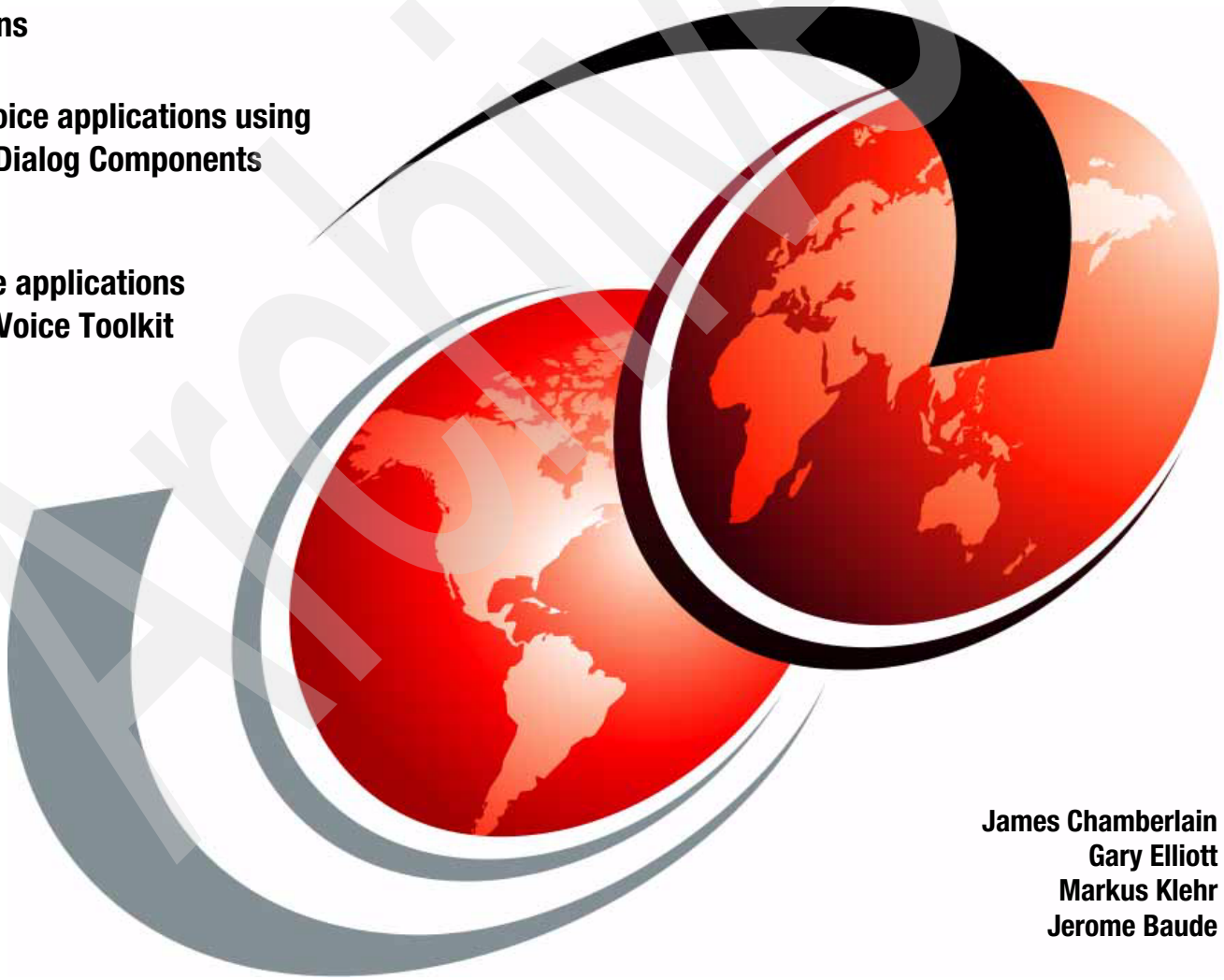
WebSphere. software

IBM

# Speech User Interface Guide

- Leverage the efficiency of your voice applications
- Develop voice applications using Reusable Dialog Components (RDCs)
- Tune voice applications using the Voice Toolkit



James Chamberlain
Gary Elliott
Markus Klehr
Jerome Baude



ibm.com/redbooks

Redpaper

International Technical Support Organization

# Speech User Interface Guide

May 2006

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.



**First Edition (May 2006)**

This edition applies to WebSphere Voice Server V5.1.3, Rational Application Developer V6.0.1, and Voice Toolkit V6.0.1.

# Contents



# Notices

This information 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 is not intended to state or imply that only that 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 10504-1785 U.S.A.*

*The following paragraph does not apply to the United Kingdom or any other 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.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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.

# Trademarks

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

Redbooks (logo) ™
AIX®
DB2®
Everyplace®
IBM®
Lotus®
PartnerWorld®
Rational®
Redbooks™
Tivoli®
WebSphere®

The following terms are trademarks of other companies:

EJB, Java, JavaServer, JavaServer Pages, JDK, JSP, J2EE, J2SE, Sun, Sun Microsystems, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

# Preface

To acquire new customers and to retain existing customers, companies must give credence to customer satisfaction. Call centers with well crafted speech-enabled applications significantly improve customer satisfaction as well as provide customers with additional services and 24x7 support.

These speech-enabled applications rely on the development of pleasant and efficient Speech User Interfaces (SUIs). Companies often fall short of fulfilling customer needs when the Speech User Interface has not been well designed and well tuned. Customers may go away feeling very frustrated when one of these speech-enabled applications does not meet their expectations.

WebSphere® Voice Server can be used in a call center to increase the call automation rate and reduce operating costs using Automatic Speech Recognition (ASR), Text to Speech (TTS) resources, and prerecorded audio. Fulfilling these goals lead to a decrease in labor cost, which result when callers are transferred to customer service representatives. If the Speech User Interface is improperly designed, constructed, and tuned pre- and post-deployment, unreasonable and unnecessary expenses will be incurred that can lead to critical situations, increased problem activity, and decreased customer satisfaction.

Speech-enabled applications involve highly complex technologies. Therefore, you must develop Speech User Interfaces with great care to realize their full potential.

In order to craft an effective SUI, you should follow proper and proven methodology (best practices). This Redpaper will detail an effective methodology that you can use to create and deliver high quality Speech User Interfaces to meet business needs. The IBM® WebSphere Voice Toolkit V6.0.1 is used throughout to test and tune WebSphere Voice Server to ensure it is optimally configured for your SUIs. However, specifics about programming languages used, such as VoiceXML, are beyond the scope of this Redpaper. While this Redpaper does summarize the core characteristics of the SUI, it refers readers to the following references noted in “Related publications” on page 135:

- *IBM VoiceXML Programmer's Guide* for detailed implementations in VoiceXML.

  http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/topic/com.ibm.voicetools.reldocs.doc/pgmguide.pdf

- *Using IBM Text to Speech Technology and Speech Synthesis Markup Language*

  http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/topic/com.ibm.voicetools.reldocs.doc/tts_ssml.pdf

- *Prototyping SUI with VoiceXML and Voice Toolkit*

  http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/topic/com.ibm.voicetools.reldocs.doc/vxmluiproto.pdf

This Redpaper leads the reader into design considerations of SUI implementations. And unlike existing IBM or publicly available books, it will introduce RDCs at a conceptual level and cover post-deployment considerations, such as tuning and data collection. It details how RDCs can be applied to WebSphere Voice Server. As an added value, the Redpaper will utilize WebSphere Voice Toolkit for examples of RDCs. It shows how to properly construct grammars to use for post-deployment tuning of the speech server.

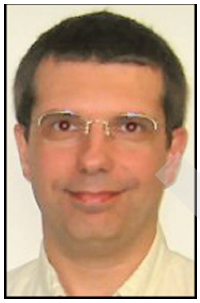
This Redpaper includes:

- An overview of the Speech User Interface in Chapter 1, “Introduction to developing Speech User Interfaces” on page 1
- An introduction to features of the IBM WebSphere Voice Toolkit and Rational® Application Development Platform in Chapters 2-7
- A detailed description of Reusable Dialog Components for speeding up the creation of your voice application in Chapter 8
- Topics about how to tune your voice application in Chapter 9

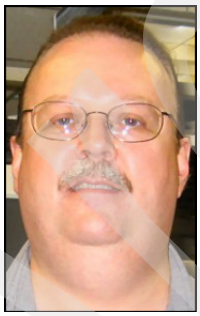
We assume that the reader has a basic knowledge of VoiceXML and grammar development in voice applications.

## The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.



**James Chamberlain** is a Senior Software Engineer and certified Senior IT Specialist. He is a project leader at the ITSO, Raleigh Center. He has over 24 years experience in the IT industry and specializes in pervasive computing technologies. His areas of expertise include e-commerce, pervasive computing, portals, AIX®, Linux®, and Java™ programming. He also architects, designs, and develops solutions using J2EE™, XML, Web Services, and IBM software products including WebSphere and DB2®. Before joining the ITSO, James worked for IBM Global Services on e-commerce system development for IBM Business Partners. He majored in Computer Science at Iowa State University.



**Gary Elliott** is a Software Engineer and Solution Architect for WebSphere Business Partner Technical Enablement in Austin, Texas. He has worked with numerous business partners in various industries related to the integration of pervasive solutions for enterprises. His experience includes integrations with WebSphere Voice Server, WebSphere Voice Application Access, and new device enablement with WebSphere Everyplace® Access. He holds a Bachelor of Science degree in Electrical and Computer Engineering from Clarkson University in Potsdam, NY, and a Master of Science degree in Computer Engineering from Syracuse University, Syracuse, NY. He has written previously on WebSphere Portal deployments on VMware, WebSphere Voice Application Access deployments, and was an author for the IBM WebSphere Voice Server for Multiplatforms V5.1.1/V5.1.2 Handbook.

**Markus Klehr** is a Software Developer in the IBM development lab in Germany. He joined IBM in 2002 and is the EMEA test Lead for the language technology. His expertise includes Voice over IP (VoIP), VoiceXML, and MRCP, as well as the language technology components.

**Jerome Baude** is a Speech Engineer for Speech Lab Based Services in the Paris Laboratory, France. He has been working on acoustic modelling, automatic speech recognition tuning, and language components for WebSphere Voice Server. His areas of expertise include ASR and TTS technologies, development and implementation of Speech Technology Applications, Pronunciation Modelling, and VoiceXML.

Thanks to the following people for their invaluable contributions that made this project possible:

**Jeff Kobal**
IBM Pervasive Computing Division, Enterprise Voice Products Development

**Aimee Sliva**
IBM Pervasive Computing Division, Enterprise Voice Products Development

**Girish Dhanakshirur**
IBM Pervasive Computing Division, Enterprise Voice Products Development

**James Lewis**
IBM Pervasive Computing Division, Enterprise Voice Products Development

**Baiju Mandalia**
IBM Pervasive Computing Division, Enterprise Voice Products Development

And a specials thanks to our ITSO support staff at the International Technical Support Organization, Raleigh Center:

Margaret Ticknor
Jeanne Tucker
Tamikia Barrow
Linda Robinson

Thanks to our ITSO management:

Jere Cline

And a special thanks to our IBM Pervasive Computing sponsor:

Mary Fisher

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks™ in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

1

# Introduction to developing Speech User Interfaces

How do you write good Speech User Interfaces? Some may answer that it depends highly on the application type. There are many ways of approaching the issue, but can we identify a set of best practices and a general methodology?

In this chapter, we first enumerate some advantages of speech applications.

We raise a few human factors to bear in mind when building speech applications. But, we are far from being exhaustive here. A wide range of scientific disciplines can be involved in the study of these aspects. We just want to make you sensitive to the speech area, its power, and its constraints.

Then, we propose a methodology and go through the case of an already existing application that we would like to speech enable.

You can find some of the matters described below in the *IBM VoiceXML Programmer's Guide*. We encourage application developers to read it for additional information.

Find the *IBM VoiceXML Programmer's Guide*:

```
http://www.ibm.com/support/docview.wss?rs=426&context=SSMQSV&dc=DA400&uid=swg27006148&loc=en_US&cs=utf-8&lang=en
```

In addition to the *IBM VoiceXML Programmer's Guide*, we recommend the following references:

- Balentine, B., & Morgan, D. M. (2001). *How to build a speech recognition application: A style guide for telephony dialogs* (2nd ed.).
- Cohen, M. H., Giangola, J. P., & Baloch, J. (2004). *Voice user interface design*.

## 1.1 Advantages of speech-enabled applications

As with any new technology, speech technology was first used by *early adopters*. We could observe this behavior ten or twenty years ago. After years of research and development, the real advantages are now clearly identified. You can divide the real advantages into two major categories: the user point of view and the service provider point of view.

A recent market study shows that users are satisfied with automatic applications when the automatic applications are better than other services by virtue of saving time or saving money.

When calling customer services, hotlines, or information systems, people are sometimes annoyed when they hear automatic applications. People like the flexibility of speaking with human beings. But customers are interested in getting around-the-clock, 24x7 support. Customers also are interested in calling a less expensive customer service number, especially, if the automatic application fulfills the same needs that speaking with a person fulfills.

In addition to this, for some applications, the caller may have to wait for a long time before reaching a human operator. Automatic applications are always available and ready for use. Once your speech application is written, Information Technologists only have to design the telephony platforms and speech servers to handle all incoming calls.

On the other side, service providers may be interested in speech applications that offer a new service to their customer. The demand may come from the marketing department to convey a positive business image to their existing customers. Offering a new service is also a key factor in the acquisition of new customers.

Speech technologies help you achieve cost reductions. Call centers find the Return On Investment (ROI) for speech often justifies the move to speech-enable their operations.

Some call centers already provide self-service application through Dual Tone Multi-Frequency (DTMF). By speech-enabling an application, you can eliminate or significantly compress confusing and multi-level menus, leading to an average call duration reduction. It is more convenient for cellular phone users to speak and keep their mobile phone close to their ear than press a DTMF key and rush the phone back to their ear before they miss the next prompt.

## 1.2 Human factors

Among the wide range of human factors related to speech applications, we would like to introduce:

- The difficulty in coping with time and its consequences
- The behavior difference between naive and expert users
- One dialog aspect related to system response
- A prompt design issue: optimizing the pause length

For additional information about human factors and their consequences for speech application, we recommend that you read chapter two of the *IBM VoiceXML Programmer's Guide*.

### 1.2.1 Time is crucial

An essential aspect of speech applications compared to graphical applications is time.

A static Web page lets the reader take as much time as needed to read it, understand it, and read it again if necessary.

Application dialogs may let the caller repeat or ask the application to repeat, but we cannot freeze speech.

Thus, developers have to bear this in mind and write dialogs and menus accordingly.

For instance, some people advise you to stick to a maximum number of five menu items when the *barge-in* feature is disabled. You can increase the number of menu items if you enable the barge-in feature.

The *IBM VoiceXML Programmer's Guide* describes appropriate menu and prompt constructions (page 48).

### 1.2.2 Applications have to fit human communication rhythm

The effects of System Response Time are still an area of research and study even if general behaviors are now well understood.

While callers will react knowing they are talking to a computer, the application still has to take into account that its outputs have to be understood by a human being.

Speech application developers have to use a consistent timing when designing the dialog in order to cope with the overall human communication rhythm.

Take the example of a caller asking for forecasts of predefined locations. We notice that the system answer was a bit too quick to reflect a communication between human beings. The back-end query was extremely fast. After the user query, we decide to add a very short musical tone before giving the forecast back to the caller. This simulates the "hold on a second, we are processing your request" that a human operator would say.

### 1.2.3 Optimizing pause length before presenting global navigation commands

It is important for Speech User Interfaces to provide an easy and intuitive method for users to perform tasks and navigate the system. One way to help users navigate is to present global navigation commands at any place in the call flow that could be the end of a task (in other words, a terminal point). Typical global navigation commands are Repeat, Help, Go Back, Start Over, Transfer to Agent, and Exit or Goodbye (*Cohen, Giangola, & Balogh, 2004; IBM, 2000*).

After a user completes a task (such as a banking transaction), speech applications typically prompt the user with a set of menu selection items (for example, to review the transaction or to make another transaction). Since this is a potential terminal point (because the user wants only to make the transaction and then to navigate to a different area of the interface), it is important to provide the navigation commands here. However, if this information plays immediately after the end of the menu selection list, usability problems can occur. Specifically, many of those users who do, in fact, want to make a selection from the menu will feel obligated to wait for the system to quit speaking before they make a selection. Alternatively, the system might interrupt users who have started to speak a desired option, causing them to stop and restart their command, which confuses the users and the speech recognizer (*Balentine & Morgan, 2001*).

To avoid these problems, it is important to provide an ample amount of silence before the presentation of the navigation commands. This allows users to make a selection from the list without interrupting or being interrupted by the system. On the other hand, the period of silence should not be so long that users give up and simply select an item from the list because they think that they have no other options.

Patrick Commarford and James Lewis published a study advising to set the pause to about 2000 ms (*P. Commarford, J. Lewis. Optimizing the Pause Length before Presentation of Global Navigation Commands*).

## 1.3 SUI Design methodology

Chapter two of the *IBM VoiceXML Programmer's Guide* describes the design methodology extensively. In this section, we briefly go through the process, but we highlight a few points.

As you can see in Figure 1-1, a SUI development is an iterative process. You will certainly want to go through some iterations before deploying the application, but you may also want to iterate after deployment based on the real users' traces.

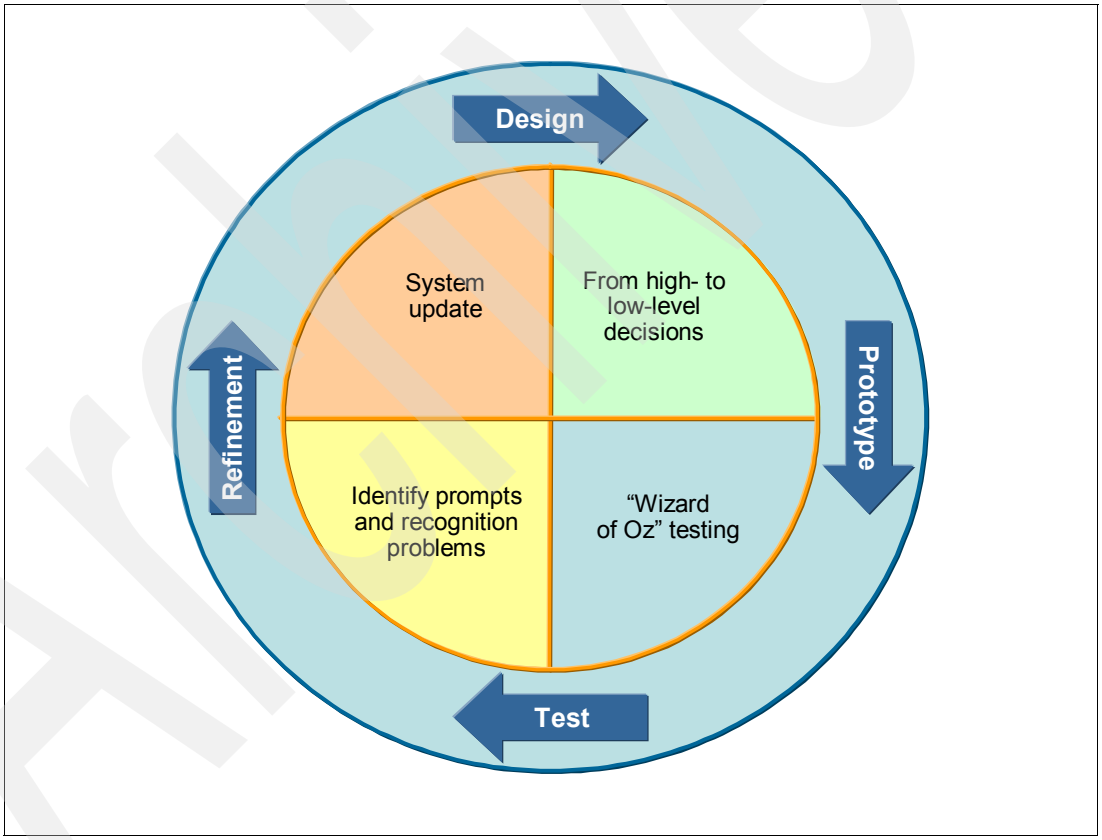



*Figure 1-1 Design methodology: An iterative process*

### 1.3.1 Design phase

This phase is aimed at defining the call flow and creating an initial script for the dialog between the application and the user.

Try to keep it as fluid as possible for as long as possible to pick fruits from the iterative process. You will make high-level to low-level decisions after analyzing your users and tasks.

### 1.3.2 Prototype phase

The use of the technique known as the *Wizard of Oz* testing is helpful especially for your first iteration.

It involves two people: One to play the role of the user and a human *wizard* to play the role of the computer system. They simulate a call from a user to the application based on a scenario (script) written according to the design phase. Wizard of Oz testing helps you fix problems in the script and task flow before committing any of the design to code.

Although it can be seen as a bit theoretical, this technique has shown real benefits in bringing a wider view of potential user requests and expected application reactions. You can also use this technique when adding a new service to an existing application.

### 1.3.3 Test phase

Once you have collected the outputs from your Wizard of Oz testing, you implement your dialog and test it. You ask testers to call the system and you record traces to analyze the results.

During the testing phase, you spot the recognition issues (mis-recognitions) as well as the potential prompt issues (unclear prompts, wrong TTS flow, mispronounced words, and so on).

You may also conduct surveys to get the caller's general feeling about the application or to clarify a specific issue.

### 1.3.4 Refinement phase

During the refinement phase, you modify the code (lexicon, grammars, and prompts) based on the issues spotted during the test phase.

Finally, you go back to the Design Phase and the Prototype Phase to work around a recognition issue or you go straight to the test phase and see if your changes now lead to better results.

## 1.4 Best practices

In this section, we outline a set of best practices to design dialogs, grammars, and prompts.

Because the dialog consists of prompts and grammars, and because you can identify prompts as the "computer voice" and you can indentify grammars as the "computer ears", they all are linked together. Changing one of these three can lead to changes in the others.

### 1.4.1 Prompt design

Reference guidelines for prompt design can be found in chapter two of the *IBM VoiceXML Programmer's Guide*. Timing guidelines are proposed in the section, "Selecting recorded prompts or synthesized speech".

Prompts are the most "visible" part of the application.

Users first judge the application based on the prompts. Therefore, for high quality speech applications, we advise the use of prerecorded professional speakers whenever possible.

But the use of TTS is mandatory for dynamic content. Playback prompts then wrap the TTS utterance. The choice of the amount of speech which needs to be synthesized (only the dynamic content or the entire sentence) highly depends on the application. This needs to be decided according to the testing results.

You can tune your TTS prompts by using Speech Synthesis Markup Language (SSML) to provide additional information in the input text. For a complete description of the TTS and SSML capabilities, refer to *Using IBM Text to Speech Technology and Speech Synthesis Markup Language:*

http://www.ibm.com/support/docview.wss?uid=swg27005819&aid=1

Different prompt designs can lead to different user responses as shown in Example 1-1 and Example 1-2.

*Example 1-1 System prompt with the potential for different user responses*

```
System: Would you like a sandwich?
User: Yes, I would.
```

*Example 1-2 System prompt to guide the user to more concise responses*

```
System: Do you want a sandwich? Say yes or no.
User: Yes.
```

In Example 1-2, we closed the question to avoid answers like “Yes, I would”, “Yes, please” and so on.

This could be an efficient trick to overcome recognition problems. You may want to modify your prompts to guide the user to choose specific commands and avoid acoustic *confusabilities*. On the other hand, you might want to let users work with the application in a more natural way, enhancing the grammar to accept the most common input variants.

## 1.4.2 Dialog design

### Consistency

As described in the *IBM VoiceXML Programmer’s Guide*, you should try to keep as much consistency as possible within the application. Users master applications more quickly when they can predict what responses are valid. For example, if you allow positive responses of “Yes”, “Okay”, and “True” to one yes or no question, you should allow the responses for all yes or no questions in your application.

Moreover, standards tend to appear for the choice of some commands. The global navigation commands like Repeat, Help, Go Back, Start Over, Transfer to Agent, and Exit or Goodbye, are widely accepted (*Cohen, Giangola, & Balogh, 2004; IBM, 2000*). Try to use aforementioned global navigation commands instead of synonyms. As callers become more and more trained to Speech User Interfaces, they instinctively think about them.

You can improve the consistency by reusing existing dialog components. See Chapter 8, “Reusable Dialog Components” on page 89 for detailed explanations about this topic.

### A means to handle tough tasks

The dialog construction should follow the SUI iterative process. It needs to be fluid during the first iterations to evolve to a more rigid form.

Along these iterations, developers may have to modify it to go over a speech recognition issue.

One of the most popular examples is the recognition of spelling sequences. It is known as being one of the hardest tasks for speech applications. Example 1-3 shows a dialog flow which can handle such a difficult task.

*Example 1-3 Dialog flow for the recognition of spelling sequences*

```
System: Part Code?

User: S X H U F
System: S X H U S?
User: No
System: S as in Sally?
User: Yes
System: X as in X-ray?
User: Yes
System: H as in ham?
User: Yes
System: U as in umbrella?
User: Yes
System: S as in Sally?
User: No
System: F as Franck?
User: Yes
System: S X H U F?
User: Yes
```

Other dialog methods making use of the n-best list are described in the *IBM VoiceXML Programmer's Guide*.

### Barge-in

When going to more detailed aspects, the speech application developers will have to make choices such as the use of the barge-in technique. In most cases, we will choose to enable barge-in. If so, then we have to decide which barge-in technique we want to use. WebSphere Voice Server V5.1.x for Multiplatforms currently supports two barge-in detection methods.

The two methods are known as the *speech* detection and the *hotword* detection.

The speech detection was significantly improved from V4.2 to V5.1.x. The new algorithm approach called the feature-based speech detector leads to nice robustness. Therefore, we strongly invite developers to try this enhanced method.

Chapter 9, "Tuning voice applications" on page 107 provides guidance. For additional information, see *Migrating Applications from IBM WebSphere Voice Server 4.2 to 5.1.x for Use with IBM WebSphere Voice Response*:

http://www.ibm.com/support/docview.wss?rs=761&context=SSKNG6&dc=DA420&dc=DA410&dc=DA440&dc=DA430&uid=swg27006260&loc=en_US&cs=utf-8&lang=en

## 1.4.3 Grammar design

### Avoid unknown pronunciations

Although WebSphere Voice Server has an automatic spelling to sound system, we recommend avoiding unknown pronunciations as much as possible. Indeed some unknown

pronunciations can lead to unexpected phonetic sequences. Typical examples are acronyms which you may utter as full words or letter by letter.

We will see in Chapter 6, “Editor and Pronunciation Builder for lexicons” on page 57 how to use the Voice Toolkit to check that we do not have any unknown pronunciations.

## Avoid acoustic confusability

When writing the application, developers can ease the recognition engine by choosing words which will not introduce any strong acoustic *confusability*. Imagine a grammar containing “cancel” and “parcel”. Since these words can be confused, we can modify the grammar replacing “cancel” by “cancellation”. As we saw in “Prompt design” on page 5, it is up to the prompt to guide the user to what they are expected to say.

## Grammar coverage

The grammar design highly depends on the task we have to achieve. In an ideal situation, finite state grammars should cover all valid entries and only these entries. This is certainly the way to get the best recognition results (We will see in Chapter 9, “Tuning voice applications” on page 107, how to get additional improvements by adding weights on a finite state grammar node.).

Although we will always try to reach the strict covering of valid entries, this is not always possible. The following examples illustrate this issue.

### *Overgeneration*

In addition to the valid entries, grammars can even cover invalid entries because of the ease of writing them this way and the almost impossible way of writing a grammar which would only cover valid entries. We say that these grammars *overgenerate*. A back-end process can filter the recognition outputs and only keep the valid outputs.

Credit card numbers are a good example. The grammar used for recognizing credit card numbers covers invalid credit card numbers, but we use a checksum algorithm to only select valid entries from the n-best list.

### *Case of the impossibility to cover all entries*

For specific grammars, valid entries are hard to guess because of the almost infinite possible entries. Imagine a directory assistance application where the application prompts you for the name of a restaurant. The complete restaurant name is *Pizza San Antonio Franklin Street,* but you may call it *Pizza San Antonio*, *San Antonio Pizza,* or *San Antonio Franklin Street*. It is almost impossible to cover all entries.

2

# Downloading and installing Rational Application Developer and VTK

The IBM WebSphere Voice Toolkit V6.0.1 (VTK) is a plug-in to any of the following Rational Software Development V6.0 Platforms:

- Rational Web Developer
- Rational Application Developer
- Rational Software Architect

For the purposes of this Redpaper, we selected Rational Application Developer.

This chapter provides the instructions for downloading and installing the Rational Application Developer Platform and the IBM WebSphere Voice Toolkit (VTK) from IBM PartnerWorld® Software Access Catalog. We describe:

- Obtaining access to IBM PartnerWorld
- Downloading Rational Application Developer V6.0
- Downloading the IBM WebSphere Voice Toolkit V6.0.1
- Installing Rational Application Developer V6.0
- Updating Rational Application Developer V6.0 to V6.0.1
- Installing the IBM WebSphere Voice Toolkit V6.0.1

**Note:** You can also download IBM WebSphere Voice Toolkit V6.0.1 from the following IBM Web site.

`http://www.ibm.com/software/pervasive/voice_toolkit`

If you already have one of the Rational Software Development V6.0 Platforms, you do not need access to IBM PartnerWorld. Just use the above link and click the **Download WebSphere Voice Toolkit** image under Highlights. On the next page, click **Tool: IBM WebSphere Voice Toolkit** (Version: V6.0.1). For the purposes of this Redpaper, we use the IBM PartnerWorld Software Access Catalog for all downloads.

## 2.1 Obtaining access to IBM PartnerWorld

PartnerWorld is a worldwide program for IBM Business Partners that offers sales and marketing tools, skill-building courses and technical support to help create opportunities to grow your business and drive increased profit. Partners can self-register with PartnerWorld to obtain access to IBM product information. Partners can also purchase additional entitlements, such as downloading IBM software products through the IBM Software Access Catalog.

To become a member of PartnerWorld, visit the following Web site:

http://www.ibm.com/partnerworld/pwhome.nsf/weblook/index.html

Then, click **Join now** under Become an IBM Business Partner.

On the next page, you must click **Independent Software Vendors** (ISVs) to be able to purchase the IBM Software Access Option.

### 2.1.1 Purchasing the IBM Software Access Option

If your company has an active registration as an ISV in PartnerWorld, the primary company contact can access the ordering system.

Purchasing the IBM Software Access Option provides the ability to use the Software Access Catalog to you and to all active registered ISV contacts in your company. Purchasing the IBM Software Access Option provides a wide variety of most major IBM software products on CD for the cost of media, plus shipping and handling, or at no charge by download.

1. Get access to the PartnerWorld Software Access Catalog at:

   http://www.developer.ibm.com/isv/welcome/guide/value.html

2. Click **Buy Now** under the IBM Software Access Option. Complete the prompts for your PartnerWorld Username and Password.
3. Follow the prompts.

### 2.1.2 IBM Software Access Catalog

The IBM Software Access Catalog includes many versions of IBM products in various languages and operating system platforms. The files are packaged into ZIP, TAR, or self-extracting executable (EXE) images. Multiple images related to a product may also be packed together into e-Assemblies for packaging convenience.

**Note:** Downloadable image files are not ISO formatted images, and therefore cannot be used to create CDs images directly.

## 2.2 Downloading Rational Software Development Platform V6.0

You can download the Rational Software Development Platform V6.0 as follows:

1. Begin the process of downloading Rational Application Developer V6.0 by clicking:

   http://www.developer.ibm.com/welcome/softmall.html

2. Click **Log in to Software Access Catalog** (in the middle of the page).
3. Enter your PartnerWorld userid and password if requested.
4. Click **Yes** if you receive any security warnings.

5. Read the IBM PartnerWorld Agreement and select **I Agree** at the bottom of the page.
6. Click **Software Access Catalog - Electronic Software Download**.
7. Under the section, Finder options, click **Find by search text**.
8. Leave the default **Download Director** selected under “Select download method”.

   The Download Director provides high-performance, reliable downloads with pause and resume capability. The Download Director allows you to download one or more files at a time. Download Director is the recommended download method for transferring data.

   HyperText Transport Protocol (HTTP) transfer only allows you to download one file at a time, with no pause and resume capability. If you lose connection while downloading and your Web client and proxy server do not support *byte-range serving,* you will need to start a download from the starting point again. IBM provides HTTP transfer for those instances where Download Director transfers are not possible.
9. In the “Find by search text” field type `rational application developer v6.0`
10. Leave the default **All** methods selected.
11. Click **Search**.
12. In the “Search text results” section, click **+** next to Rational Software.
13. In the “Search text results” section, click **+** next to Rational Application Developer V6.0 Windows® Multilingual e-Assembly (CR2FYML).
14. You will see a list of associated images in the e-Assembly. Click the check box for each image listed below:
    - Rational Application Developer V6.0 Windows Part 1 - REQUIRED. Contains core installation files - ESD extractor. Multilingual. (C81CIML)
    - Rational Application Developer V6.0 Windows Part 2 - REQUIRED. Contains core installation files. Multilingual. (C81CJML)
    - Rational Application Developer V6.0 Windows Part 8 - OPTIONAL: Contains WebSphere legacy test environments. Multilingual. (C81CQML)

    **Note:** If you are going to develop applications for the WebSphere Voice Application Access (Portlets) product, download all WebSphere Portal V5.1 images, too. We do not cover this subject in this Redpaper.
15. At the bottom of the window, click **I accept**.
16. Click **Download now.**
17. A status window will appear to show the overall download status of all selected files. The Download Director will place all downloaded files in the same default directory. We recommend that you place the downloaded Rational Application Developer files in a separate directory on your local server.

    **Note:** You can return to this page and select another file while the transfer of a previous file is taking place.

## 2.3 Downloading IBM WebSphere Voice Toolkit V6.0.1

The following steps assume you are still logged into the Software Access Catalog and have just completed the Rational Application Developer download. Begin downloading VTK by performing the following actions:

1. Click **Change search text criteria** on the top of the page.
2. In the “Find by search text” field, type `websphere voice server v5.1.3`
3. Leave the default **All** methods selected.
4. Click **Search**.
5. In the “Search text results” section, click **+** next to WebSphere Products.
6. In the “Search text results” section, click **+** next to WebSphere Voice Server V5.1.3 Multiplatforms (CR39XNA).
7. You will see a list of associated images in the eAssembly. Click the check box next to the image listed below:
   - Voice Toolkit V6.0.1 Multiplatform English US (C87LNNA)
8. At the bottom of the window, click **I accept**.
9. Click **Download now**.

Place the downloaded file in a separate directory on your local server.

**Important:** At the time of this publication, IBM Support had just released a V6.0.1.1 fix pack for the IBM WebSphere Voice Toolkit. We strongly recommend that you download this latest version from the following Web site:

`http://www.ibm.com/software/pervasive/voice_toolkit`

1. Click the Download WebSphere Voice Toolkit icon located in the Highlights section on the right-hand side of the Web page.
2. Select **Tool: IBM WebSphere Voice Toolkit**

   `Operating system:` **`Windows XP`** `| Version:` **`6.0.1.1`** `| Release date:` **`21 Apr 2006.`**

This version also requires you to upgrade to Rational Application Developer V6.0.1.1 using the IBM Rational Product Updater from within the Rational Application Developer tool.

Also, note that WebSphere Voice Server V5.1.3.1, that includes cumulative fixes and several new fifth generation voices, is available as a download from the WebSphere Voice Server Support Web site:

`http://www.ibm.com/software/pervasive/voice_server/support`

## 2.4 Installing Rational Application Developer V6.0

Installation of Rational Application Developer V6.0 consists of the following steps:

1. Extract the Voice Toolkit by invoking C87LNNA.exe and read the Readme\readme_install.html file for Rational Application Developer pre-installation tips (such as, to determine if you need to use short path names instead of taking the default installation path name).
2. Build the Rational Application Developer installation disk images by running extractor.exe. Extractor.exe will create several folders including disk1-4.
3. Installing Rational Application Developer consists of two main steps. Begin by invoking launchpad.exe on disk1 or you can launch launchpad.exe as an option when the extractor.exe completes.
   a. Click **Install IBM Rational Application Developer** on the Launchpad wizard.
      i. Follow the prompts.

ii. On the “Select the features for “IBM Rational Application Developer V6.0” you would like to install:” page, you can deselect **IBM WebSphere Application Server V6.0 Integrated Test Environment** if you are not using Reusable Dialog Components (RDCs) in the Voice Toolkit.

iii. On the “Installation of IBM Rational Application Developer V6.0 complete. Read the information below and then choose Finish” page, deselect the **Launch Agent Controller install**.

iv. Click **Finish** when complete.

b. Click **Install WebSphere test environment V5.x** from the Launchpad wizard.

i. Follow the prompts.

ii. On the “Select the features for “IBM Rational 5.x Legacy Integrated Test Environments” you would like to install:” page, select only **WebSphere Application Server 5.1**. This is for use by the Integrated Runtime Environment in the Voice Toolkit.

iii. Click **Finish** when complete**.**

c. Click **Exit** on the Launchpad wizard.

## 2.5 Updating Rational Application Developer V6.0 to V6.0.1

After installation is complete, we will update it to V6.0.1. To begin, perform the following steps:

1. Start Rational Application Developer using **Start** → **All Programs** → **IBM Rational** → **IBM Rational Application Developer V6.0** → **Rational Application Developer**.
2. On the “Select a workspace” prompt, if you used a short path name when installing Rational Application Developer, then use the same short path name for your workspace. For example:

   ```
   C:\RAD6\workspace
   ```

   Otherwise, use the default.
3. To update Rational Application Developer V6.0 to V6.0.1, click **Help** → **Software Updates** → **IBM Rational Product Updater**.
4. Click **Find Updates**.
5. Click **OK** on the Updates Required window. Updates will take a considerable amount of time even when using a fast network connection.
6. Restart Rational Application Developer when completed.

## 2.6 Installing IBM WebSphere Voice Toolkit V6.0.1

Install the Voice Toolkit by following these steps:

1. Extract the Voice Toolkit by invoking C87LNNA.exe, if you have not done this in 2.4, “Installing Rational Application Developer V6.0” on page 12, step 1.
2. Insure Rational Application Developer is not running.
3. Start the IBM WebSphere Voice Toolkit wizard by invoking VoiceTools_Setup.exe.
4. Follow the prompts.

5. On the “Choose the setup type that best suits your needs” page, click **Custom,** then click **Next**.
6. Check **Local Voice Server Environment - Technical Preview**, and then click **Next**.
7. Select your desired language, and click **Next** until the activity bar appears. Languages other than US English will require a download. For the purpose of this Redpaper, we chose US English.

After installation, be sure you update the js.jar file, as specified in the Voice Toolkit readme_install.html.

Also, review readme_vt.html from the Voice Toolkit download for information regarding Getting Started, What’s New, and Known Problems and Software Limitations.

3

# Rational Application Developer and VTK overview and concepts

This chapter provides an overview of the Rational Application Developer Platform and the IBM WebSphere Voice Toolkit (VTK). We also introduce the concepts that this voice application development environment uses.

Rational Application Developer is a premier environment for developing advanced enterprise applications. It provides the tools necessary to create sophisticated e-business applications that handle transactions, security, client connectivity, and database access. The IBM WebSphere Voice Toolkit installs as a plug-in to Rational Application Developer to extend this development environment to voice applications.

## 3.1 Rational Application Developer concepts

Rational Application Developer Platform utilizes many concepts in application development. If you are using this tool for the first time, explore the product Welcome (**Help** → **Welcome**) to learn about major features in the product and relevant technologies, and to receive guidance for performing your first steps. For detailed information in understanding concepts, explore the product Help Contents (**Help** → **Help Contents**).

Here we highlight several of the concepts used in voice application development and analysis.

**Project** A collection of files used in building your application. The first step in creating your application or a file is to create a Project with **File** → **New** → **Project** → **Voice Project.**

**Perspective** Defines the initial set and layout of views in the Workbench window. They provide a set of functionality aimed at accomplishing a specific type of task or working with specific types of resources. Perspective are also associated with the Project type. For example, if you create a Voice Project and are not in the Voice Perspective, you are asked if you would like to change to that perspective. You can always change the Perspective by using **Window** → **Open Perspective** → **Other...**

**View** Views support editors and provide alternative presentations, as well as ways to navigate the information in your Workbench. For example, the Navigator view displays projects and other resources with which you are working.

Views also have their own menus. To open the menu for a view, click the icon at the left end of the view's title bar. Some views also have their own toolbars. The actions represented by buttons on view toolbars only affect the items within that view.

A view can appear by itself, or stacked with other views in a tabbed notebook. You can change the layout of a perspective by opening and closing views and by docking views in different positions in the Workbench window.

**Note:** For the purpose of this Redpaper, we will primarily be using the Voice and Voice Trace Analyzer perspectives. We also assume that you have created a Voice project for placement of your files.

You can break the development of an application down into three parts:

1. The front end, or user interface: How it looks or in the case of a speech application, how it sounds.
2. The organization of the application into parts: modularity and structure.
3. The business, logic, back-end, or *what goes on inside the black box.*

A well designed Web-based application can have the *work* and *structure* separate from the *presentation* to the user, which allows different input or output modalities to utilize the same logic.

The Rational Application Developer Platform provide a rich set of tools to create these application building blocks.

Features of Rational Application Developer include:

- Tools Framework
  - Eclipse
  - Extensibility
- Web Services
  - Java, J2EE, and EJB™ development
  - WebSphere Application Server/Portal
- Process
  - Modeling and Requirements
  - Source Control
  - Reporting
  - Database Connections

## 3.2 VTK components

The IBM WebSphere Voice Toolkit builds on the Eclipse-based tools framework in the Rational Application Developer infrastructure to enable the development of effective voice applications. Voice Toolkit Components include:

- Communications Flow Builder

  The Communication Flow Builder is a Graphics Editor that enables you to create and test communication flow models for voice applications. It provides an easy-to-use graphical interface, allowing you to drag and drop graphical objects to create a communication flow. The Communication Flow Builder also builds your code from the communication flow design and creates the voice application. You can also write VoiceXML code by hand using the built-in VoiceXML Editor.

  Communication Flow Builder includes:

  - A wizard with communication flow files that contain the graphical representation of the communication flow model
  - A Graphics Editor that enables you to create visual communication flow models of your voice applications
  - A code generator module that translates the communication flow model into one of the following:
    - VoiceXML V2.1 code
    - An XML file
    - A VoiceXML JavaServer™ Page (JSP™) file
    - VoiceXML JSP fragment file
- Prompt Manager

  You can use the Prompt Manager to view audio files and prompt text that is associated with your VoiceXML and JSP files. When you generate voice files using the Communication Flow Builder, the Prompt Manager collects the audio information and makes it easily accessible so that you can generate Text to Speech (TTS) and play the resulting audio files.
- VoiceXML Editor

  VoiceXML is a markup language designed for creating audio dialogs that feature:

  - Synthesized speech
  - Digitized audio

- Recognition of spoken and telephone keypad input
- Recording spoken input
- Telephony
- Mixed-initiative conversations

The Voice tools include a VoiceXML Editor to help you develop, test, and validate your VoiceXML files.

- VoiceXML Debugger

  The VoiceXML debugger tool tracks the behavior and state of VoiceXML programs, and pinpoints logic errors in VoiceXML applications. Using the debugger, you can pause the running application, examine the working code, and locate the source of the current bug or bugs you might have missed. The VoiceXML debugger pinpoints logic errors by performing the following functions:

  - Trace or highlight code as you run it.
  - Inspect and set variables.
  - Step through VoiceXML (.vxml) and VoiceXML JavaServer Pages™ (.jsp) applications.
  - Set conditional breakpoints.
  - Simulate DTMF keypad input while running the application.
  - Display the stack.

  Additional information is available in the *IBM VoiceXML V2.0 Programmer's Guide*.

- Call Control eXtensible Markup Language (CCXML) Editor

  CCXML is a markup language designed for creating call flow dialogs that feature synthesized speech, digitized audio, recognition of spoken and telephone keypad input, recording spoken input, telephony, and mixed-initiative conversations. The IBM WebSphere Voice Toolkit includes a CCXML Editor to help you develop, test, and validate your CCXML files.

- Lexicon Editor

  The Pronunciation Lexicon Markup Language is designed to allow open, portable specification of pronunciation information for the speech recognition and speech synthesis engines within voice browsing applications. The language is intended to be easy to use by developers while supporting the accurate specification of pronunciation information for international use.

- Grammar Editor

  In voice applications, *grammars* identify the words and phrases that can be spoken by the user. The *Grammar Editor* is a multi-page editor consisting of a Graphics Editor page and a Source Editor page. It simplifies the development of command-and-control grammars and aids in testing these grammars to enhance robustness and performance.

- Test Grammar on Media Resource Control Protocol (MRCP)

  The Test Grammar on MRCP tool tests the grammars you created in the toolkit, or grammars that exist on a Web server, for example, Speech Recognition Grammar Specification (SRGS) XML or Augmented Backus Naur Form (ABNF). The Voice Toolkit connects via the MRCP open standard to either the WebSphere Voice Server in the Integrated Runtime Environment or an external WebSphere Voice Server. You can use this tool to test grammars using speech, text, or enumeration.

- Pronunciation Builder

  Use the Pronunciation Builder to help create, test, and tune pronunciations of words in your voice application (grammars, VoiceXML files, and so on). You store these pronunciations in lexicon files that are used by the IBM Automatic Speech Recognition (ASR) and Concatenative Text to Speech (CTTS) engines.

- Voice Trace Analyzer

  The Voice Trace Analyzer lets you examine recognition data from an IBM WebSphere Voice Server system. Using the data obtained from the WebSphere Voice Server collection utility, it can read multiple trace.log files to build a comprehensive overview of your system's recognition performance and improve awareness of recognition problems.

- Audio Recorder, Conversion, and Analysis

  This toolkit includes functions for use with audio files. Among these functions are an Audio Recorder, Audio Conversion, and Audio Analysis. Use the audio recorder to create the file extensions shown in Table 3-1.

*Table 3-1 Audio file extensions and meanings*

| Audio File Extension | Meaning |
|---|---|
| AU | Headered, 8 kHz 8-bit mu-law single channel format |
| VOX | Headerless, 8 kHz 8-bit mu-law single channel format |
| WAV | Headered, 16-bit PCM_SIGNED |
| ALW | Headerless, 8 kHz 8-bit a-law single channel format |

The Audio Player can play ULW files, as well as those audio formats mentioned previously.

The Audio Conversion wizard supports the conversion of 8 kHz 16-bit PCM_SIGNED audio format to 8 kHz 8-bit mu-law audio format and 8 kHz 8-bit mu-law audio format to 8 kHz 16-bit PCM_SIGNED audio format. The Audio Conversion wizard makes conversion of multiple files and user selection of file extensions possible.

Use Audio Analysis to determine the quality of live audio or an audio file.

### 3.2.1 Testing the microphone

If this is your first voice project, the Audio Analysis Tool window opens so that you can check your system microphone. Click **Display Script** to open the prepared script to read. Then, click **Start** and recite the script until you hear a tone. The Status message on the window lets you know the quality of your recording.

You can also test the microphone at any time by clicking the microphone icon in the tool bar or via **Run** → **Test Microphone**. Refer to Figure 3-1 on page 20.

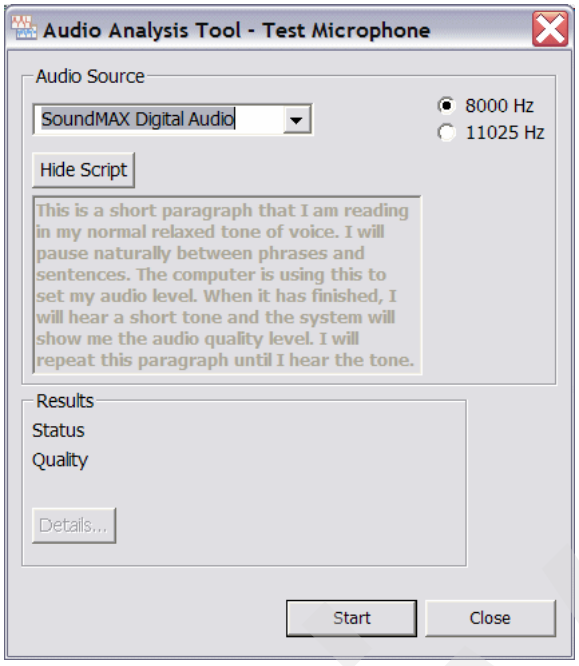



*Figure 3-1 Voice Toolkit Test Microphone dialog*

## 3.3 Integrated runtime environment

Rational Application Developer allows you to install various runtime environments (such as WebSphere Application Server) for locally testing the applications you create. The Voice Toolkit provides an optional *Technical Preview* environment (see Figure 3-2 on page 21) to test your voice applications. Choosing this custom environment during Voice Toolkit installation installs WebSphere Voice Server in WebSphere Application Server V5.1 (that you installed with Rational Application Developer). Plus, it installs a Session Initiation Protocol (SIP)-based VoiceXML Simulator that communicates with the local WebSphere Voice Server via MRCP.

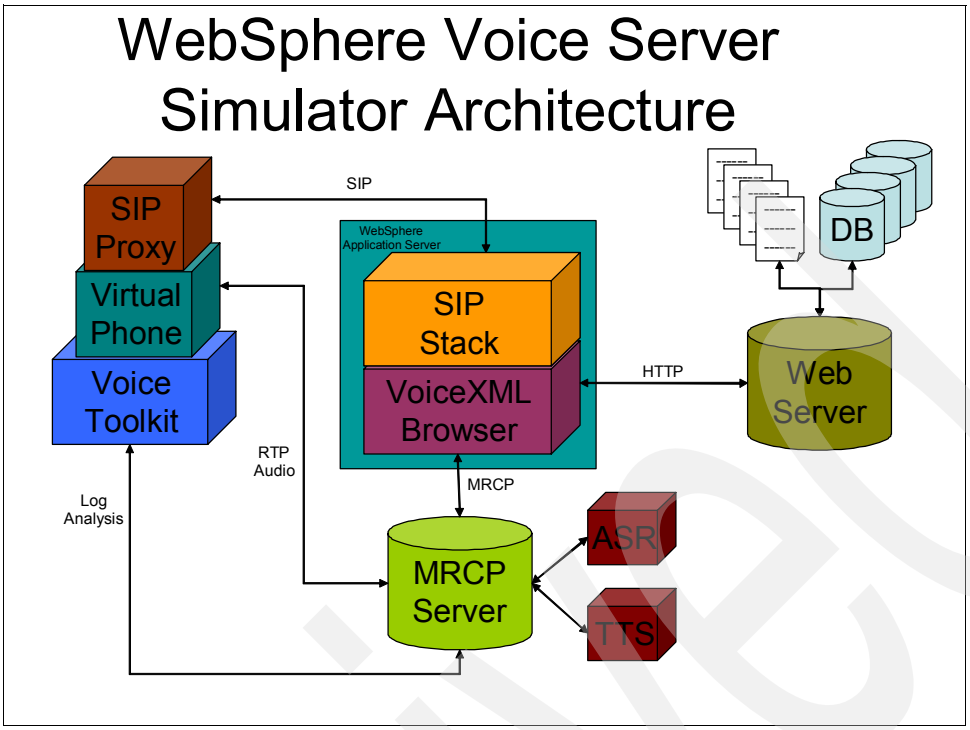



*Figure 3-2 WebSphere Voice Server Simulator environment*

### 3.3.1 Starting the integrated voice server

Testing or debugging your VoiceXML application, compiling grammars from within the Grammar Editor, or testing grammars on MRCP requires the use of a local or remote Voice Server. We chose to use the local Voice Server in the Integrated Runtime Environment, which is the default configuration setting. To start the Integrated Voice Server, select **Run** → **Start Voice Server**. A Progress window displays as shown in Figure 3-3 on page 22.

If you wish to use a remote Voice Server, refer to 3.4.1, “Configuring Toolkit MRCP settings” on page 23 for instructions about how to change the configuration.

**Tip:** The Start Voice Server and Stop Voice Server options are only available in the Voice Perspective.

Optionally, you can start the Voice Server in the Integrated Runtime Environment from a command line window.

*Example 3-1 Starting voice server from a command line window (optional)*

```
cd C:\IBM_Rational_dir\runtimes\base_v51\bin
startServer IBMVoiceServer
```

**Note:** IBM_Rational_dir is the full path to the IBM Rational Software Development Platform. By default, this path is:

```
C:\Program Files\IBM\Rational\SDP\6.0
```

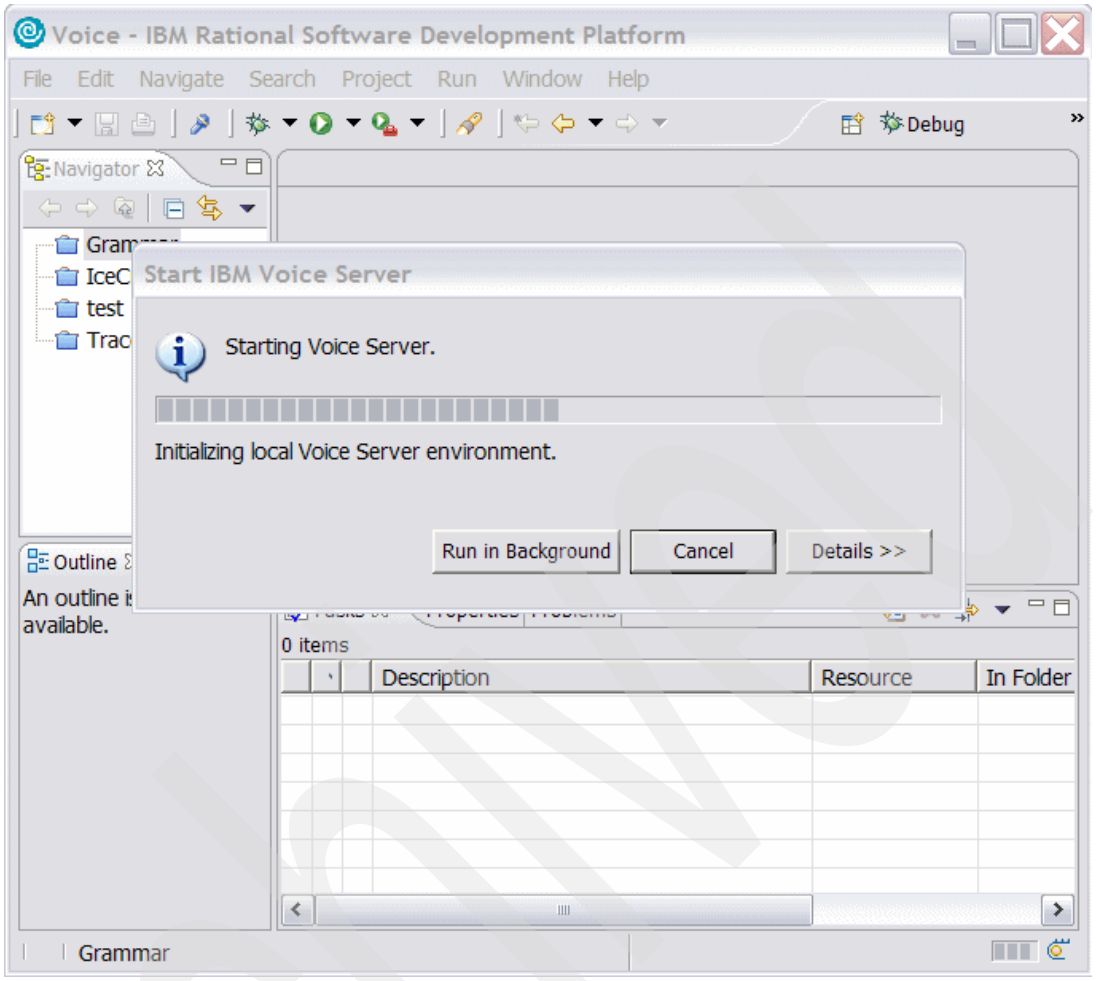



*Figure 3-3 Start IBM Voice Server progress bar*

**Tip:** If the voice server fails to start you can first check for problems by clicking **Details >>** on the progress window. Secondly, you can check for possible errors in the voice server SystemOut.log file:

```
IBM_Rational_dir\runtimes\base_v51\logs\IBMVoiceServer\SystemOut.log
```

**Tip:** If you are unsure if your Voice Server in the Integrated Runtime Environment is actually running, you can check the status from a command prompt by typing the following commands:

```
cd IBM_Rational_dir\runtimes\base_v51\bin
serverStatus IBMVoiceServer
```

## 3.4 VTK preferences

Change preferences or settings for the Voice Toolkit by selecting **Window** → **Preferences,** then click **Voice Tools** as seen in Figure 3-4 on page 23.

### 3.4.1 Configuring Toolkit MRCP settings

Click **Speech Engines,** then click **WebSphere Voice Server V5.1 (MRCP)**.

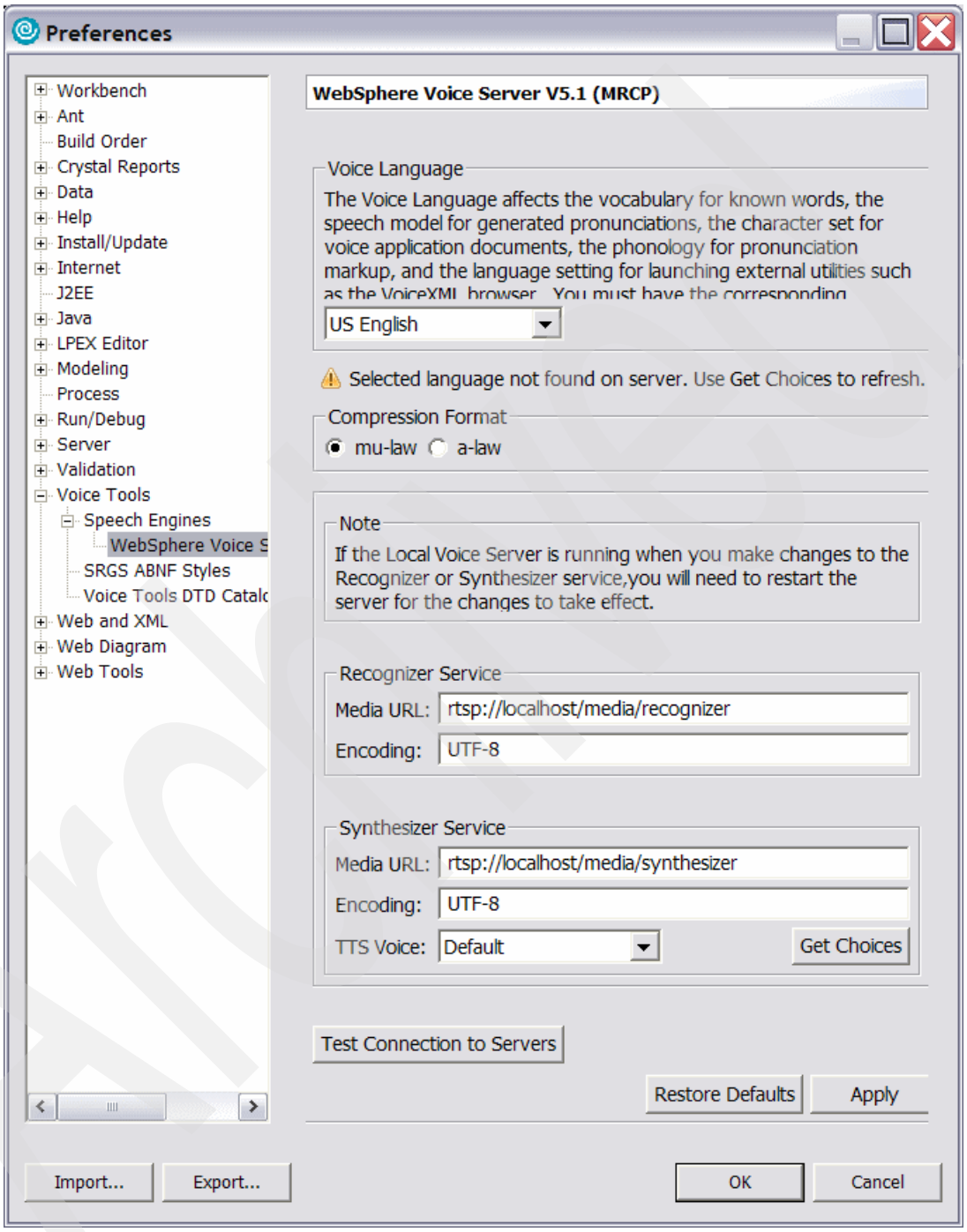



*Figure 3-4 Voice Toolkit MRCP Preferences*

To change to a remote WebSphere Voice Server, change localhost to the IP address or the hostname of your remote server, then click **Apply**. Verify connection to either your Remote or Integrated Voice Server by clicking **Test Connection to Servers**.

### 3.4.2 Configuring SRGS ABNF styles

You can customize the colors of the syntax highlighting for various content types as shown in Figure 3-5.

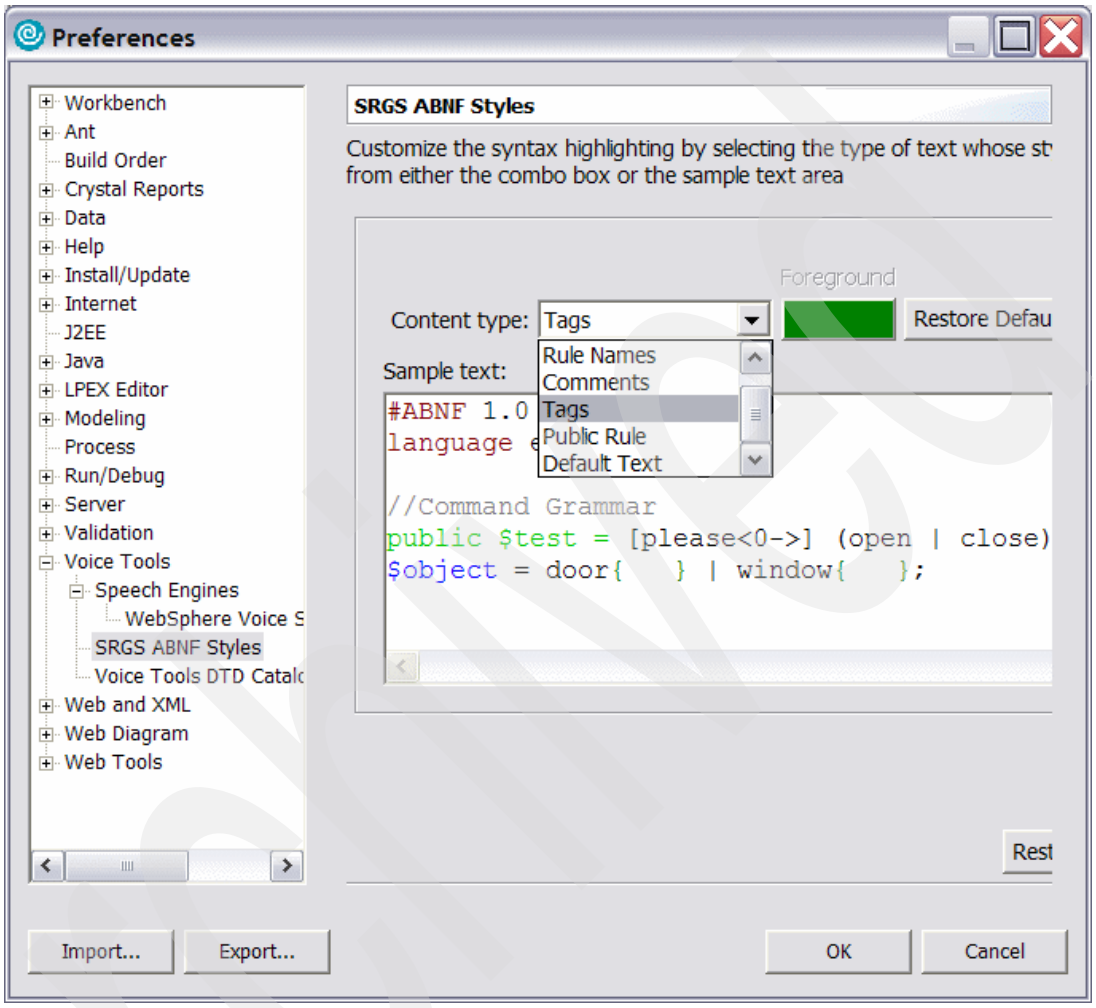



*Figure 3-5 SRGS ABNF Styles*

1. Select the content type from the list or from the sample text area.
2. Click **Foreground** to select a color.
3. Click **OK** to save your changes and close the preferences window, or **Cancel** to discard your changes.

### 3.4.3 Configuring the Voice Toolkit Document Type Definition Catalog

The Voice Tools Document Type Definition (DTD) Catalog (see Figure 3-6 on page 25) contains a list of all known extensible markup languages supported by the Voice Tools. Currently, these are:

- Voice eXtensible Markup Language (VoiceXML)
- Call Control eXtensible Markup Language (CCXML)
- Speech Recognition Grammar eXtensible Markup Language (SRGXML)
- Lexicon eXtensible Markup Language (LXML)

For each of these extensible markup languages, the catalog contains information about the DTD, as well as additional attributes used to generate the initial content for new files generated by the new file wizards.

Voice Tools also adds DTD information to the Eclipse XML catalog. This enables the editors to use this information to control Content Assist and validation.

Refer to the Voice Toolkit help facility by clicking **Help** → **Help Contents** to learn how to add new versions, add new DTDs, restore the default DTD, or set a default DTD.

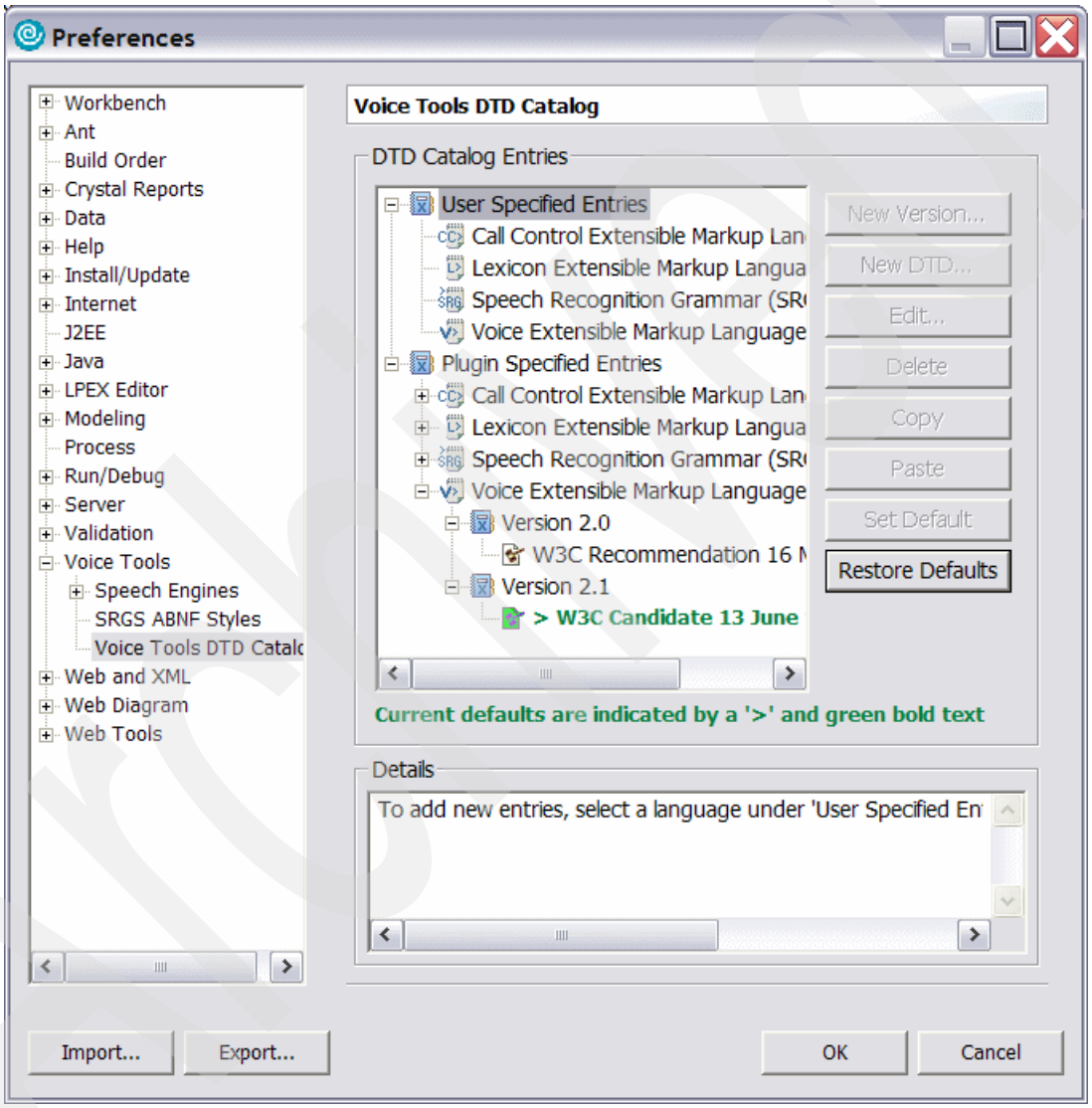



*Figure 3-6 Voice Tools DTD catalog*

## 3.5 VTK help

There are various online and integrated help facilities available for the Voice Toolkit. A good place to start is the Voice Toolkit Information Center, which is available online:

http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/index.jsp?topic=/com.ibm.voicetools.doc/help_home.html

It includes a Getting Started Guide as well as detailed information. These Information Center help facilities are also available within the Rational Application Developer environment by clicking **Help** → **Help Contents**. They are listed in the subsequent left frame under Developing Voice Applications.

Additionally, the Voice Toolkit download package includes additional help:

- Prerequisites and installation

  If you have any questions about hardware and software prerequisites, see the IBM WebSphere Voice Toolkit Installation Readme, readme_install.html, that is located in the *IBM_Rational_dir*\VoiceToolkit\Readme\ directory.

- General Readme

  For additional information about the toolkit features, see the general Readme file, readme_vt.html, that is located in *IBM_Rational_dir*\VoiceToolkit\Readme.

- Sample Readmes

  For additional information about the samples included with the toolkit, see the Readme files located in each of the sample directories in IBM_Rational_dir\VoiceToolkit\Samples.

- Additional Information

  For additional technical information about IBM WebSphere Voice Toolkit, visit WebSphere Studio zone and search for WebSphere Voice Toolkit at:

  `http://www.software.ibm.com/wsdd/zones/studio`

- User forum

  For up-to-date information and technical support, refer to the WebSphere Voice Toolkit User Forum at:

  `news://news.software.ibm.com/ibm.software.websphere.voice-server.voicetoolkit`

## 3.6 VTK samples

The Voice Toolkit includes several sample application files to assist your initial development activities, which we list in Table 3-2.

*Table 3-2 Voice Toolkit sample applications*

| Sample application | Purpose |
|---|---|
| Audio | Lets you verify that your audio input and output devices are working correctly. |
| Communication Flow | Prebuilt samples for use in the Communication Flow Builder. |
| Ice Cream | Represents a typical file written in the VoiceXML Editor, without using the Communication Flow Builder. Plus, a simple SRGS XML grammar that you can reference from the VoiceXML file, or you can import it to familiarize yourself with the grammar test tool. |
| Voice Portlets | A weather and an inventory sample Voice portlet. |

For additional information about the samples included with the toolkit, see the sample Readme files located in each of the sample directories:

*IBM_Rational_dir*\VoiceToolkit\Samples

## 3.7 Additional features

The Rational Application Development Platform and IBM WebSphere Voice Toolkit have many features described in the help facility. We note the following features here since they can significantly assist your application development.

### 3.7.1 Editor Content Assist feature

Content assist helps you insert or finish a tag or function or finish a line of code in a Structured Text Editor. The placement of the cursor in the source file provides the context for the content assist to offer suggestions for completion.

Content Assist is a pop-up window (see Figure 3-7 on page 28) that provides a list of valid tags for the element or attribute at the cursor location.

To open or enable the Content Assist window, place the cursor where you need the help in the Source Editor, following these steps:

1. Hold down the Ctrl key and press the Space bar.
2. To select a tag listed in the pop-up window, use one of the following methods:
   - Double-click the tag.
   - Use the keyboard arrow keys to highlight the item and press the Enter key.

   When you select a tag, the associated beginning and ending tags are inserted into your file at the current cursor location.
3. To close the Content Assist window, click outside the window, or press the Esc key.

#### Changing Content Assist

By default, Content Assist is set to automatically show suggestions as you enter source code. To edit the preference:

1. Select **Window** → **Preferences** → **Web and XML** → **XML Files** → **XML Source**.
2. There is a check in the **Automatically make suggestions** check box. Click it to deselect it, if preferred.
3. Type values in the **Prompt when these characters are inserted** text box to specify how you want to open the Content Assist pop-up window. By default, the < symbol opens the Content Assist window.
4. Click **OK** to save the setting.

**Note:** Here is additional information on Content Assist:

- The DOCTYPE declaration is necessary to enable Content Assist.
- The Content Assist window opens only when the system can provide some assistance. If the cursor location does not have Content Assist, you either hear a tone or see a message at the bottom of the workbench. The Content Assist window does not open if the cursor is at a location for which you cannot add any valid tags, such as when the cursor is inside an invalid tag or all available attributes have been added. Also, make sure there is a space before the cursor to guarantee that tags and attributes can be added.
- The Content Assist feature modifies code based on the current cursor location. If the cursor is placed so that it appears at the first character of an attribute, it replaces the attribute with the Content Assist selection; it does not replace the value of the attribute.

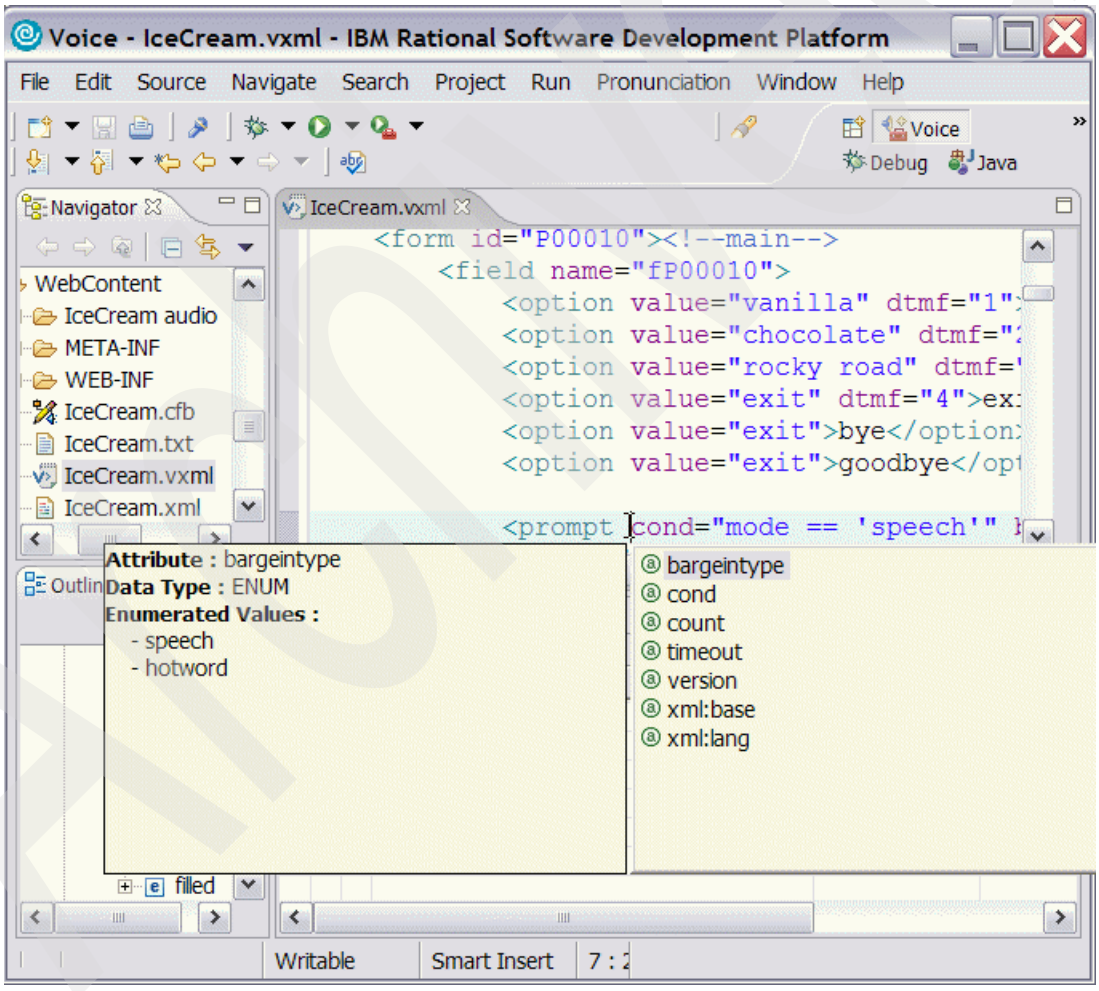



*Figure 3-7 Content Assist in a VoiceXML Editor*

## 3.7.2 File validation through the Problems view

The editor automatically checks the validity of the document during a save operation. Build problems display in the Problems view (see Figure 3-8 on page 29) and the editor annotates

the problems in the vertical ruler of your source code. Other validation of VoiceXML documents can occur when adding elements through the Outline or Design View. The Problems view lists the errors. To clear the list, correct the errors and save the .vxml file.

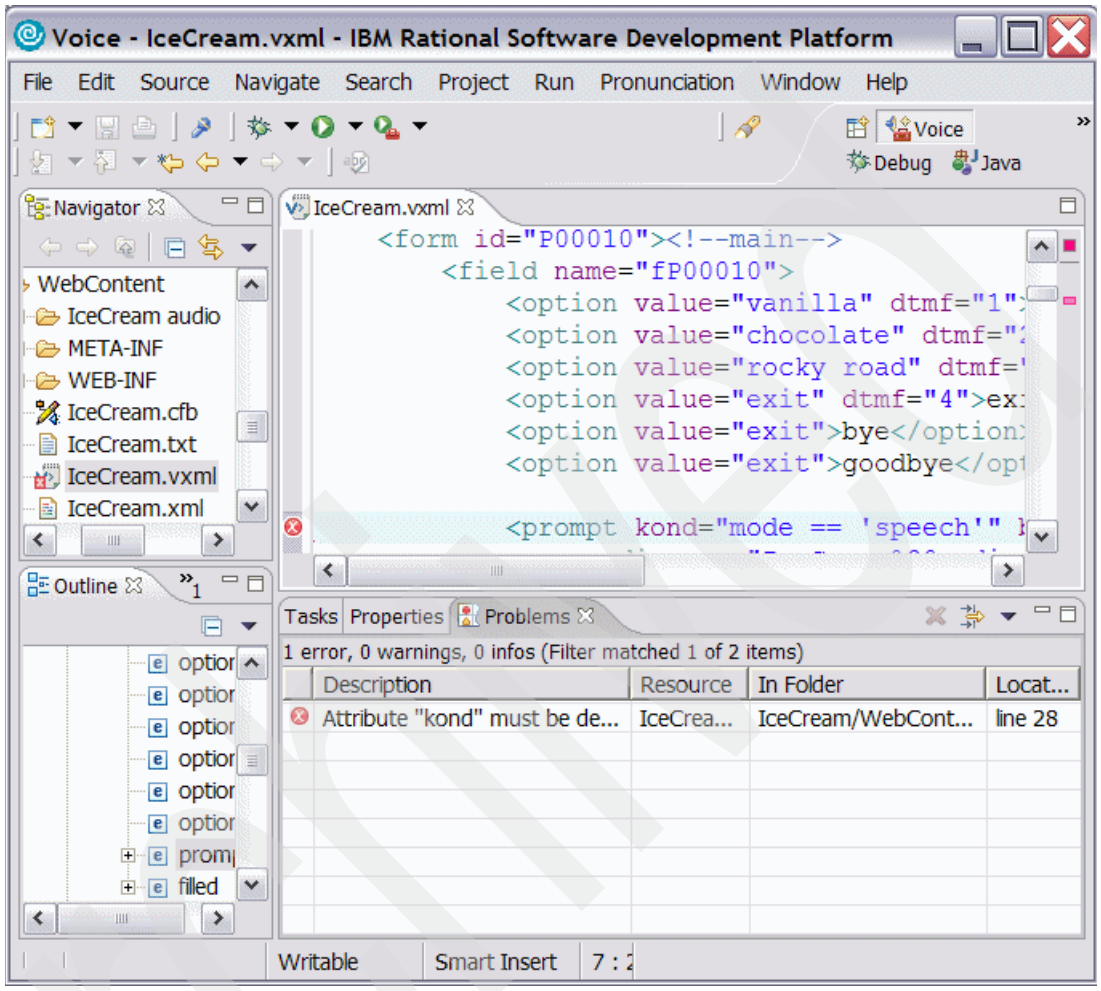



*Figure 3-8 Problems View in VoiceXML Editor*

4

# Grammar Editor

The Grammar Editor is a multi-page editor consisting of a Graphics Editor page and a Source Editor page. It simplifies the development of command-and-control grammars and aids in testing these grammars to enhance robustness and performance.

The Grammar Editor supports the Speech Recognition Grammar Specification (SRGS) that was developed as a W3C standard. It is the preferred markup for grammar syntax used in speech recognition. SRGS has two forms: ABNF and XML. The ABNF is an augmented Backus-Naur Form (BNF) grammar format, which has been modeled after Java Speech Grammar Format (JSGF), which is owned by Sun™ Microsystems™, Inc. The XML syntax uses XML elements to represent the grammar constructs. The SRGS-ABNF Grammar Editor is a Source Editor, while the SRGS-XML Editor is a multi-page editor that has a Graphics and a Source Editor page.

This chapter discusses features of the editor, including:

- Graphics and Source Editor options.
- How to identify and handle unknown pronunciations.
- Supported grammar formats and conversions.

**Tip:** While working with grammars, if you receive a message, `Unable to connect to the grammar compiler`, in the Tasks View, be sure you start the Voice Server (either local Integrated Runtime Environment or remote Voice Server). Do this by clicking **Run** → **Start Voice Server** from within the Voice perspective. Grammars are compiled in the Voice Server when you create or save the file in the Grammar Editor.

## 4.1 SRGS XML Grammar Builder

Creating a grammar file is a relatively easy process. The Grammar Editor gives you the option of using an existing DTD or creating the file from scratch. You can select from multiple DTD files in your installation. See 3.4.3, “Configuring the Voice Toolkit Document Type Definition Catalog” on page 24 for information about how to add and customize your DTD files.

Using an existing DTD gives you the additional option of building grammars from a word list or word file. For our example, we will build a grammar from a file of wine types. This will also

help us show the capabilities of the Unknown Pronunciations view since some wine types are from global regions and not necessarily in the US English lexicon.

To create a new file while in the Voice perspective:

1. Click **File** → **New** → **SRGS XML Grammar File** (see Figure 4-1).
2. Leave the default option “Create GRXML file from a DTD file” selected, then click **Next.**
3. Enter the parent folder in your projects that are open and the file name. Note that the file must be in the WebContent folder of our Grammar project. Click **Next.**

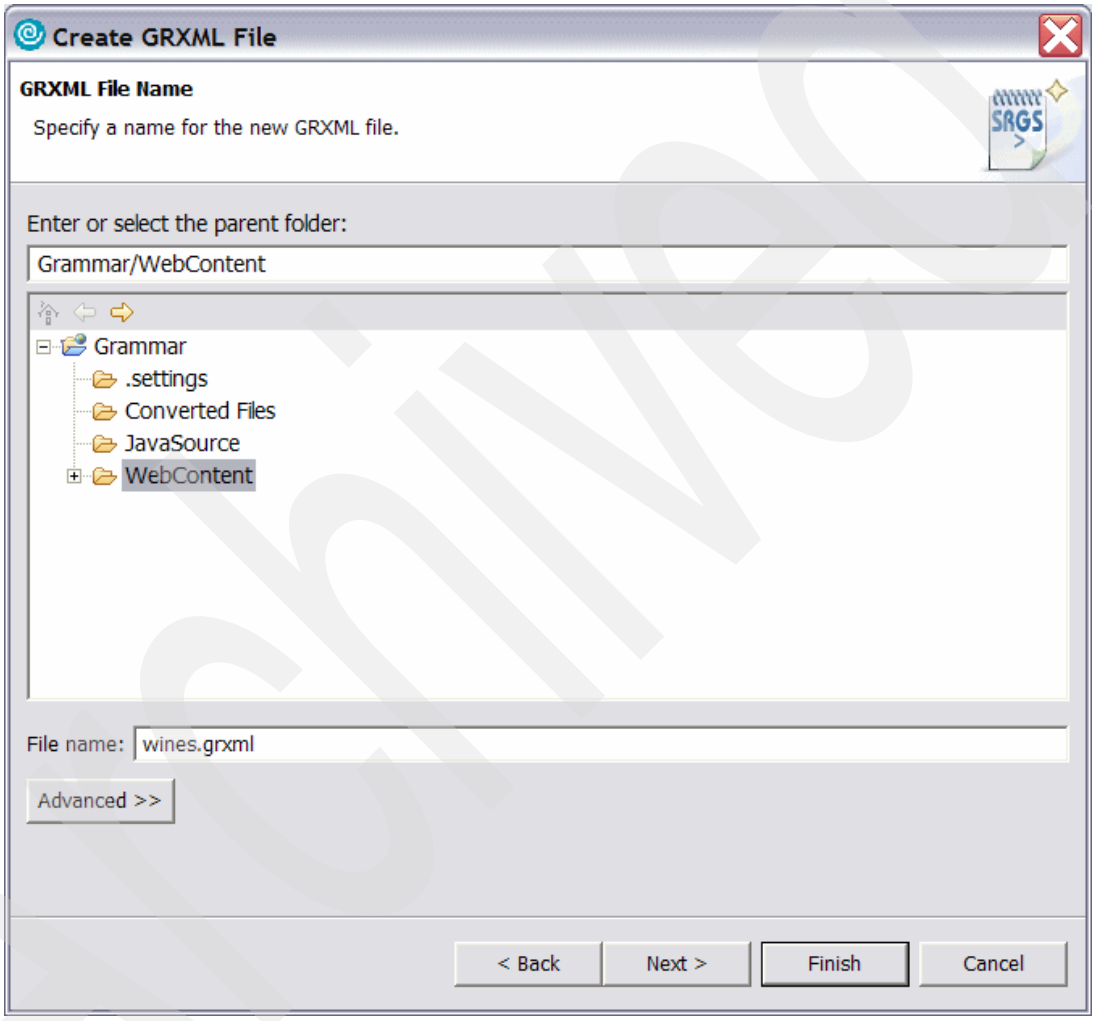



*Figure 4-1 Grammar Editor Create File*

4. You can use the highlighted default DTD or select a different DTD. Refer to Figure 4-2. If you wish to modify your DTD catalog, click **Voice Tools DTD Catalog**. For our example, we chose the default DTD. Click **Next.**

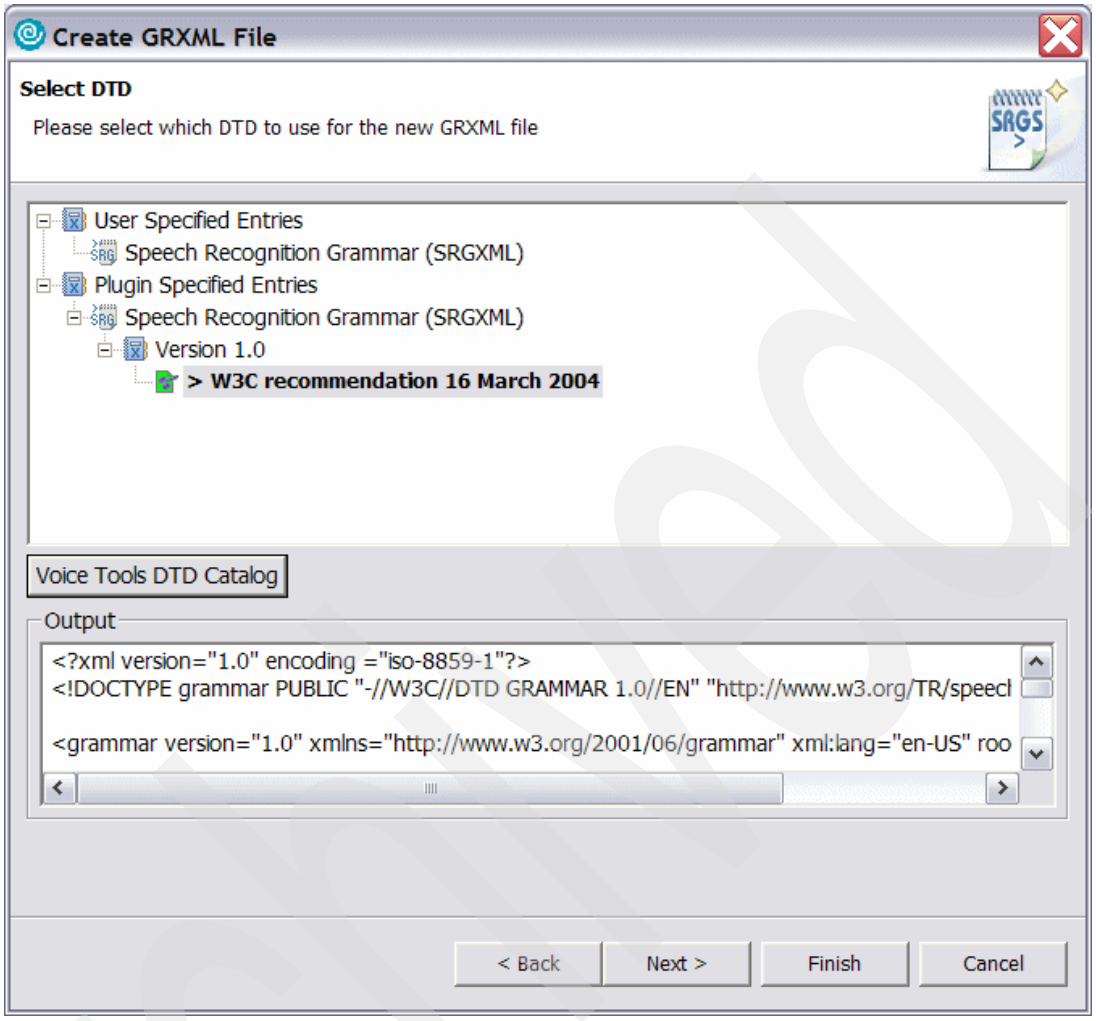



*Figure 4-2 Grammar Editor Create Select DTD*

5. You now have the option to build a rule from an existing list of items. See Figure 4-3 on page 34. This is especially useful for many items. The available options are either to specify a Text File or cut and paste into a Word List field. When complete, click **Finish**.

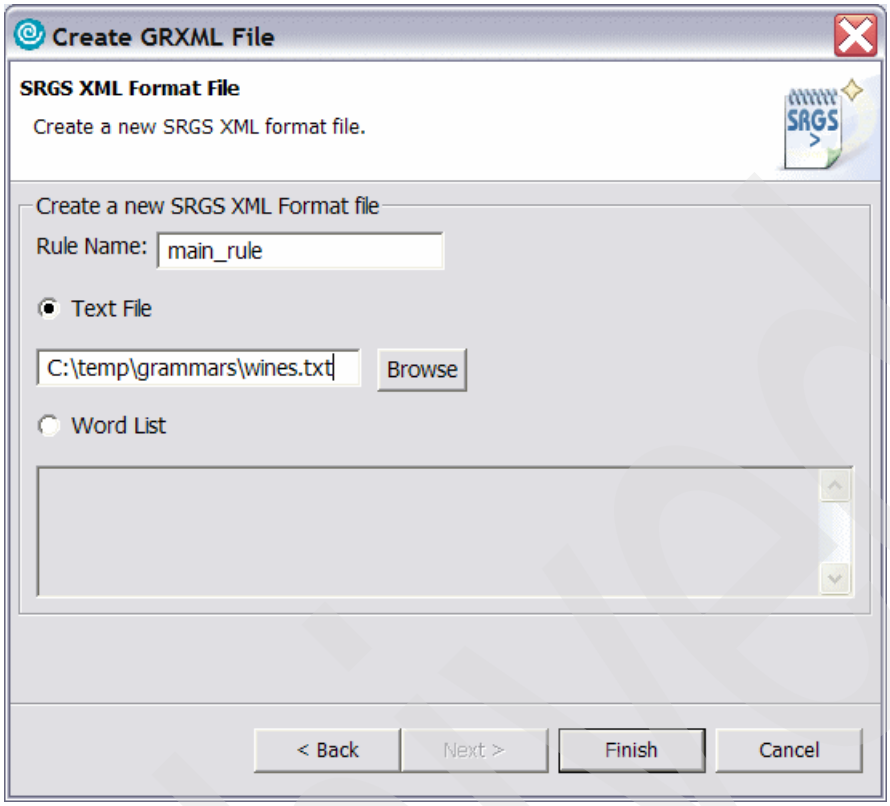



*Figure 4-3 Grammar Editor Create Format File*

Figure 4-4 on page 35 shows the resulting new file.

The SRGS-XML Editor includes two main development environments: a (default) Graphics page and a Source page. You can easily identify each editor by associated tabs on the main window. These tabs allow you to toggle between views and work in either environment. When you make a change on one page, the change is reflected on the other page. For example, if you add a Rule object to the Graphics page, the SRGS-XML source code is automatically generated and displayed on the Source page.

The SRGS-XML Editor also includes the following views:

- Outline

  Provides a zoomed-out view of the workspace to aid in quickly navigating to rules on the canvas.

- Unknown Pronunciations

  The toolkit cannot determine these pronunciations. See 4.5, “Unknown pronunciations view” on page 40 for further details. In Figure 4-4 on page 35, The Outline view currently hides the view that would normally show on a full screen image. Clicking >> (Show List) will allow you to select any hidden view. The number next to the >> icon signifies the number of hidden views.

- Properties

  Allows you to change the properties of an object. Note that the properties reflect the object or item currently selected.

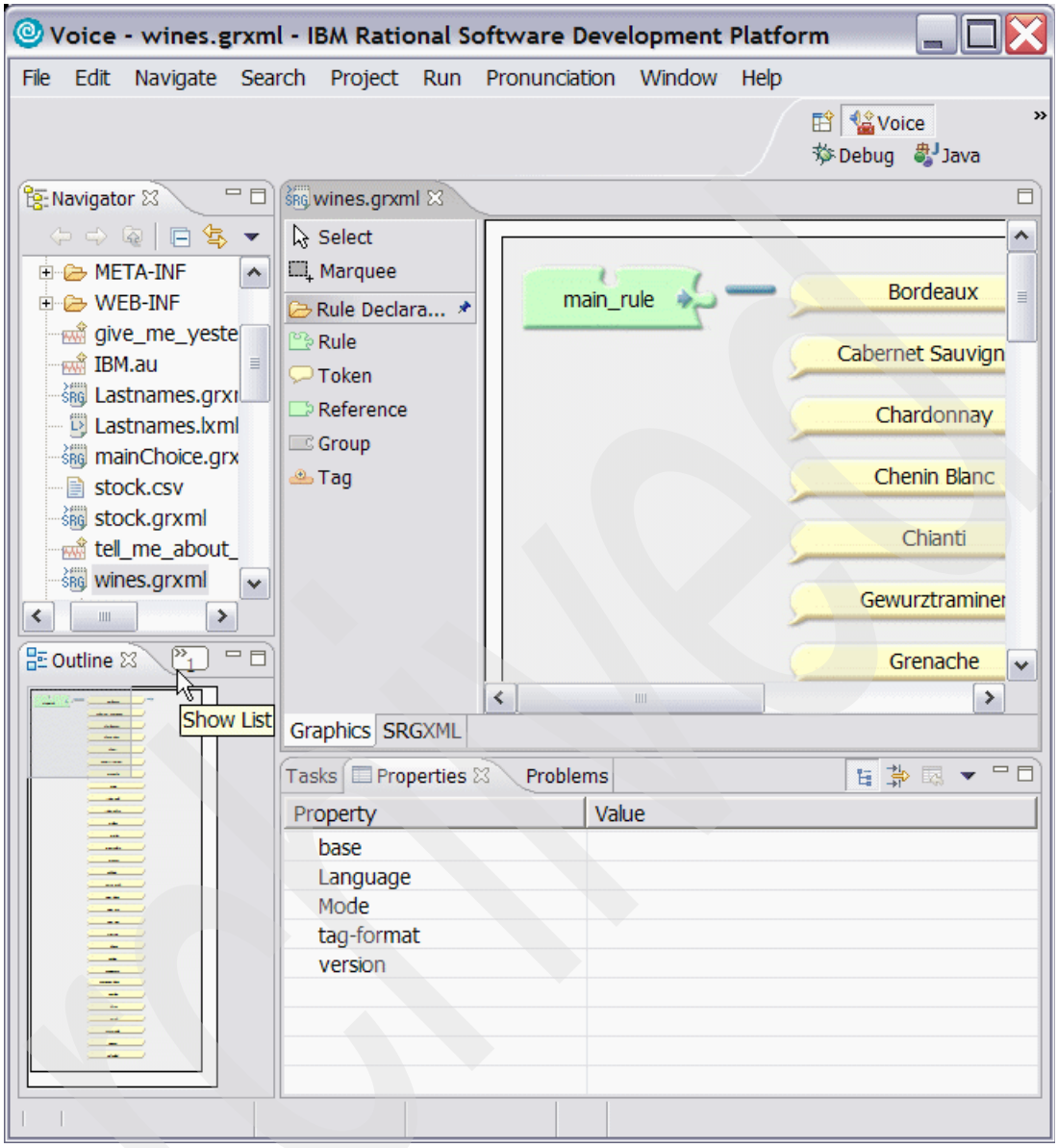



*Figure 4-4 Resulting wines.grxml generated file*

## 4.2 Graphics tab

Use the Graphics Editor to graphically build grammar rules. Select Rule elements from the Object Palette and place them on the Canvas. You can modify element properties in the Property Sheet. The Outline View lists a summary of rules built.

Components of the Graphics Editor page include:

- Canvas

  This is the area on which grammar objects are arranged and linked to represent grammar rules.

- Object Palette

  This is the grammar object you use to build your grammar by dragging and dropping it onto the canvas. You can edit object values directly on the graphical object or edit values in the Properties view. Click **Rule Declaration** to either hide or show the object components. For detailed descriptions of the objects and their usage, refer to the Voice Toolkit help facility in Rational Application Developer by clicking **Help → Help Content,** then by clicking **Developing Voice Applications → Grammar Editor → The Object Palette**.

- Drop targets

  The drop targets and mouse icon feedback refer to the visual aids that the editor gives when dropping grammar elements onto the Canvas. Use them when adding new rules or grammar elements by dragging and dropping them from the palette.

- Outline view

  This is a zoomed-out view of the entire canvas. Selecting an item in this view transitions the editor to that item on the canvas.

- Properties view

  This is for viewing and editing the properties of the elements on the canvas. You can access these properties by right-clicking in any white space on the canvas.

- SISR editing view

  This is for editing SISR. We cover this in detail in 4.2.1, “SISR Editing View” on page 36.

- Pop-up menu

  This displays by right-clicking on a rule element on the canvas. Certain menu options are invalid for some objects. The menu items include:

  – Add Repetition

    Adds repetition to the selected object

  – Edit Comment

    Adds a comment to the selected object

  – Delete

    Deletes the selected object

  – Add Tag

    Adds a Script Tag to the selected object

  – Make Optional

    Changes the *scope* of the selected rule object to optional

### 4.2.1 SISR Editing View

To demonstrate the Semantic Interpretation for Speech Recognition (SISR) Editing View, we used the Stock.grxml example in “stock.grxml” on page 123, since this is a more complex grammar.

Semantic Interpretation for Speech Recognition (SISR) is the programming language adopted by the World Wide Web Consortium (W3C) to define the syntax and semantics of the contents of tags in a grammar. Semantic interpretation tags provide a practical way of handling the information that is presented in the utterance of the user by specifying how to reformat the returns based on rules and tokens that were matched by the speech recognizer.

For example, you want to translate a recognition result into a language-independent format, or reformat dates and numbers into a standard notation.

The Graphics Editor provides an SISR Editing View, which allows you to edit the content of SISR tags. To invoke this view, either click the *tag* object or click **Window** → **Show View** → **Other** → **SISR Editing View**.

You use the SISR Editing View (see Figure 4-5) for editing SISR. Clicking on an element that contains a tag in the Graphics Editor makes its content available in the SISR editing view. This view provides:

- Syntax checking for SISR
- Content Assist for SISR
- Source coloring for SISR

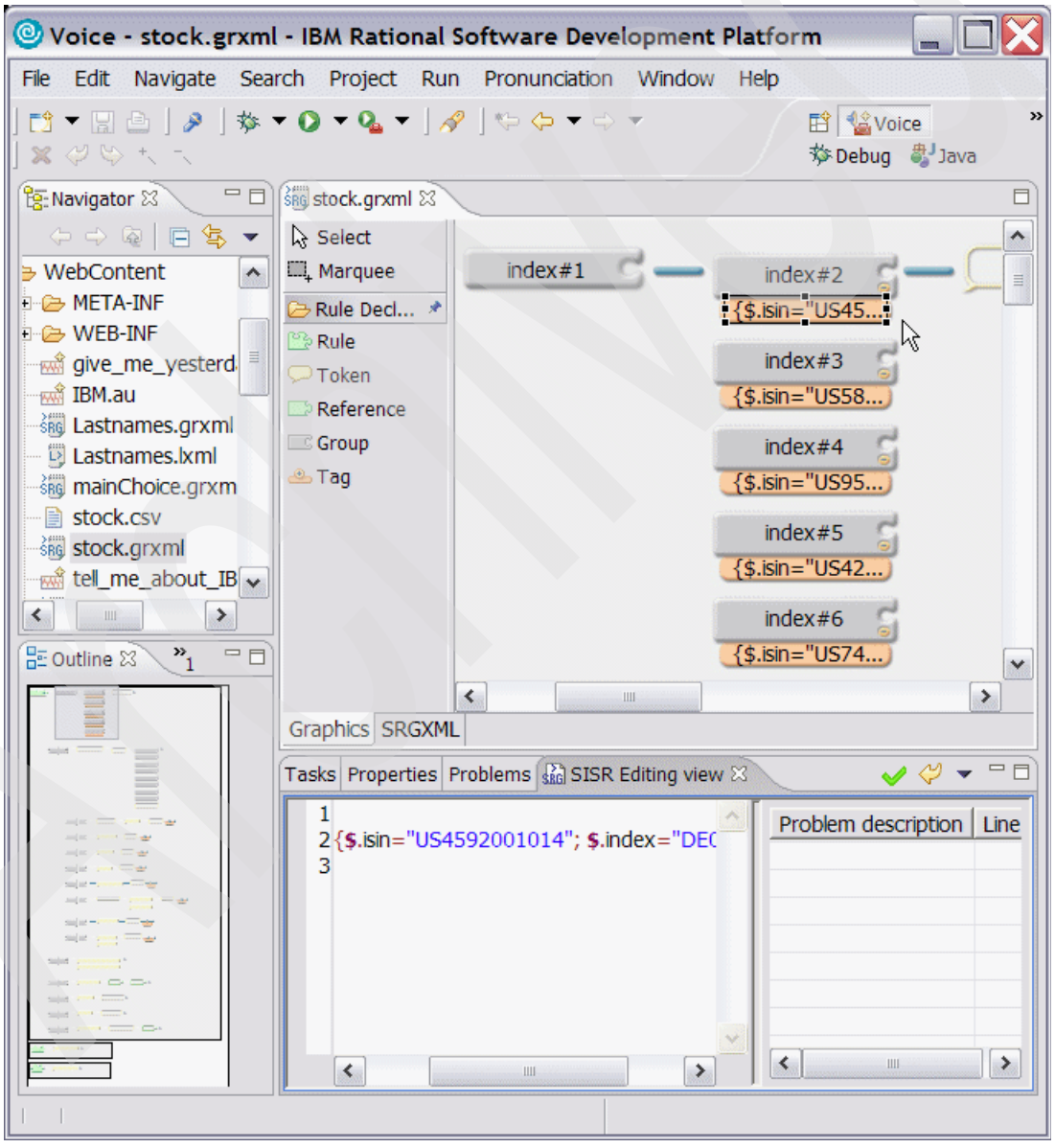



*Figure 4-5 SISR Editing View*

## 4.3 SRGXML tab (Structured Text Editor)

The Source Editor is a fully functional XML-based editor for writing grammars. It validates the SRGS source and reports errors. It contains features for Source Formatting and Content Assistance.

When you make any updates in the SRGS-XML Source Editor, the updates reflect in the Graphics Editor.

The SRGS-XML Source Editor (see Figure 4-6) automatically colorizes the source code using the settings specified on the XML Styles preferences. To change these colors, use the preference page at **Window** → **Preferences** → **Web and XML** → **XML Files** → **XML Styles**.

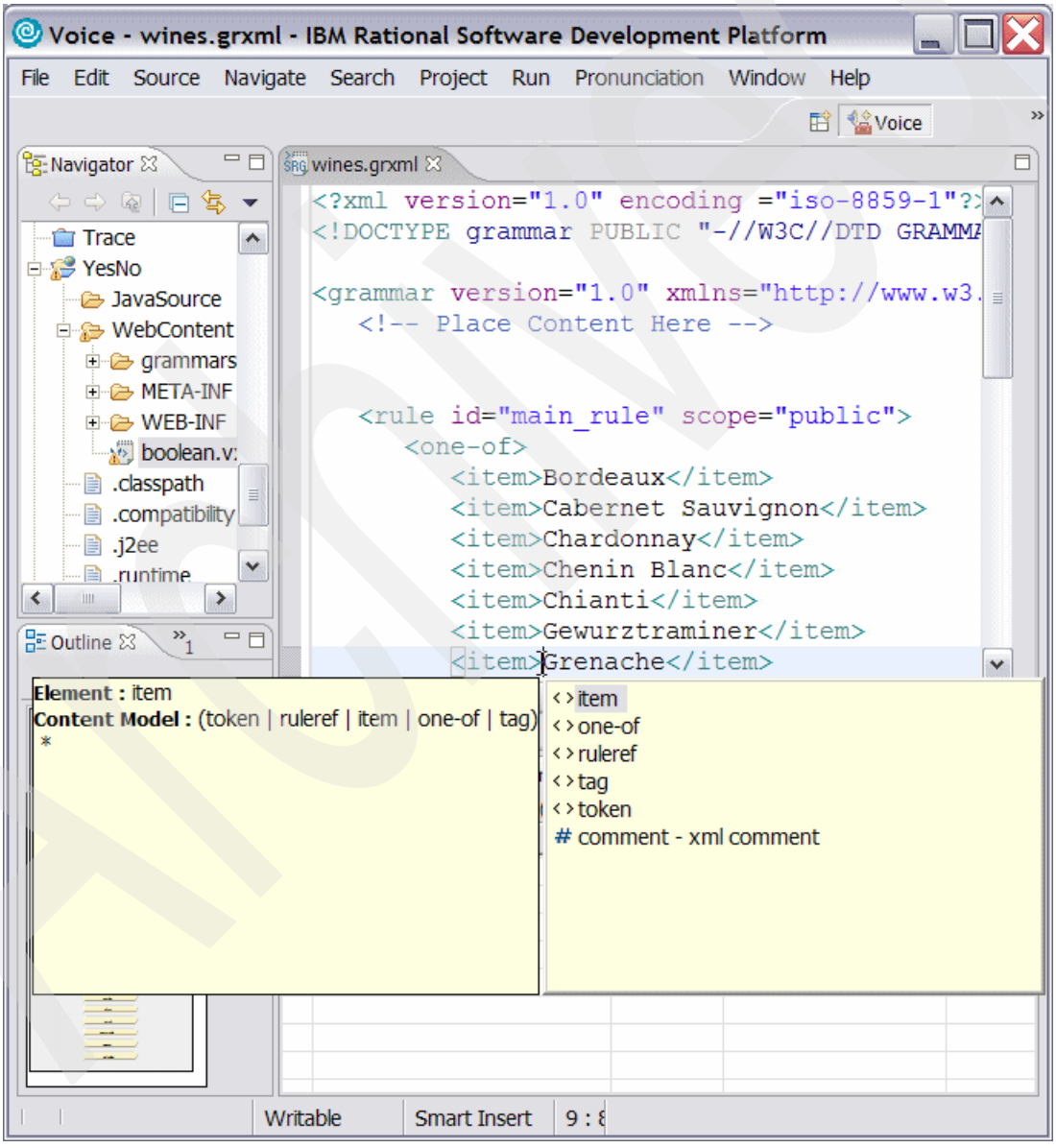



*Figure 4-6 SRGXML Structured Text Editor version of wines.grxml (content assist shown)*

Content Assist is a pop-up window that provides a list of valid tags for the element or attribute at the cursor location. Refer to 3.7.1, “Editor Content Assist feature” on page 27 for information about how to use Content Assist.

## 4.4 SRGS-ABNF Grammar Editor

The SRGS-ABNF Grammar Editor (see Figure 4-7) is a colorizing Source Editor that allows you to create and update SRGS-ABNF grammars. You can also syntax check grammars in the editor.

We used similar steps to those used to create an XML grammar in 4.1, “SRGS XML Grammar Builder” on page 31 to create our ABNF grammar, except we started by clicking **File** → **New** → **SRGS ABNF Grammar File**. Note that you do not get the option to select a DTD file for ABNF grammars.

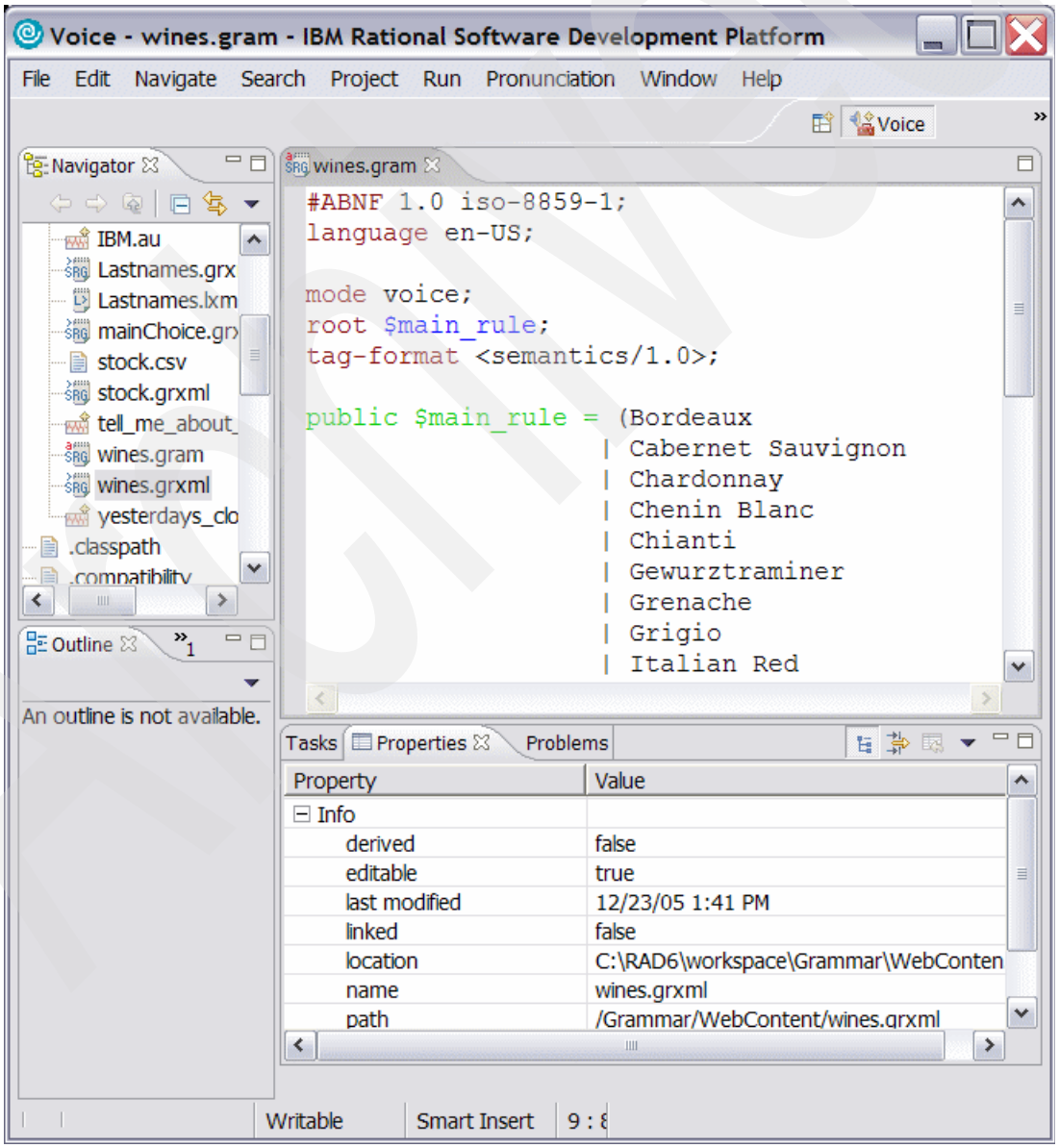



*Figure 4-7 Resulting wines.gram generated file*

## 4.5 Unknown pronunciations view

If you add words, that are not in the vocabulary of the speech engine, to your files, the IBM Speech Recognition engine automatically creates a default pronunciation based on the spelling of the word, but you will probably want to list those words and verify their pronunciations.

**Tip:** To view the possible Unknown Pronunciations, you must open the grammar file in the Grammar Editor and give it focus by clicking the editor window (either the Graphics view or GRXML view in the SGRS XML Editor, or in the ABNF Editor).

Run Verify Pronunciations to update the list, using one of the following methods:

- From the Pronunciation menu, select **Verify Pronunciations** (see Figure 4-8 on page 41).
- Right-click in the Source Editor, and select **Verify Pronunciations**.
- Right-click in the Unknown Pronunciations view, and select **Verify Pronunciations.**

You can play, create, or tune a pronunciation for any word. To do this, right-click each word and select **Play Pronunciation** to hear the TTS pronunciation, or **Compose Pronunciation** to create or tune the pronunciation. Use the **Pronunciation Builder** dialog to create a pronunciation and add it to a pronunciation file. We discuss the Pronunciation Builder in 6.4, “Add words and pronunciations with Pronunciation Builder” on page 61.

**Note:** Make sure your local voice server is running. If it is not, select **Run** from the menu bar, then select **Start Voice Server**. If the voice server is already running, then the Run menu bar option displays `Stop Voice Server`.

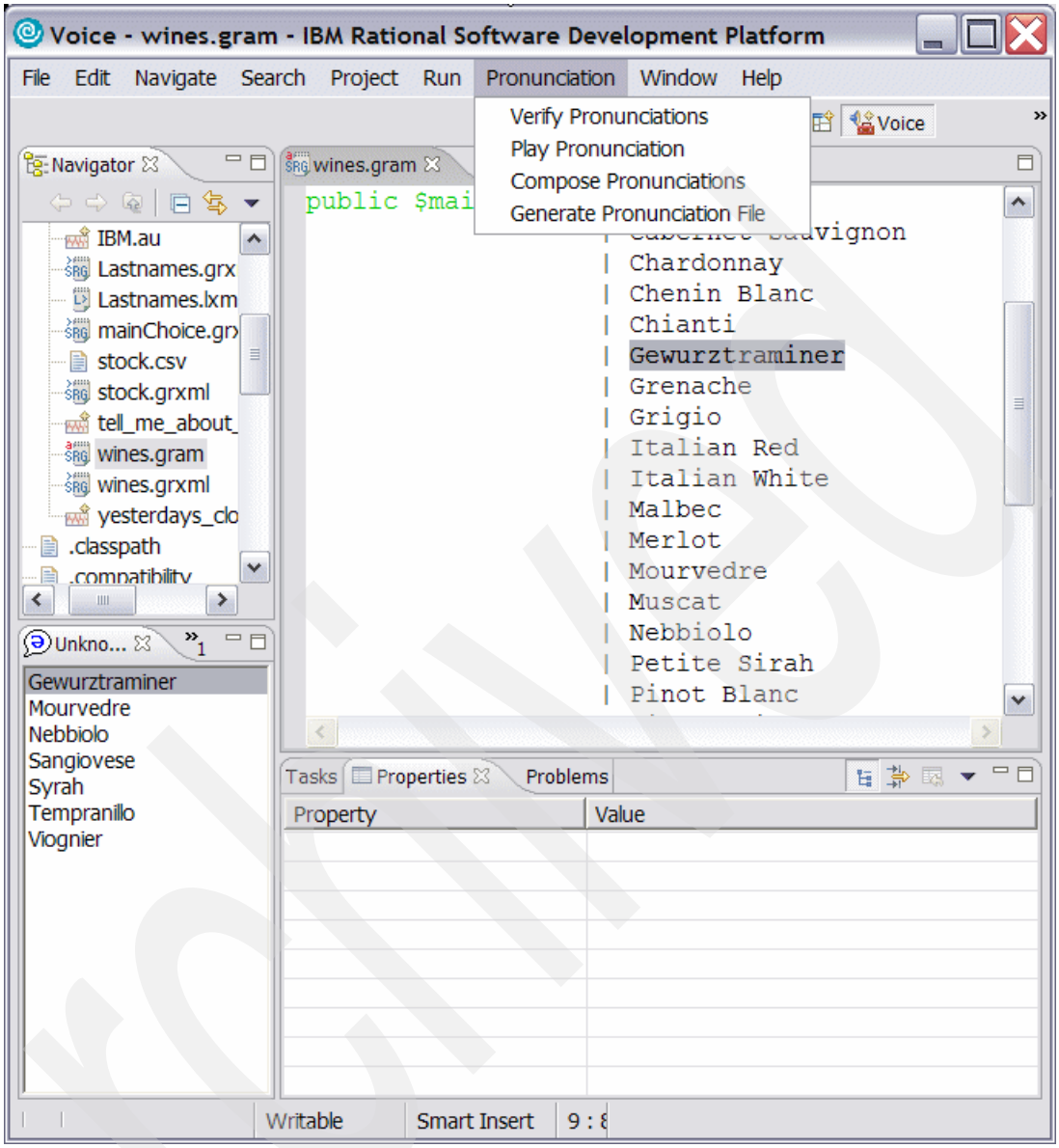



*Figure 4-8 Unknown pronunciations*

## 4.6 Grammar format converters

The Voice Toolkit provides the capability to convert between various grammar types that conform to the Speech Recognition Grammar Specification V1.0. Available conversions are:

- JSGF → SRGS (XML or ABNF)
- BNF → SRGS-XML or SRGS-ABNF
- SRGS-XML → SRGS-ABNF
- SRGS-ABNF → SRGS-XML

You can convert a single file by right-clicking the file in the Navigator view and selecting **Grammar Conversion.** Use the Grammar Conversion Wizard (see Figure 4-9 on page 42) to select one or more grammar files in your project for conversion. To begin:

1. Click **Run** → **Grammar Conversion**.

2. Select the file type and files for conversion.


*Figure 4-9 Grammar Conversion Wizard: Select Files to convert*

3. Select the Grammar Format (ABNF for our example). Click **Next**.
4. The Grammar Conversion Wizard informs you that the resulting files are placed in a *Converted Files* folder in your project. Remember to copy the converted file to the correct location in your project and resolve references to the new file name. Click **Next**.
5. The Status window (see Figure 4-10 on page 43) shows the results of the conversion for each selected file as shown in Table 4-1.

*Table 4-1 Grammar Conversion status window results*

| Status | Meaning |
|---|---|
| Pass (Green) | The conversion completed successfully, and the Grammar Conversion Wizard created a new file. (This message does not indicate validation, which must take place in the Grammar Editor.) |
| Fail (Red) | The conversion did not complete properly, and the new file was not created. Read the message for more information. |
| Skipped (Black) | The file was already converted, and in that process, you selected not to overwrite this original file. |

6. Click **Finish** to complete the Wizard.


*Figure 4-10 Conversion Complete*

**Note:** Conversion from JSGF to SRGS involves certain rules for conversion of IBM ECMAScript Action Tags to SISR and their use in accessing simple and complex objects in VoiceXML. The Voice Toolkit help facility in Rational Application Developer covers these rules, and you can access the rules by clicking **Help** → **Help Content,** then clicking **Developing Voice Applications** → **Grammar Editor** → **Converting JSGF to SRGS**.

5

# Testing Grammars on MRCP

The Test Grammar on MRCP tool tests the grammars you created in the toolkit, or grammars that exist on a Web server, for example, Speech Recognition Grammar Specification (SRGS) XML or ABNF. The toolkit submits the grammars via the Media Resource Control Protocol to a local or remote WebSphere Voice Server.

You can use this tool to test grammars using speech, text, or enumeration.

**Note:** The Test Grammar on MRCP tool displays up to 100,000 characters. If the test exceeds that number of characters, the tool clears the buffer and refreshes the display. It does not return results of ECMAScript Objects or translations. The tool uses the default pronunciations from the speech recognition engine, unless the grammars you are testing reference a lexicon document.

The Voice Toolkit gives you a graphical method to test your grammars. For command line interaction, use the voiceTest tool which is available on the actual WebSphere Voice Server installation. We explain the voiceTest tool in Section 10.10.1, “Testing from the command line on a WebSphere Voice Server machine”, of *IBM WebSphere Voice Server for Multiplatforms V5.1.1/V5.1.2 Handbook*, SG24-6447.

## 5.1 Validating grammars on the MRCP server

Our example includes a grammar for determining the stock quotes of fictitious company names. We also utilize semantic interpretation to define combinations, types of request, and the resulting fields that you can pass to a database server. You can find this file in “stock.grxml” on page 123.

The grammar allows the user to ask for stock information such as the low, high, close, and so on. It also allows for varying combinations of the same company name, such as:

- Current price of ITSO Blue
- High for ITSO Blue Incorporated
- News about IBM
- Yesterday’s close of Tivoli® Software

We will use the Integrated Runtime Environment (in effect, the local Voice Server) to test the grammar. By default, if you install this optional Voice Toolkit environment, the configuration uses *localhost* as the host name of the Voice Server. To change the configuration to use a remote Voice Server, refer to 3.4.1, “Configuring Toolkit MRCP settings” on page 23.

Before selecting a grammar test, first insure you have started the Integrated Runtime Environment (refer to 3.3.1, “Starting the integrated voice server” on page 21); otherwise, you receive a connection error, such as the one in Figure 5-1.

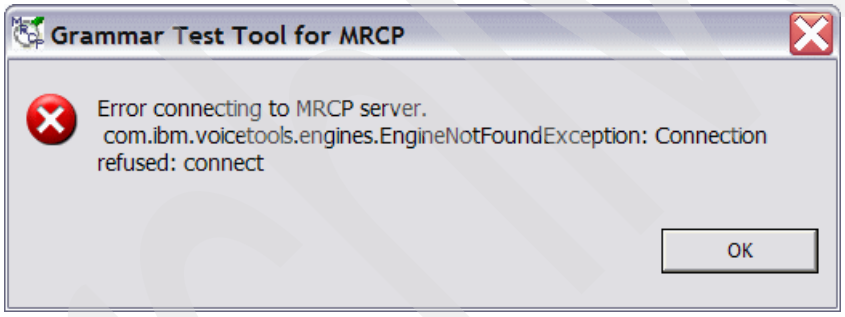



*Figure 5-1 MRCP server connection error*

To begin enumerating a grammar save and selecting the grammar file in the Navigator View, click **Run** → **Test Grammar on MRCP.** The grammar compiles and lists any problems in the Status column (see Figure 5-2 on page 47). Right-click the status and click **Show Details** for further information. Correct any errors in the grammar file before proceeding.

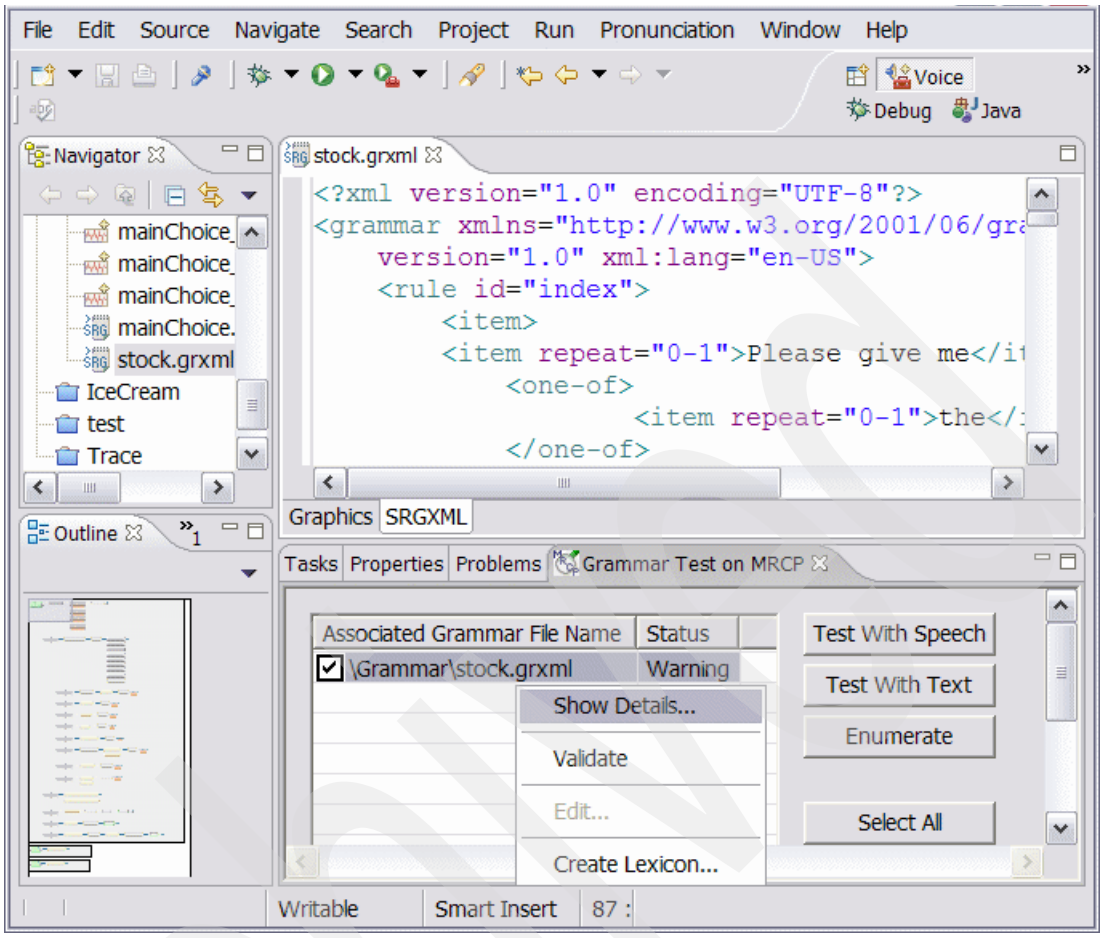



*Figure 5-2 Grammar Test status*

In this case, there was a Warning that the term *WebSphere* was added automatically.

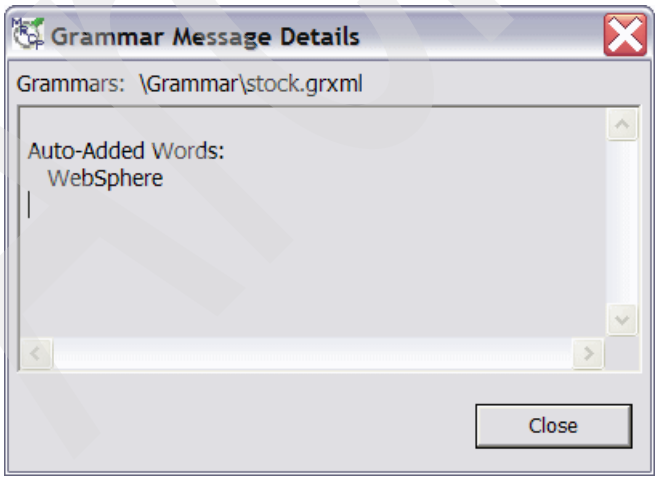



*Figure 5-3 Grammar Test status details*

## 5.2 Enumerating a grammar

The primary purpose of the enumeration function during grammar development is to check to make sure the grammar is not producing unexpected output, such as “I T S O Blue please please”. Finding these types of unexpected phrases provides clues about how to fix bugs in the grammar.

Open the Enumeration dialog by clicking **Enumerate** in the Grammar Test on MRCP view.

In the Options section, complete the following:

- In the **Number of Random Sentences** field, specify the number of random sentences to generate. The program randomly generates sentences until it reaches the specified number. The default value is zero. If you use the value zero, the program lists all valid words and phrases of the grammar.
- In the **Maximum number of words per Sentence** field, specify the maximum number of words to include in the test sentences. The program tests the entire grammar, creating sentences of the specified length. The default is twenty-five.

  **Note:** The program might display duplicate sentences, if there were not enough unique sentences generated.

- Select **Show probabilities** to show probabilities in the Test Results field.
- Select **Show Semantic Interpretation** to show the semantic interpretation in the Test Results field.

Click **Enumerate** to begin the test. Figure 5-4 shows a short list of random responses.

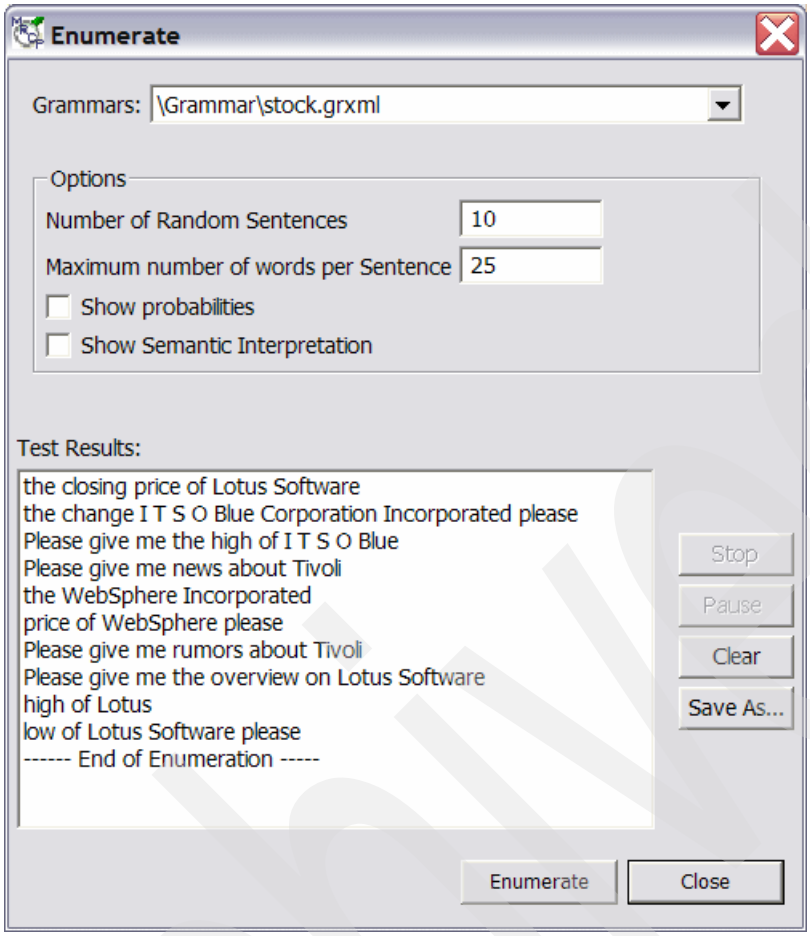



*Figure 5-4 Enumerate random sentence results*

Selecting zero random sentences (see Figure 5-5) lists all valid words and phrases in the grammar. This can take a considerable amount of time to complete for very large grammars.

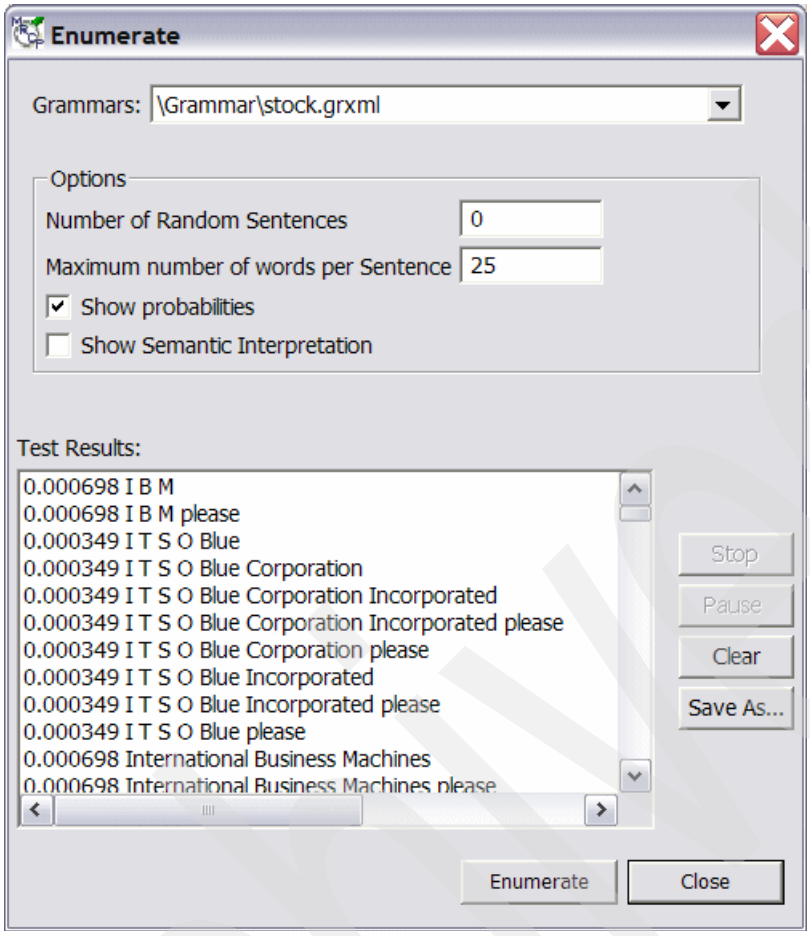



*Figure 5-5 Enumeration probabilities results*

Probabilities shown in the left portion of the Test Results field are the probability of taking that path in your grammar. The sum of all probabilities equals one.

Our stock grammar sets various tags to utilize as parameters for accessing a back-end database. The Semantic Interpretation shows the resulting values for each iteration of the grammar. Refer to Figure 5-6.

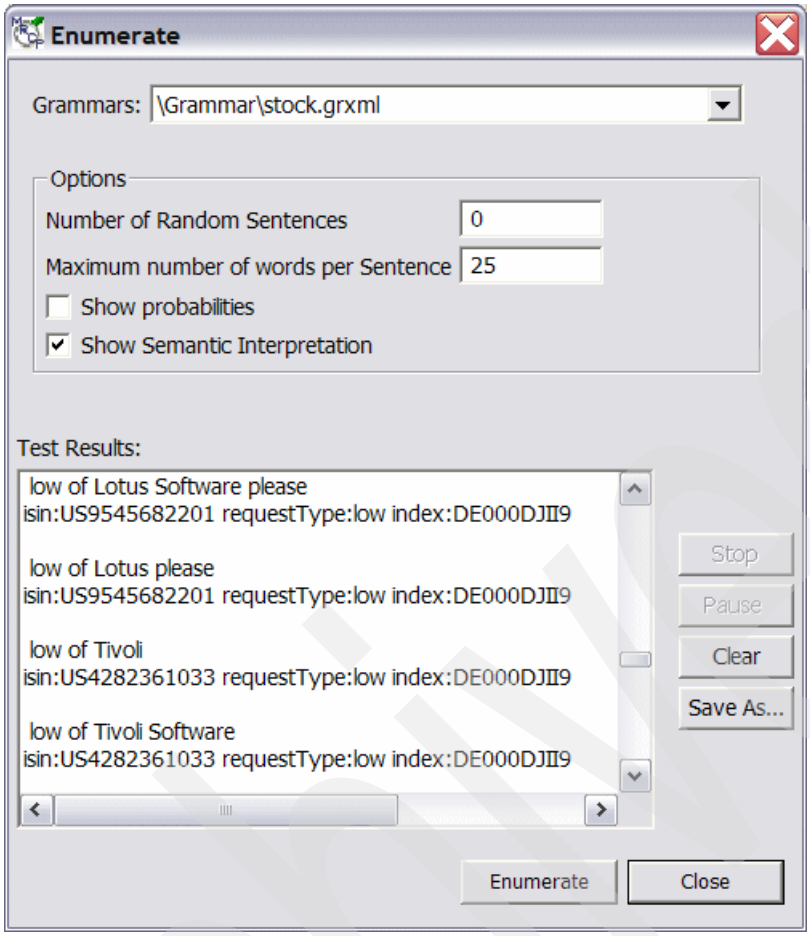



*Figure 5-6 Enumerate Semantic Interpretation results*

## 5.3 Testing grammars with text

Click **Test With Text** in the Grammar Test on MRCP view and you will see a dialog similar to Figure 5-7.

In the Options section, select how you want to test the grammar:

- Select Test With String (default setting) if you want to type a string of words into the text field. You can also select a text string from the menu list. The list displays the last five text strings.
- Select Test With File to test text from a file. Click **Browse** to locate the path of the file.
- Select **Show Semantic Interpretation** to show the semantic interpretation in the Test Results field.

Click **Run Test** to begin the test.

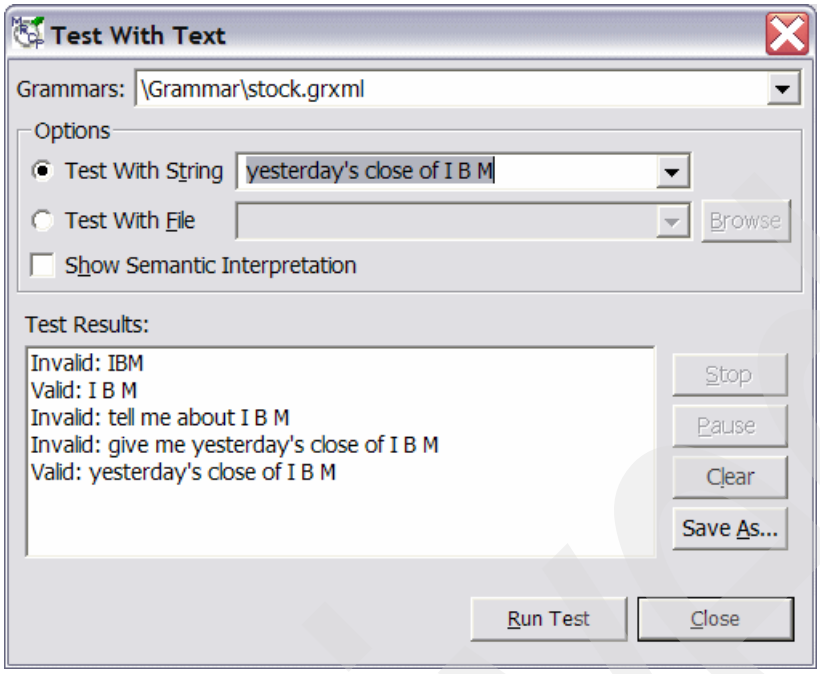



*Figure 5-7 Test With Text string results*

In these results, we see that additional changes to the grammar are necessary to handle the user response, `give me yesterday's close of I B M`.

**Tip:** If you are developing a grammar, then one of the iterative techniques is to create a text file that contains phrases that the grammar should recognize and use Test With File to ensure that all the phrases are valid in the grammar. If not, fix the grammar and retest until they all work. This is a complementary test to enumeration. You do not normally want to have to retype repeatedly tested phrases using Text With String. If you find a new string that you need to test repeatedly, add it to the file.

## 5.4 Testing grammars with speech

Testing with speech requires you to prerecord your utterances and save them individually in a file. You can use the Voice Toolkit to record each utterance.

### 5.4.1 Recording utterances

Begin by creating a new audio file in your project:

1. Click **File** → **New** → **Audio File**.
2. Enter the name of the audio file and select the parent folder. You can use the default AU file type and mu-law compression format setting in Test With Speech.
3. Click **Finish**.
4. Click Record  (see Figure 5-8) to begin recording your audio. For best results, pause briefly before you begin speaking.
5. When you finish, click Stop  . This automatically saves the audio file in the project.
6. To play the audio file, click Play  .

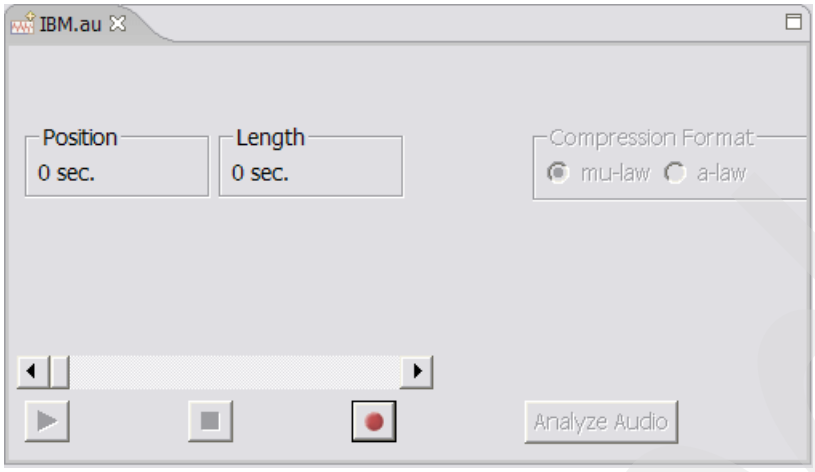



*Figure 5-8 Audio recorder*

## 5.4.2 Speech dialog

Open the Speech dialog by clicking **Test With Speech** in the Grammar Test on MRCP view.

1. In the Bulk Audio Test field, click **Add** and select the audio files to use for testing. You can navigate to your saved project audio files in your Rational Application Developer workspace:

   *IBM RAD root dir*\workspace\<project name>\WebContent

2. Select one or more audio files to use for the test.
3. Click **Open.**
4. Optionally, edit the Expected Results for the audio files by clicking on the Expected Results field for the audio file and editing the field directly.
5. Optionally, click **Save** to save all files as Comma Separated Value (CSV) files that contain the path to audio files and the expected result for each file.
6. Optionally, select or deselect the options in the Test Results field.
   a. Select **Show Statistics** to show the statistics of the tests run, such as Number of tests, Number failed, Percent failed, and so on.
   b. Select **Failures Only** to show only the failed tests. If the utterance does not match the Expected Result, this displays as a failure.
   c. Select **Show Confidence Scores** to show how confident the engine is that it recognized the word or phrase correctly. Results appear in the Test Results field. If the test recognizes the phrase, the program displays the confidence score in brackets right after the utterance.
   d. Select **Show Semantic Interpretation** to show the semantic interpretation in the Test Results field.
   e. Select **Max N-Best.** Enter the number of N-Best matches you want to display. The default is five.
7. Click **Run** to begin the test.

In Figure 5-9, we used two utterances and entered the Expected Results.

**Note:** Note that the second Expected Result did not match, since we entered:

```
IBM
```

instead of the recognized

```
I B M
```

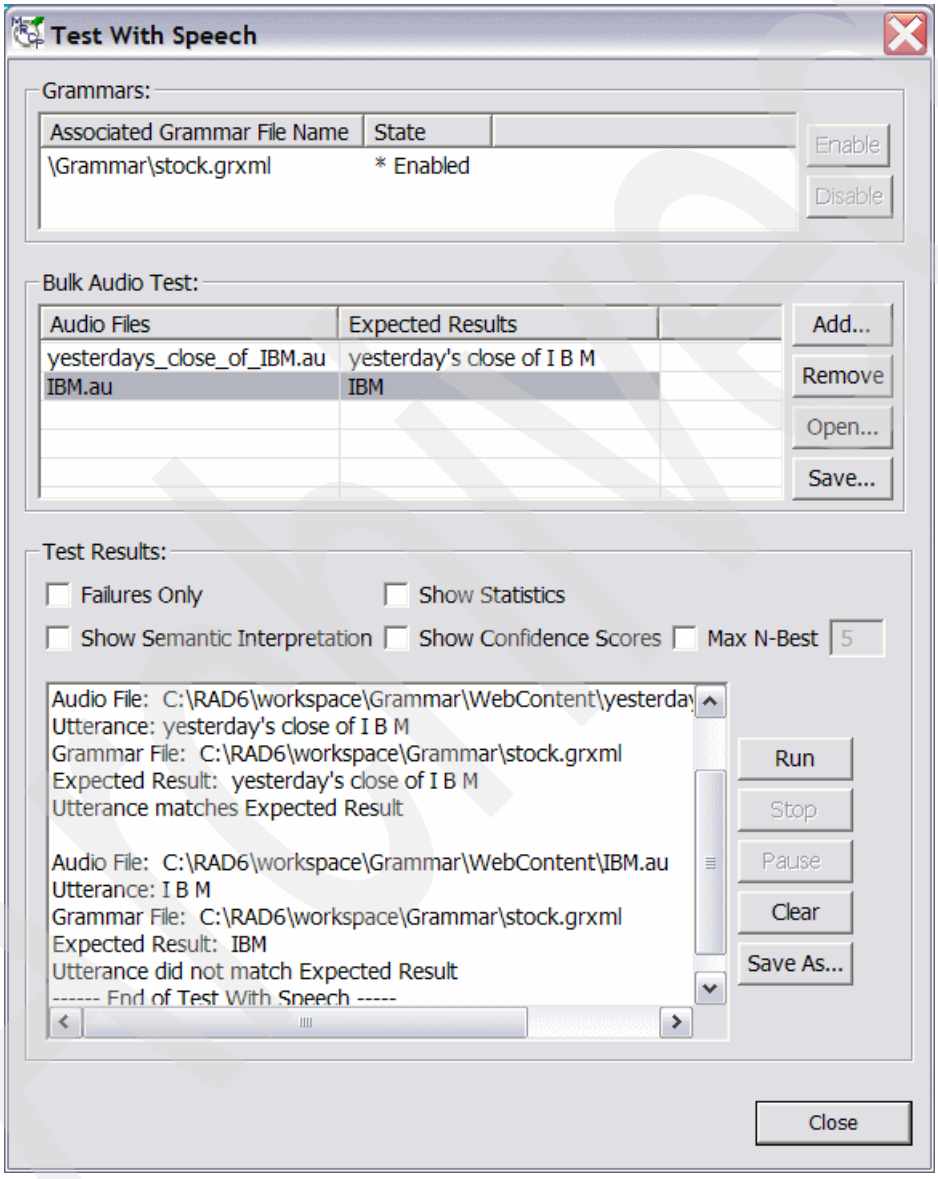



*Figure 5-9 Test With Speech (no options) results*

In Figure 5-10, we changed the options slightly by requesting the Confidence Scores and Max N-Best. Setting Max N-Best tells the recognizer to return results when there can be more than one possible match to a grammar. Our test case shows the recognizer determined there are two possible results:

- The utterance, `yesterday's close of I B M`, with a score of 84 matched our expected result.
- The utterance, `the yesterday's close of I B M`, with a score of 54 did not match our expected result.

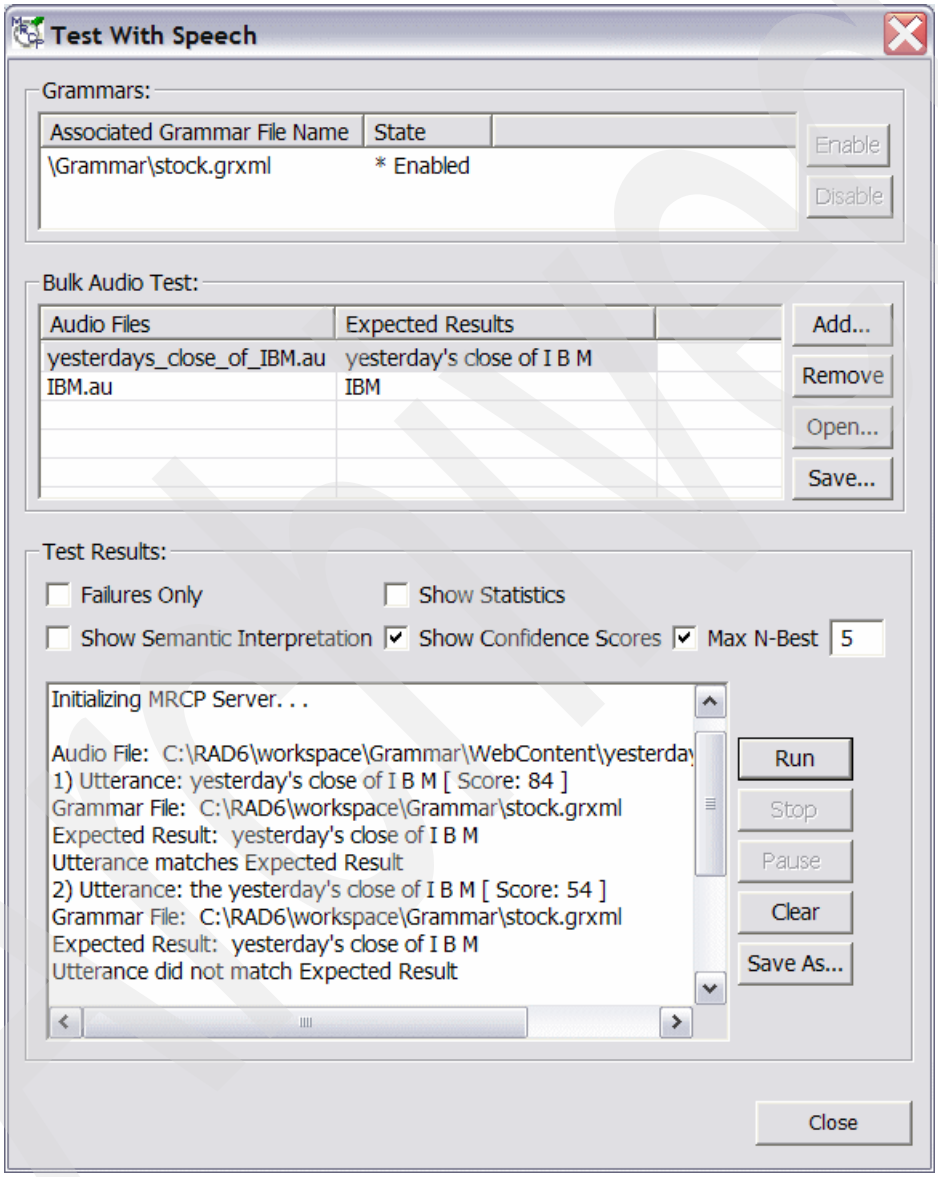



*Figure 5-10 Test With Speech (Confidence Scores and Max N-Best) results*

In Figure 5-11, we added two more audio files to get no-match conditions. The audio file containing the phrase, `tell me about IBM`, produces the expected no-match. The second file is more interesting. The phrase, `give me yesterday's close of IBM`, produces an invalid result for Test With Text. But for Test With Speech, it matched both phrases, `the yesterday's close of I B M` and `the yesterday's close of I B M please`. Both have a confidence score of 79.

**Note:** You might ask yourself how this can happen that the text phrase fails to match the grammar. The spoken phrase matches. Determine the relevance of this to whether or not these results indicate a need to change the grammar or the test case.

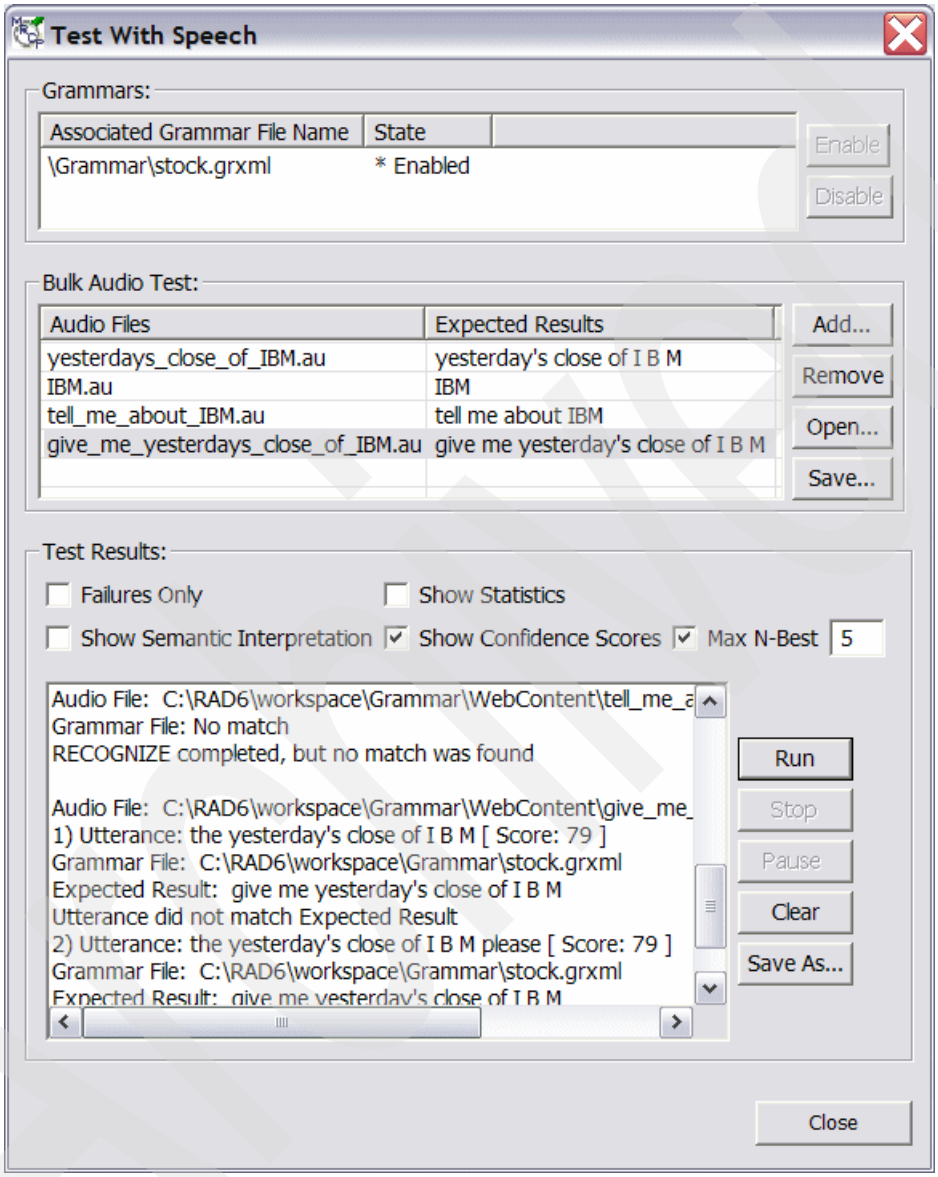



*Figure 5-11 Test With Speech (No-Match Grammar) results*

6

# Editor and Pronunciation Builder for lexicons

In this chapter, we cover the use of the Voice Toolkit as a tool for handling ASR and TTS lexicons.

We are only going to describe the most important features and basic concepts for creating and modifying lexicons. For more details, go the IBM Information Center:

http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/index.jsp

Alternatively, see the section, “Developing Voice applications”, in the Help - IBM Rational Software Development Platform.

## 6.1 Definition of a lexicon

For speech technologies, *lexicons* are pieces of code which contain word spellings and their corresponding phoneme sequences.

A *lexicon* creates a component that tells the Automatic Speech Recognition (ASR) and Text to Speech (TTS) systems how to pronounce a word (or a word sequence). The lexicon writes these pronunciations in a file using the Pronunciation Lexicon Markup Language (LXML).

IBM supplies its recognition and synthesis engines with lexicons containing pronunciations for the vast majority of words. In addition, the engine can produce dynamic pronunciations for unknown words based on the spelling.

For instance, handling proper nouns or uncommon acronyms requires this process. Although the automatic pronunciation generation is good in most cases, you may want to customize some pronunciations in an attempt to improve the ASR accuracy or the TTS quality.

We discuss 9.3, “Using lexicons for pronunciations” on page 109 to show how to improve results when using tuned pronunciations.

## 6.2 Lexicon Editor

The Voice Toolkit provides a Lexicon Editor that color codes the source code as you type. See Example 6-1. A Content Assist window displays when pressing the Ctrl-Space bar. It provides a list of valid tags for the element or attribute at the cursor location. For additional information about the use of the Lexicon Editor, refer to Section, “Developing Voice applications” in the Help - IBM Rational Software Development Platform.

*Example 6-1 Example of a lexicon with default syntax colors*

```
<?xml version="1.0" encoding="UTF8"?>
<!DOCTYPE lexicon PUBLIC "-//com/ibm/speech/grammar/lexicon//DTD Lexicon 1.0//EN"
"ibmlexiconml.dtd">

<lexicon version="1.0" xml:lang="en-US" alphabet="x-ibmasr" case-sensitive="false">

  <import uri="sourcelexicon.xml"/>
 <lexeme>
    <spelling>preferred</spelling>
    <phoneme>P R AX F ER DD</phoneme>
    <phoneme>P R IX F ER DD</phoneme>
  </lexeme>
 <lexeme>
     <spelling>colour</spelling>
     <spelling>color</spelling>
     <phoneme>K AH L AXR</phoneme>
 </lexeme>
 <lexeme>
    <spelling>IEEE</spelling>
    <sounds-like>I triple E</sounds-like>
  </lexeme>
</lexicon>
```

## 6.3 Creating a lexicon file

The following process first creates an empty lexicon file with the appropriate XML header. Then, this process guides you through adding your words with their pronunciations.

**Note:** IBM voice products give you the freedom to customize your pronunciations the way you want. Although this should not be a problem in most circumstances, for optimal ASR and TTS results, there might be pronunciation rules to take into account. Pronunciation rules are language-specific. If you are not satisfied with the system performance, talk to your IBM representatives to get this information from IBM development.

Use the following steps to create a new lexicon file (see Example 6-1 on page 58):

1. If you do not have a project where your lexicon file resides, create a voice project by navigating to **File** → **New** → **Voice Project**. Otherwise, select the desired project in the Navigator.
2. You can create a subfolder within the WebContent folder in which to place your lexicon file. To do this, select **File** → **New** → **Other** → **Simple** → **Folder** and type the folder name. *Folders* are optional containers for organizing files within a voice project.
3. In the Navigator view, select (highlight) the folder you created for your lexicon file, and then, select **File** → **New** → **Lexicon File**.
4. The New File wizard appears and guides you through the steps for creating a new Grammar file. Select the manner in which you want to create the new Grammar file, from scratch, resulting in an empty file, or with DTD assistance.
5. Click **Next**.
6. In the **File Type** box, select the type of file:

- **Recognition**

  Creates pronunciations for the speech recognition engine using International Phonetic Alphabet (IPA) phonologies. Select this option when you create grammars with words that you expect users to say when interacting with the application. This is the default selection.

- **Synthesizer (Text to Speech)**

  Creates pronunciations for the synthesized speech (TTS engine) using Symbolic Phonetic Representations (SPR) phonologies. Select this option when you create grammars with words that the synthesized voice will say to users.

**Note:** You want to add pronunciations to a Synthesizer lexicon file anytime you find a word that the TTS engine is not pronouncing correctly by default, whether the word is produced by virtue of being in a grammar tag or in a prompt/message.

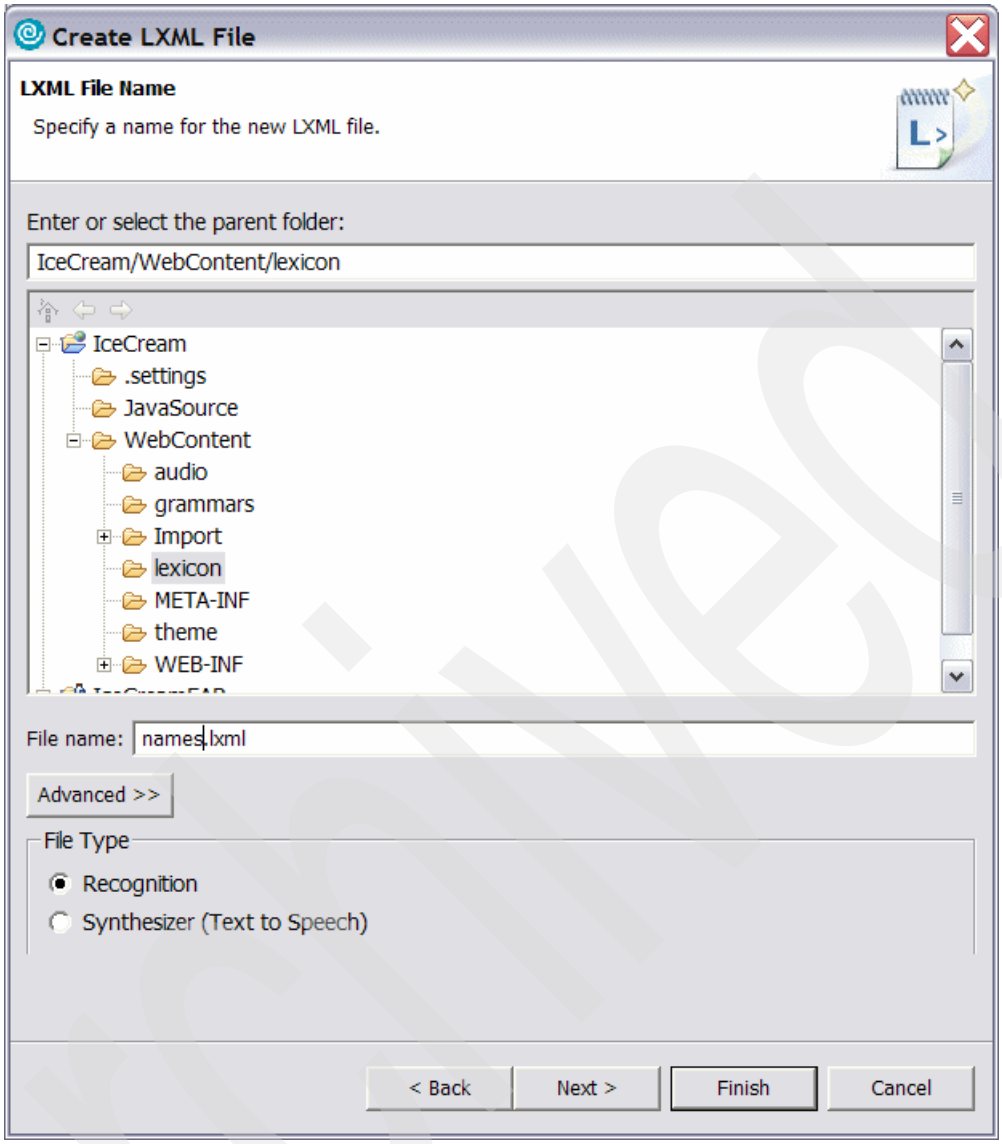



*Figure 6-1 Create LXML file*

7. Type your new file name. The default extension for a lexicon file is .lxml.

> **Tip:** Filenames are case sensitive. You should always make the lexicon browser URLs case-consistent to avoid duplicate loads. For example, do not name one file A.lxml and another file a.lxml.

8. Optional, click **Advanced** (available in some installed products) to reveal or hide a section of the wizard used to create a linked file. Select **Link to file in the file system** if you want the new file to reference a file in the file system. In the field below the check box, enter a file path or the name of a path variable. Click **Browse** to locate a file in the file system. Click **Variables** if you want to use a path variable to reference a file system file.
9. Click the **Finish** button and the Lexicon Editor launches your file with the basic <lexicon> tag.

You can now add your words within the <spelling> and </spelling> markups and pronunciations within the <phoneme> and </phoneme> markups, but we strongly recommend the use of the Pronunciation Builder whenever it is possible.

## 6.4 Add words and pronunciations with Pronunciation Builder

Now, you have created your lexicon file, but it contains no words or pronunciations yet.

**Note:** If you want to play pronunciations, you must connect to an MRCP server. To use the local MRCP server, simply go to **Run** → **Start Voice Server**.

Perform the following steps to create a pronunciation:

1. In the Lexicon Editor, position the cursor between the <lexicon> and </lexicon> tags, right-click, and click **Compose Pronunciation**. You should see a window similar to Figure 6-2.

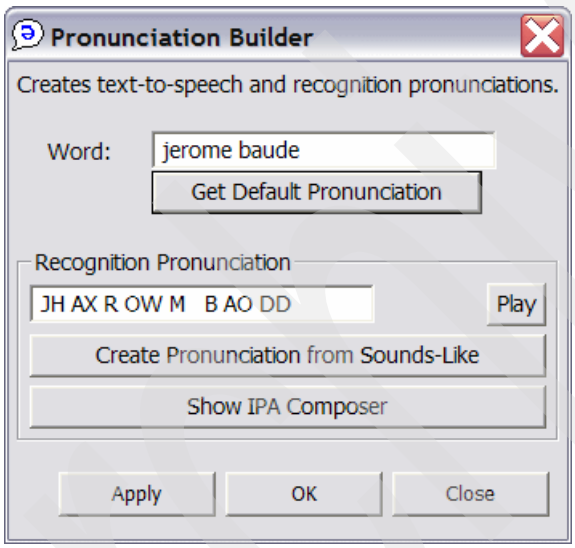



*Figure 6-2 Pronunciation Builder window*

2. In the Pronunciation Builder, type a word and click **Get Default Pronunciation**.
3. Test the pronunciation at any time by clicking **Play**.
4. If the default pronunciation is **incorrect**, use one of the options on the dialog box to create or tune the pronunciation. You create or tune the pronunciation by using one of the methods below:
   - To create a new pronunciation using a sounds-like spelling, click **Create Pronunciation with Sounds-Like** and type a word or phrase that is spelled the way you want the target word pronounced.
   - To edit a pronunciation, such as tuning the pronunciation produced from the spelling or sounds-like spelling, click **Show IPA Composer**. Select the symbol buttons for each sound in the word. Study the IPA phonology symbols with the representative words. The symbols are pronounced like the underlined sound in the representative word. For more detailed information about using the IPA composer, refer to the Section, “Developing Voice applications” in the Help - IBM Rational Software Development Platform.

– You can also modify the pronunciation by typing IBM internal phonemes into the text box. You can find the definitions for the IBM internal symbols in the IBM Information Center for IBM WebSphere Voice products.

http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/index.jsp?topic=/com.ibm.websphere.wvs.doc/wvs/dev_tune.html

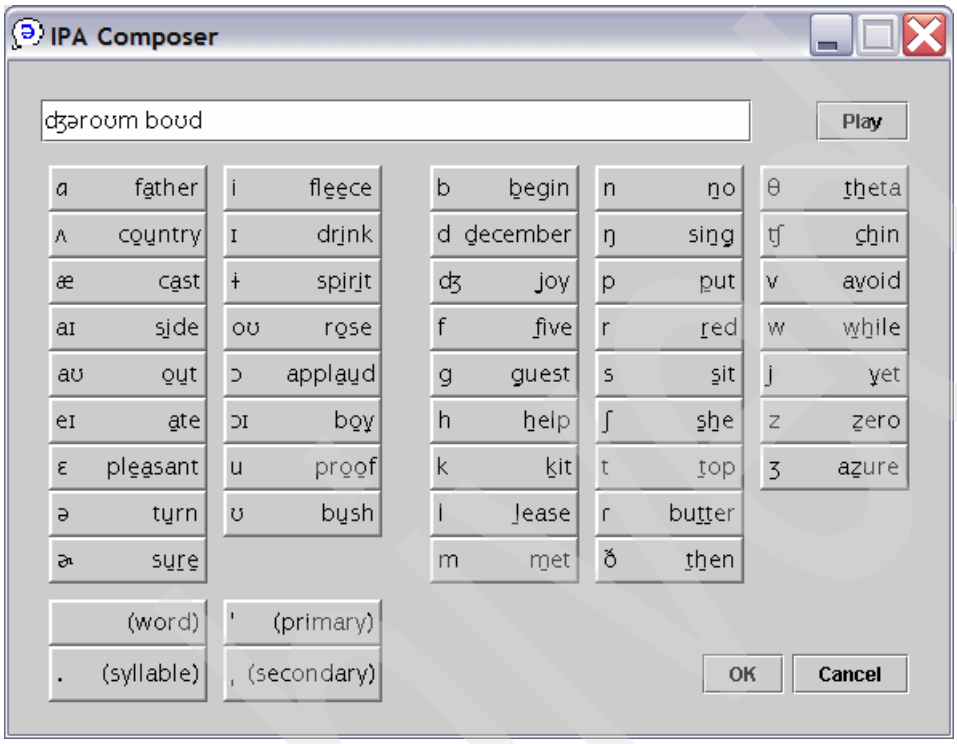



*Figure 6-3 IPA Composer*

5. When you are satisfied with the pronunciation, click **OK,** and you add the word and phonology to the lexicon file with the correct tags.
6. Continue to add the pronunciations, one word at a time. You can add multiple pronunciations for the same word, if necessary.
7. When you finish, save your file.

Example 6-2 and Example 6-3 on page 63 show what a lexicon file might look like for ASR and TTS respectively.

*Example 6-2 our names.lxml for ASR*

```
<?xml version="1.0" encoding ="UTF-8"?>
<!DOCTYPE lexicon PUBLIC "-//com/ibm/speech/grammar/lexicon//DTD Lexicon 1.0//EN"
"ibmlexiconml.dtd">

<lexicon version="1.0" xml:lang="en-US" alphabet="x-ibmasr" case-sensitive="false">
   <lexeme>
      <spelling>gary elliott</spelling>
      <phoneme>G EY R IY   EH L IY IX TD</phoneme>
   </lexeme>
   <lexeme>
      <spelling>james chamberlain</spelling>
      <phoneme>JH EY M Z   CH EY M B AXR L IX N</phoneme>
   </lexeme>
   <lexeme>
```

```
      <spelling>markus klehr</spelling>
      <phoneme>M AA R K AX S   K L EH R</phoneme>
   </lexeme>
   <lexeme>
      <spelling>jerome baude</spelling>
      <phoneme>JH AX R OW M   B OW DD</phoneme>
   </lexeme>
</lexicon>
```

*Example 6-3 Our names.lxml for TTS*

```
<?xml version="1.0" encoding ="UTF-8"?>
<!DOCTYPE lexicon PUBLIC "-//com/ibm/speech/grammar/lexicon//DTD Lexicon 1.0//EN"
"ibmlexiconml.dtd">

<lexicon version="1.0" xml:lang="en-US" alphabet="x-ibmtts" case-sensitive="true">
   <lexeme>
      <spelling>chamberlain</spelling>
      <phoneme>.1Cem.0bR.0lXn</phoneme>
   </lexeme>
   <lexeme>
      <spelling>elliott</spelling>
      <phoneme>.1E.0li.0Xt</phoneme>
   </lexeme>
   <lexeme>
      <spelling>klehr</spelling>
      <phoneme>.1klEr</phoneme>
   </lexeme>
   <lexeme>
      <spelling>baude</spelling>
      <phoneme>.1bod</phoneme>
   </lexeme>
</lexicon>
```

The previous guidelines only explain a way to create a lexicon file. The Section, “Developing Voice applications” in the Help - IBM Rational Software Development Platform describes other options.

## 6.5 Referencing a lexicon file

After you have created your own lexicon, you need to reference it. We look at referencing your lexicon in a grammar when used in ASR. We also consider the case of a lexicon designed for TTS usage.

### 6.5.1 Grammar

A grammar can optionally reference one or more external pronunciation lexicon documents. You identify a lexicon document by a URI with an optional media type.

Example 6-4 and Example 6-5 on page 64 show the ABNF and XML forms respectively.

*Example 6-4 ABNF form*

```
#ABNF 1.0 iso-8859-1;
language en-US;

mode voice;
```

```
root $main_rule;
tag-format <semantics/1.0>;

lexicon <http://www.example.com/names.lxml>;

public $main_rule=...
```

*Example 6-5 XML form*

```
<?xml version="1.0" encoding ="iso-8859-1"?>
<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN" "http://www.w3.org/TR/speech-grammar/grammar.dtd">
<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar" xml:lang="en-US" root="main_rule">

   <lexicon uri="http://www.example.com/names.lxml"/>
   <rule id=”main_rule” scope=”public”>
...
```

## 6.5.2 TTS

We can preload TTS lexicon files from the WebSphere Application Server Administrative Console. See Section 4.3.4, “Configuring ASR and TTS”, in *IBM WebSphere Voice Server for Multiplatforms V5.1.1/V5.1.2 Handbook*, SG24-6447.

We can also specify a different lexicon in a SSML document (see Example 6-6):

http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/topic/com.ibm.voicetools.reldocs.doc/tts_ssml.pdf

*Example 6-6 Use of a lexicon file in a VoiceXML document using the SSML tag <lexicon>*

```
...
<prompt>
<lexicon uri=”http://www.example.com/names2.lxml”/>
...
</prompt>
...
```

**Note:** The TTS engine uses the first pronunciation when there are multiple pronunciations specified for a given word.

7

# Voice Trace Analyzer

The Voice Trace Analyzer lets you examine recognition data from an IBM WebSphere Voice Server system. Using the data obtained from the WebSphere Voice Server collection utility, it can read multiple trace.log files to build a comprehensive overview of your system's recognition performance and improve awareness of recognition problems. We implement the Voice Trace Analyzer as an Eclipse plug-in for use in Rational Application Developer.

Chapter 9 of *IBM WebSphere Voice Server for Multiplatforms V5.1.1/V5.1.2 Handbook*, SG24-6447, fully describes the Voice Trace Analyzer from V6.0 of the Voice Toolkit.

This chapter focuses on changes, such as computing Accuracy Values, to the Voice Trace Analyzer in the newer V6.0.1 Voice Toolkit.

## 7.1 Voice Trace Analyzer steps to obtain data for use

While we use the Voice Trace Analyzer primarily to process data from remote Voice Server installations, we cover how to generate and process data collected from your Integrated Runtime Environment. This gives you the opportunity to familiarize yourself with the Voice Trace Analyzer while working with data we generated in Chapter 5, "Testing Grammars on MRCP" on page 45. Once you collect the trace data on the Voice Server, the process for importing and viewing the data in the Voice Trace Analyzer tool is the same, regardless of whether the Voice Server is local or remote.

The steps to obtain data for use in the Voice Trace Analyzer from a local (Integrated Runtime Environment) or remote Voice Server are:

1. Set the Voice Server trace specification, which we cover in 7.2, "Setting up files for analysis" on page 66.
2. Run your voice application or test to generate trace data.
3. Collect (package) the trace data into a single file on the Voice Server. If you do not have access to the remote Voice Server file system, then this step is *required*. This step is *optional* if you have access to the local (Integrated Runtime Environment) Voice Server. We cover this in 7.3, "Running the Collector Tool" on page 74.
4. Import the data into the Voice Trace Analyzer, which we cover in 7.4, "Starting the Voice Trace Analyzer" on page 74.

## 7.2 Setting up files for analysis

In this section, we describe how you can use your Integrated Runtime Environment to collect the necessary information for use by the Voice Trace Analyzer.

**Tip:** When using the Integrated Runtime Environment as your local Voice Server, you have two options to input data to the Voice Trace Analyzer. You can create the Collector output (covered in 7.3, "Running the Collector Tool" on page 74) or point to the trace files directly. If you are going to point to the trace files directly, it is a good idea to erase all the files in the IBM_Rational_dir\runtimes\base_v51\logs\IBMVoiceServer directory, before starting the Voice Server and running your test, because you would not get the benefit from the Collector tool of removing old audio files.

You set the necessary trace configuration by using the Voice Server Administration Console. This console is similar to a WebSphere Voice Server Administration Console, except it does not include the Voice Server plug-in. Therefore, you cannot see the status of the Voice Server parameters, such as CPU utilization, number of engines running, and so on, on the main window, but you can enable tracing.

1. First insure that you have started your Integrated Voice Server (**Run** → **Start Voice Server**).
2. Open a Web browser to:

   `http://localhost:9091/admin`

   You should see something similar to Figure 7-1.

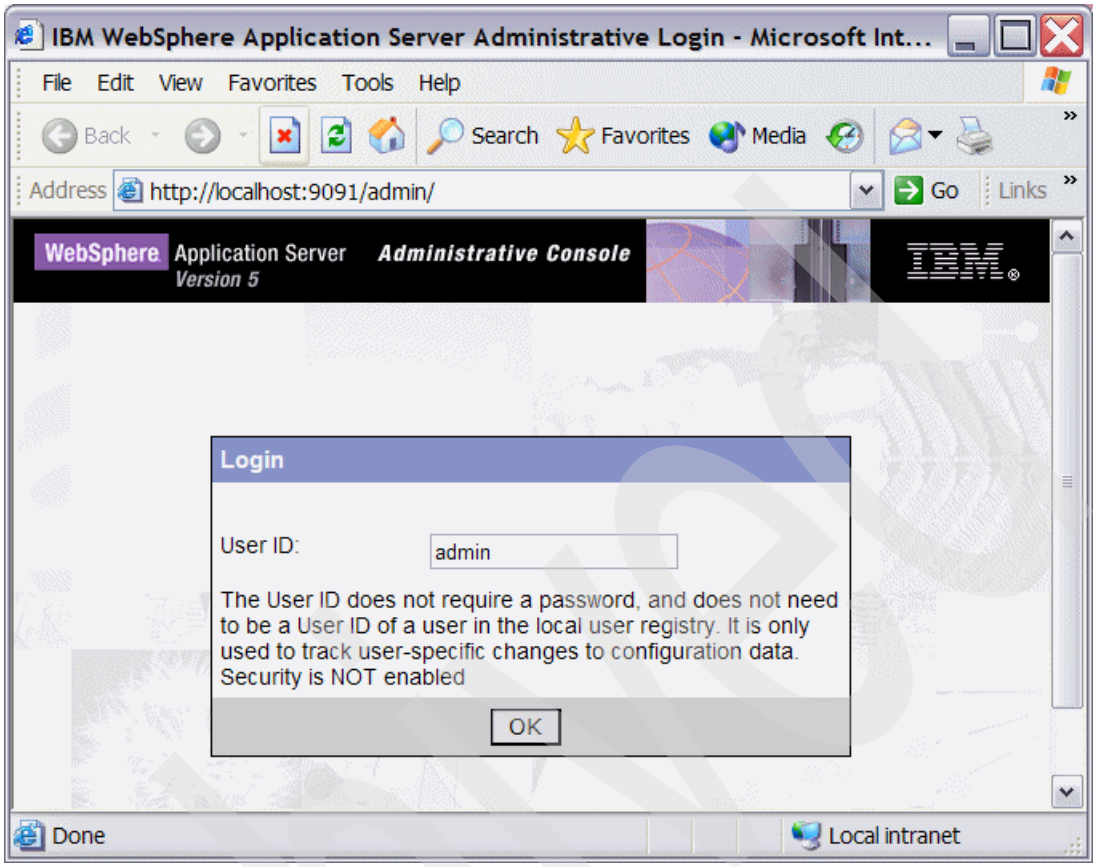



*Figure 7-1 Administrative Console Login*

3. Enter any name in the User ID field (for example, **admin**).
4. Click **OK**.
5. Click **+** next to **Troubleshooting** in the left frame.
6. Click **Logs and Trace**.

   You should see something similar to Figure 7-2.

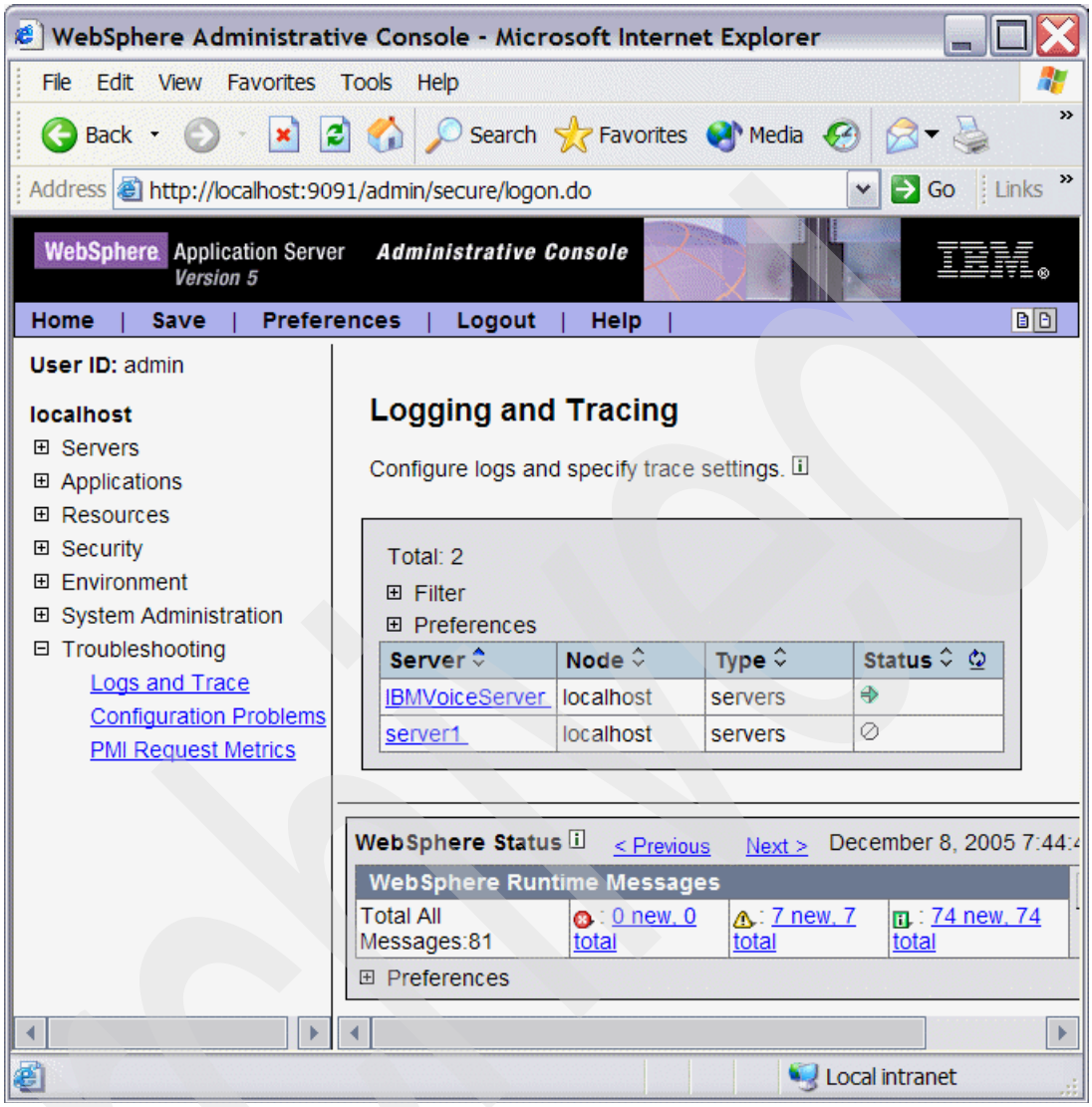



*Figure 7-2 Admin Console Logging and Tracing*

7. Click **IBMVoiceServer** in the Server column.

> **Note:** The green arrow in the Status column indicates the IBMVoiceServer is running.

You should now see something similar to Figure 7-3.

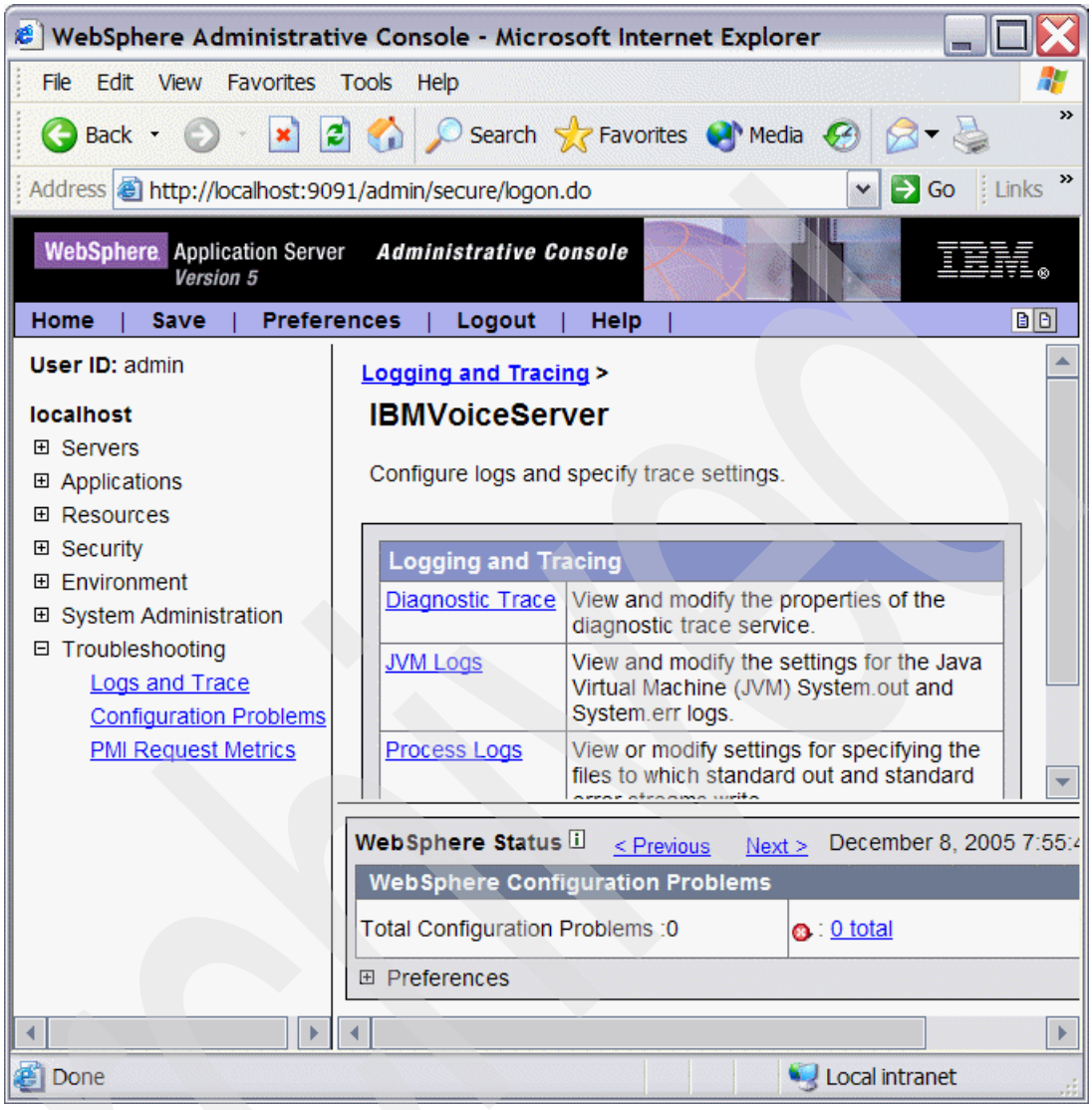



*Figure 7-3 Administrative Console IBMVoiceServer*

8. Click **Diagnostic Trace**.
9. You will see the Configuration and Runtime tabs. If you make changes on the Configuration page, the changes apply from then on whenever the Voice Server starts. Changes you make on the Runtime page are in effect for this session only. Refer to Figure 7-4.

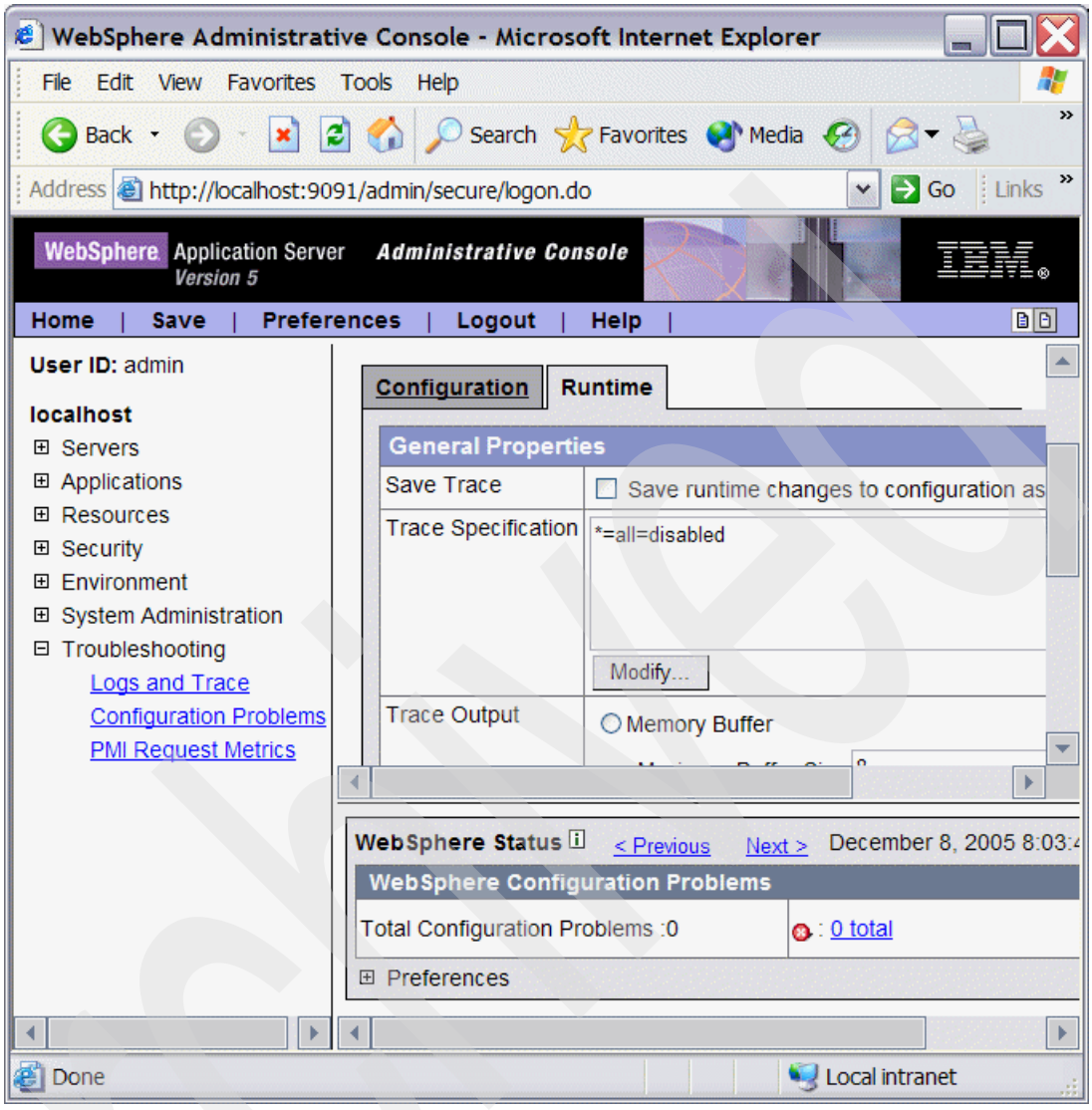



*Figure 7-4 Administrative Console Runtime tab*

**Tip:** It is not a good idea to run your Voice Server for extended periods of time at the trace level necessary for the Voice Trace Analyzer, because doing so saves all audio recognition data files. These audio recognition data files are only removed from the Voice Server when the Collector tool runs, and they will eventually fill up your hard disk (See 7.3, “Running the Collector Tool” on page 74).

Click **Runtime**.

10. Click **Modify** in the Trace Specification field.

You should now see the trace specification window as shown in Figure 7-5.

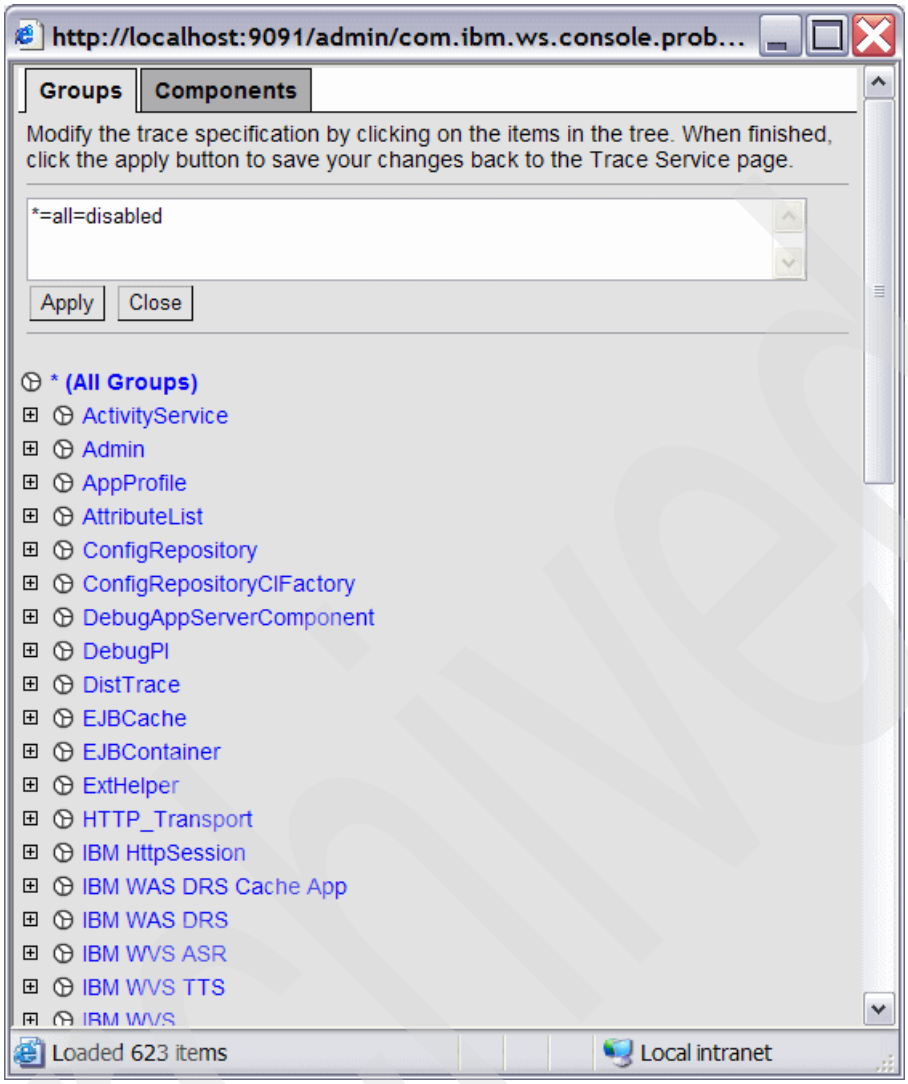



*Figure 7-5 Administrative Console Trace Groups*

11. You can either cut and paste the following text (exactly, with no spaces) into the Groups field (replacing `*=all=disabled`), or select each trace specification individually in the lower area:

```
ASRAPI=entryExit=enabled:ASRBEAN=event=enabled:MediaConv=entryExit=enabled,event=enabled
:RTSPBridge=all=enabled
```

To set the trace specifications individually (see Figure 7-6), expand the trace group, and then, select the trace specification and the trace level according to Table 7-1.

*Table 7-1 Trace specifications and levels*

| Trace group | Trace name | Trace level |
|---|---|---|
| IBM WVS ASR | ASRAPI | entry/exit |
| IBM WVS ASR | ASRBEAN | event |
| IBM WVS | MediaConv | entry/exit + event |
| IBM WVS | RTSPBridge | all enabled |

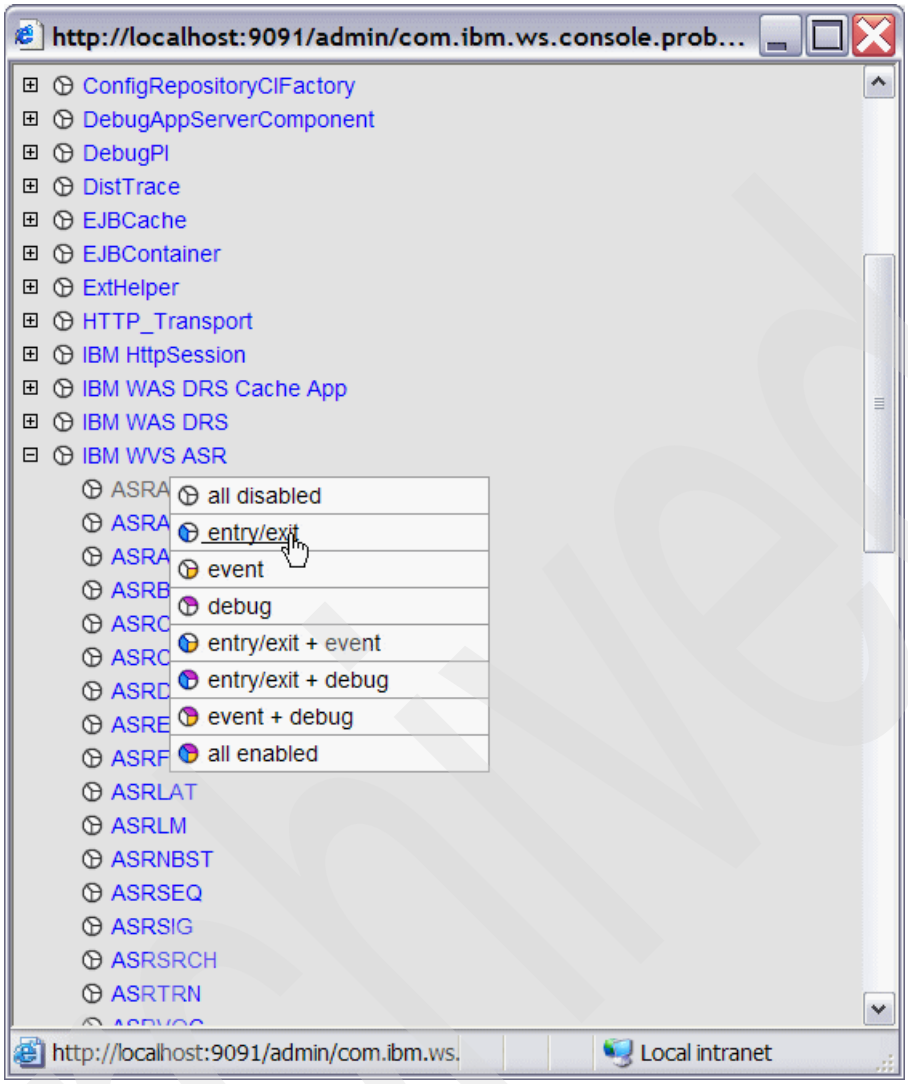



*Figure 7-6 Administrative Console: Setting trace specifications individually*

12. When you finish setting the trace specifications in the Groups view, click **Apply,** then click **Close.**

13. On the Runtime page view, scroll down, click **Apply**, and then, click **OK.**

    Optionally, you can verify that the change to the trace specifications occurred by viewing the WebSphere Voice Server trace.log file. You find this file in:

    IBM_Rational_dir\runtimes\base_v51\logs\IBMVoiceServer\

    At the bottom of the file, you should see:

```
Current trace specification =
ASRAPI=entryExit=enabled:ASRBEAN=event=enabled:MediaConv=entryExit=enabled:MediaConv=eve
nt=enabled:RTSPBridge=all=enabled
```

**Tip:** If you trace a live WebSphere Voice Server system, and you expect a significant number of calls, set the number of Trace History files to 20. This prevents the single trace.log file from rolling over. Also, avoid server restarts during the collection period, since server restarts lose session data.

You can now begin to run your VoiceXML application (**Run** → **Run as** → **VoiceXML Application (Audio mode)**) or test grammars (**Run** → **Test Grammar on MRCP**) in order to generate some trace data. The WebSphere Voice Server writes trace entries when it processes an MRCP request. When you complete this, you should then run the WebSphere Application Server Collector tool to assemble the appropriate WebSphere Voice Server trace data.

With trace now enabled, we reran the grammar test tool from Chapter 5, “Testing Grammars on MRCP” on page 45 with the three **Test With Speech** scenarios. The following audio files were saved from the trace specification settings (see Figure 7-7).

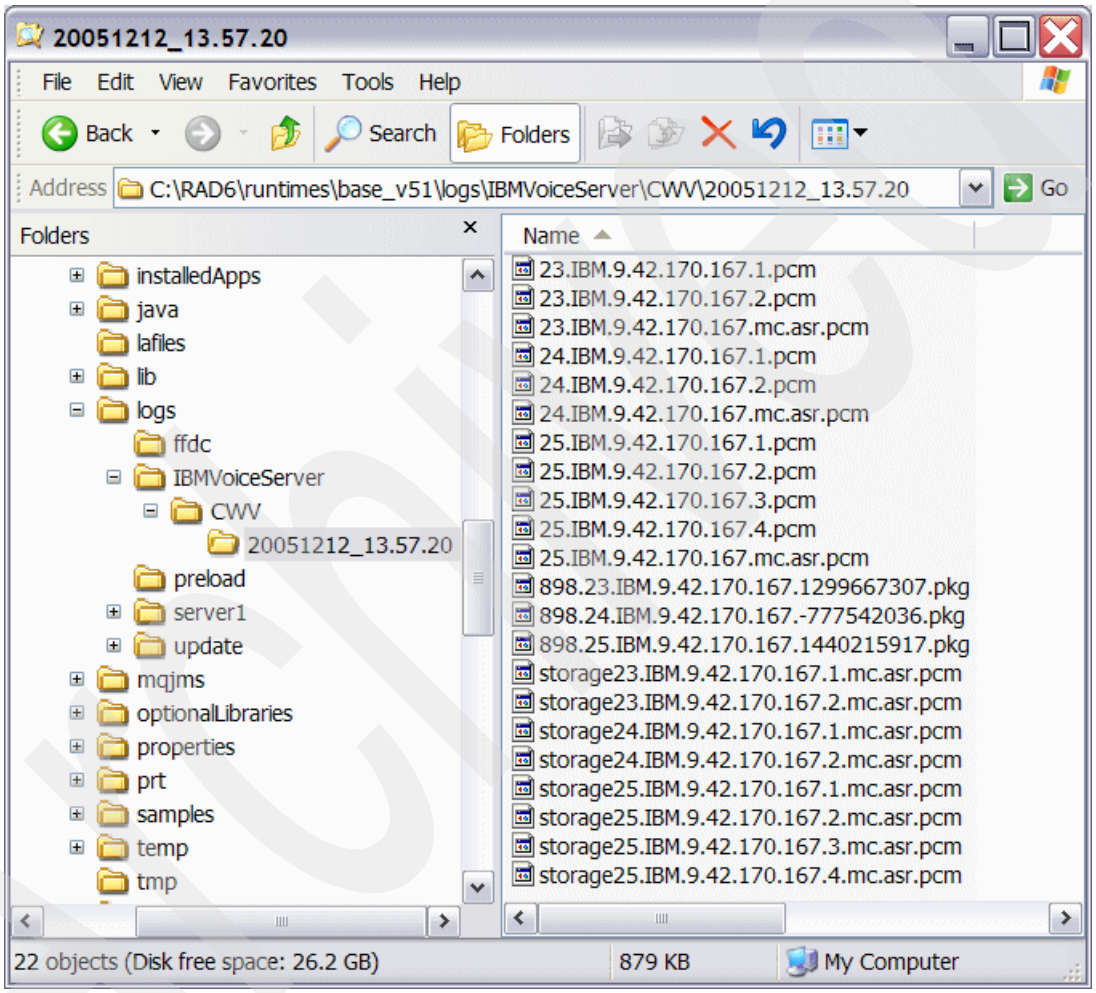



*Figure 7-7 Saved audio files*

**Note:** The trace specifications we mention save the default Media Converter audio files in the Voice Server. Therefore, the Voice Trace Analyzer only references Media Converter audio types. To see Endpointed and UnEndpointed audio files in the Voice Trace Analyzer, you need to set the appropriate ASRAUD=entryExit=enabled trace setting in the Voice Server. For additional information, refer to:

`http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/index.jsp?topic=/com.ibm.websphere.wvs.doc/wvs/trouble_trace.html`

Then scroll down to “Table 4. Additional log files”.

## 7.3 Running the Collector Tool

The Collector Tool compresses all the Voice Server information into a single JAR file. Normally, you use this utility to collect information to send to IBM Support for problem resolution. The Voice Toolkit uses this format to import data into the Voice Trace Analyzer. Follow these steps to run the Collector Tool in the Integrated Runtime Environment:

1. Make sure you log into your system with a user id that has administration authority, because the Collector Tool requires system access authority.
2. Open a command line window and execute the commands shown in Example 7-1.

*Example 7-1 Running the Collector Tool*

```
cd IBM_Rational_dir\runtimes\base_v51\bin
setupCmdLine
cd \temp
md work
cd work
"IBM_Rational_dir\runtimes\base_v51\bin\collector"
```

For the Integrated Runtime Environment, the resulting JAR file in your *work* directory will have a name similar to *machine name*--rad6-runts-base_v51-WASenv.jar where *machine name* is your computer name.

If you encounter problems running the Collector Tool or want detailed information, refer to the WebSphere Voice Server V5.1 Information Center:

http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/index.jsp

## 7.4 Starting the Voice Trace Analyzer

After setting the trace specifications, we ran the MRCP Grammar Test Tool against the stock.grxml file that we discussed in Chapter 5, “Testing Grammars on MRCP” on page 45. We now import the trace data into the Voice Trace Analyzer.

You will want to first create a new Rational Application Developer project and switch to the Voice Trace Analyzer perspective. In order to import the trace data, you create a new file:

1. Click **New** → **Voice Trace Analyzer File**.

   You should see a window similar to Figure 7-8.

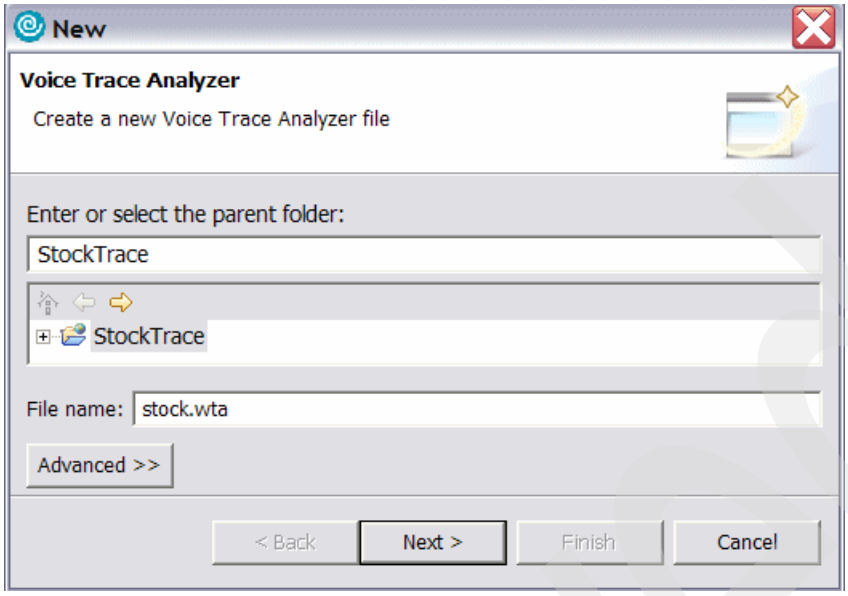



*Figure 7-8 Create new Voice Trace Analyzer file*

2. Click the parent folder and enter a name for the new Voice Trace Analyzer file (file type .wta).
3. Click **Next**.

   Now, you see a window similar to Figure 7-9.

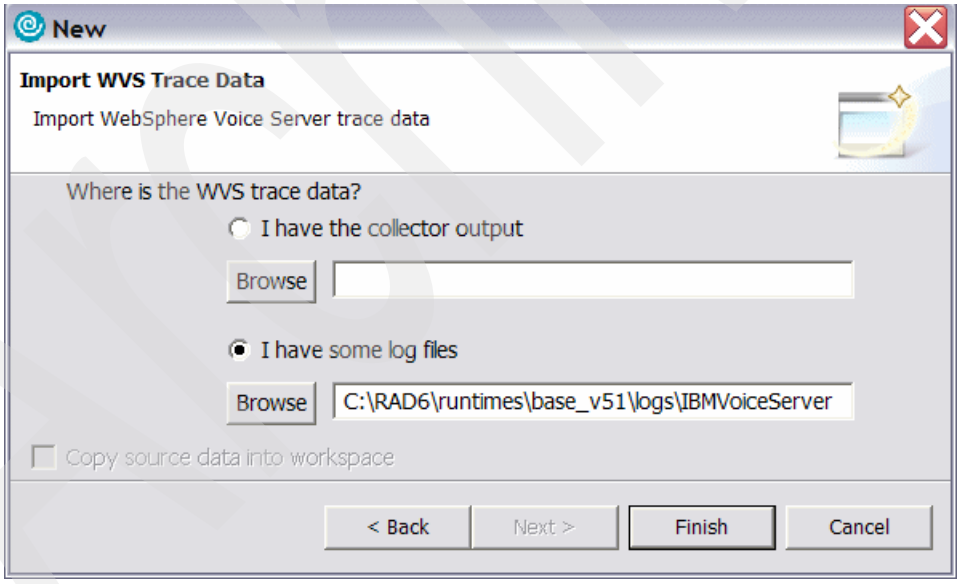



*Figure 7-9 Import new Voice Trace Analyzer file*

You now have the option to use the Collector output or point to the files directly (in the Integrated Runtime Environment).

4. Perform either (a) or (b):
   a. Click **I have the collector output**.

      Enter or browse to your collector JAR file location.

Check **Copy source into workspace**.

b. Click **I have some log files**.

Enter or browse to *IBM_Rational_dir*\runtimes\base_v51\logs\IBMVoiceServer

5. Click **Finish**.

**Important:** The **I have some log files** option does not copy the traces and audio files into your Rational Application Developer workspace. If you point to the local Integrated Runtime Environment, such as we did above, the data may no longer be there when you restart your Voice Server and run new tests. You should copy this directory if you intend to keep it.

## 7.5 Voice Trace Analyzer perspective

The Voice Trace Analyzer provides its own perspective of the log analyzer. It organizes the workbench into three windows:

- The outline view occupies the top left corner.
- The properties view occupies the bottom left corner.
- The editor window comprises the remainder of the screen.

### 7.5.1 Outline, Properties, and Editor Views

- Outline View

  The outline view is a tree with two root elements.

  - Filtered sessions
  - All sessions

  You can select filtered sessions manually by using one or more of the available search widgets of the editor window, or automatically by using the menu.

  The second root element is for all sessions. Opening the .wta file populates this list and this cannot be changed.

- Properties View

  Use the properties view to show information about the selected session. The information contained in this view is as follows:

  - Average Turn Duration
  - Completed (Can be modified by the user)
  - Duration
  - End time
  - Gender (Can be modified by the user)
  - ID
  - Last utterance (Can be modified by the user)
  - Longest turn
  - Number of turns
  - Start time

- Editors View

  The editor window contains two primary sections. The first is located at the top and contains widgets for selecting filters and preferences. The rest of the editor window is allocated to a tab folder.

There are five tabs on the editor window:

- Recognitions

  Shows a list of recognitions for all the filtered sessions with selected sessions highlighted

- Grammars

  Shows the grammars for the session selected in the outline or for all sessions

- Problems

  Shows all recognitions that were not successful for the filtered sessions

- Call Flow

  Shows a graphical representation of the session selected in the outline

- Statistics

  Shows basic statistical information about the data

The Outline View in Figure 7-10 shows three sessions, since we made three separate tests with the grammar test tool.

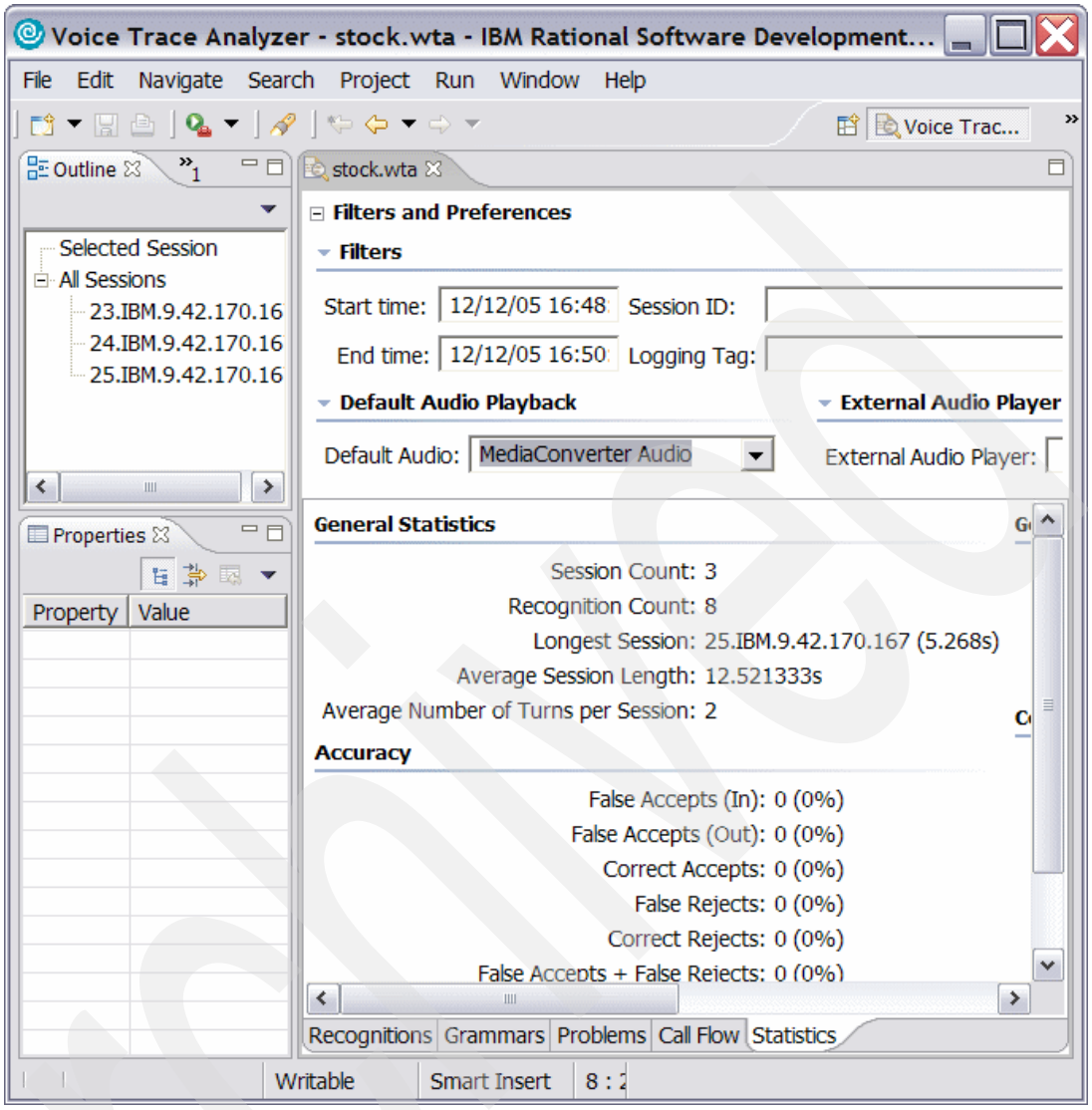



*Figure 7-10 Default Statistics tab*

The three sessions are:

- No options selected
- Statistics and Max N-Best selected
- No-match added

The Statistics tab in the Editors View shows various overall statistics of the entire file. The Accuracy data is blank until you provide the translations for each turn and have the toolkit compute the accuracy.

Refer to 7.11, “Statistics tab” on page 87 for additional information.

In order to view data in the remaining tabs (Recognitions, Grammars, Problems, and Call Flow), we need to add sessions to the Filter list.

Right-click on any session in the *Outline* View and click **add all sessions to the filter list.**

## 7.6 Recognitions tab

The Recognitions tab displays the recognitions (turns) for each session in the filtered session list.

On this page, you can right-click and select Computer Accuracy Values, so that for every turn, the Voice Trace Analyzer goes to get all the packages defined for the turn, picks up the transcription, and runs a grammar test. It only returns the transcription if there is one in the set of grammars. If there is no transcription available, the Accuracy column's value for the turn is cleared.

The Recognitions tab, the Grammars tab, and the Problems tab all use the Recognitions table. The rows of the Recognitions table contain 25 columns, each one describing an element of the recognition that occurred. Column descriptions are available in the online help.

Click **Recognitions**.

The Recognitions View in Figure 7-11 on page 80 has three scrollable windows. The first window (with the Session, Started, Completed, and so on columns) shows each turn for the sessions in the filter list. The Recognitions View highlights No-match conditions in red.

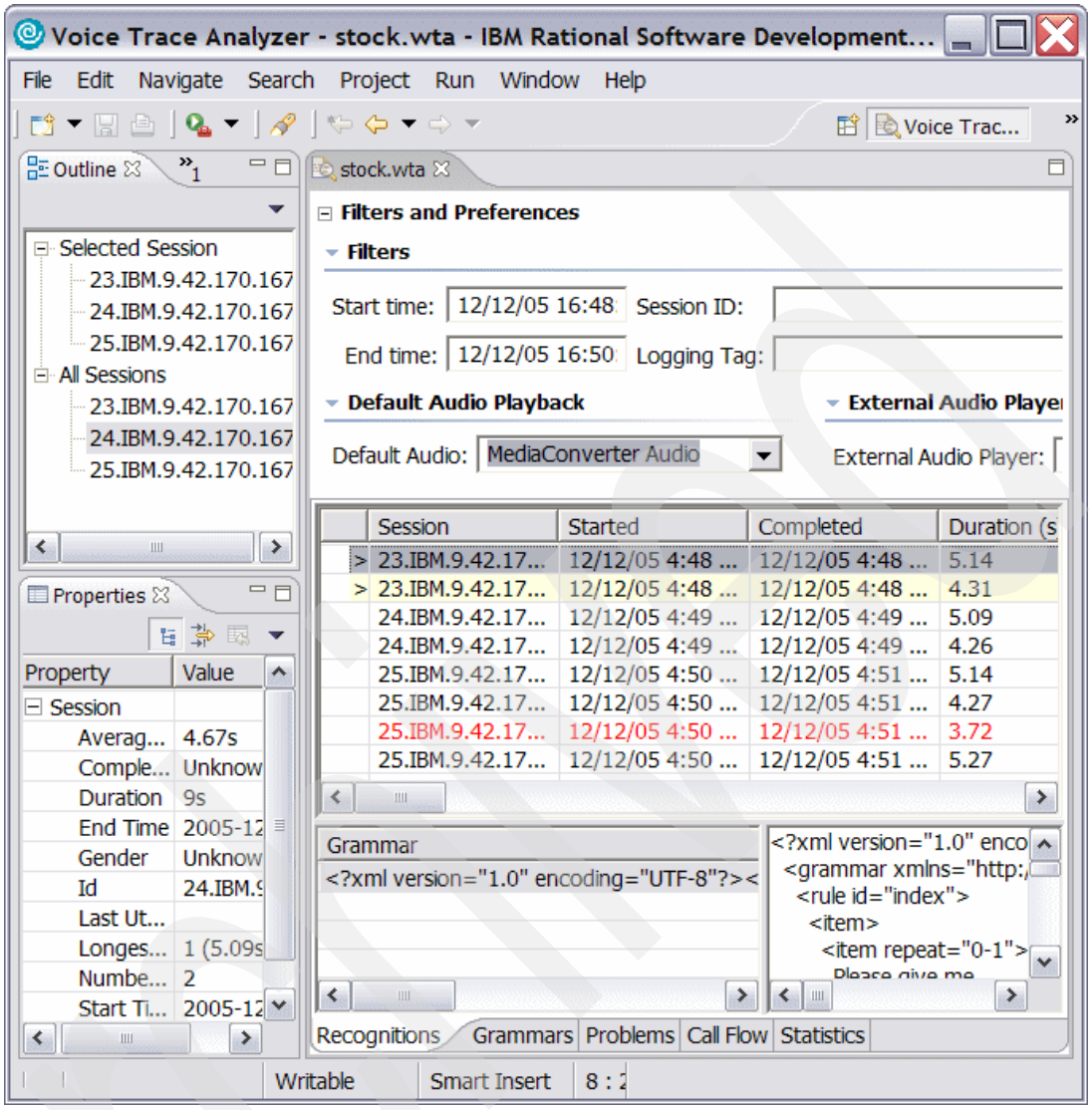



*Figure 7-11 Recognitions tab*

1. Click one of the turns.

The lower two windows show all the active grammars in the left window (in our case, there is only one) and the matched grammar for that turn in the right window.

## 7.6.1 Recognition context menus

You can customize the Recognitions table to use on the Recognitions tab, the Grammars tab, and the Problems tab. Right-click any turn (row) to show additional options you can perform for that turn, including showing detailed information, listening to and analyzing the user's spoken audio, and exporting and computing accuracy values. Refer to Figure 7-12.

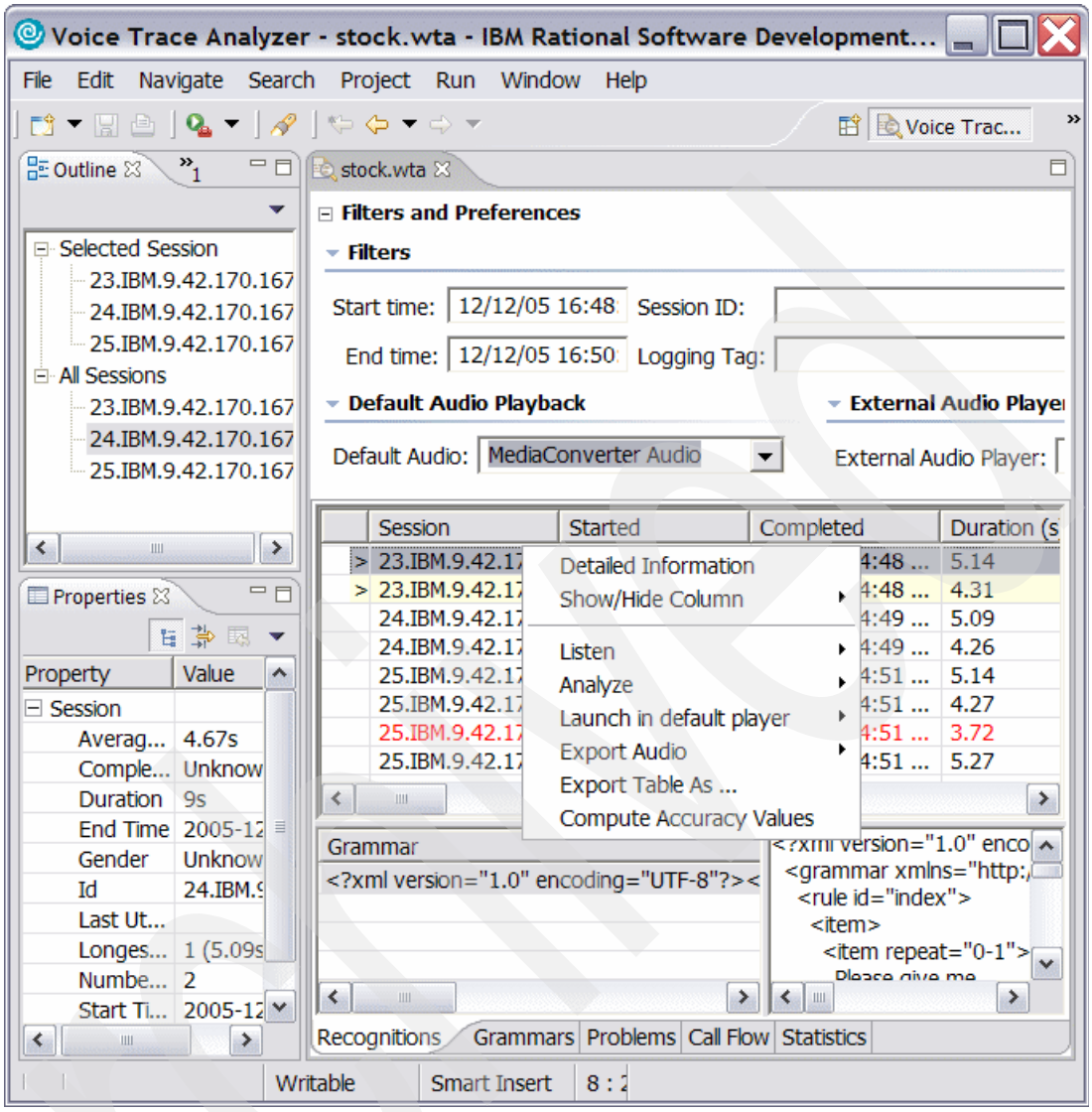



*Figure 7-12 Recognition context menu*

Figure 7-13 shows the results of right-clicking on the first turn in session 24 (the third turn in the overall list). Note that since we selected N-best for the second test grammar on MRCP, both N-Best results display with their confidence levels. A blank field (for example, Sensitivity Level or Speech vs. Accuracy) means that we did not specify information for that MRCP transaction (we used the Voice Server system defaults). This is typical when using the Test Grammar on MRCP tool. More values are available when testing VoiceXML applications using VoiceXML Application (Audio mode).

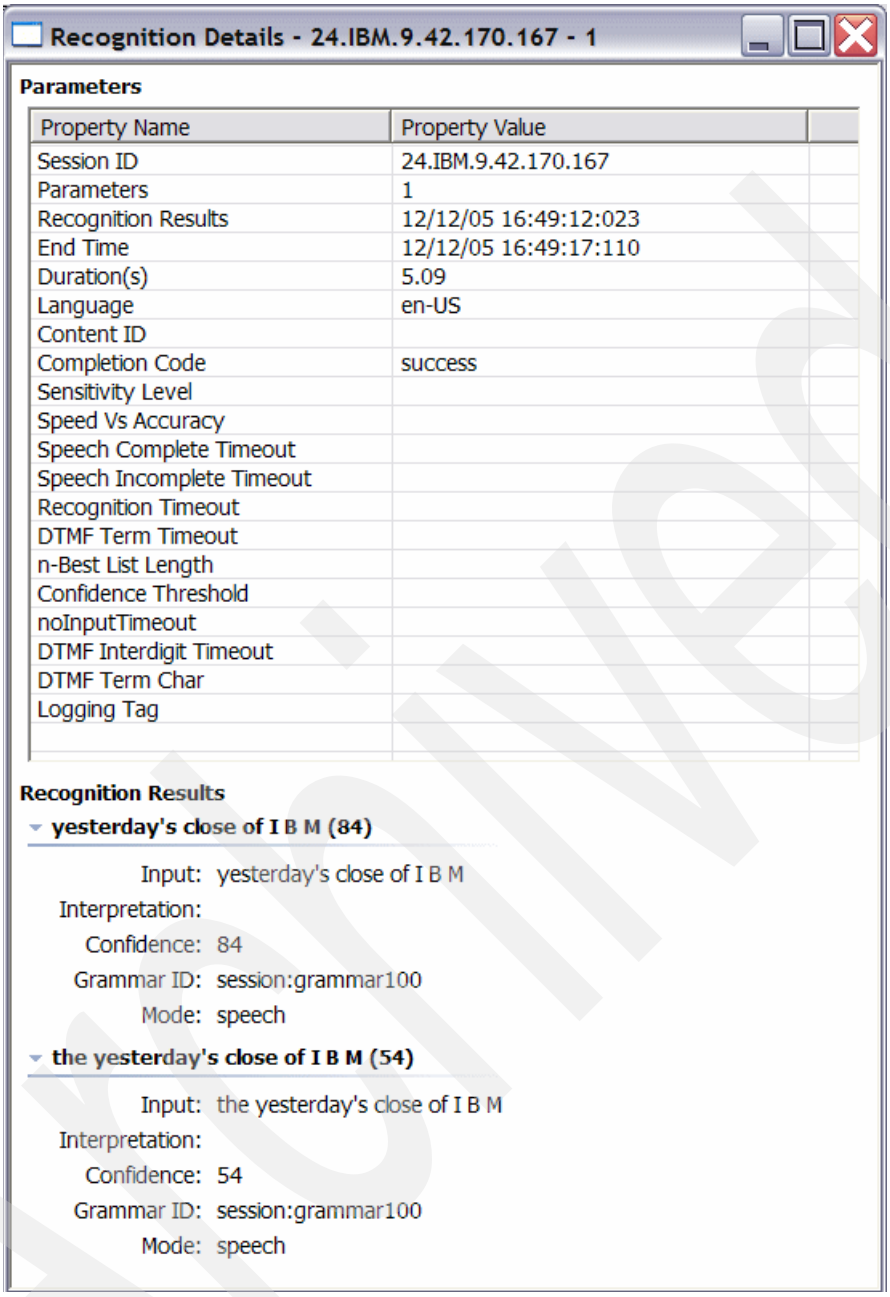



*Figure 7-13 Recognition Details*

## 7.7 Grammars tab

The Grammars tab (see Figure 7-14) shows you specific information regarding your grammars.

Select **All** to analyze your grammars from all sessions, or **Selected Sessions** to view only the sessions you have filtered.

This displays the grammar list in a table, the first column shows the detail of the grammar and the second column shows the number of times that the grammar was matched. The area, to the right of the second column, displays the complete, available detail of the selected

grammar, and the table at the bottom displays the recognitions that match the selected grammar.

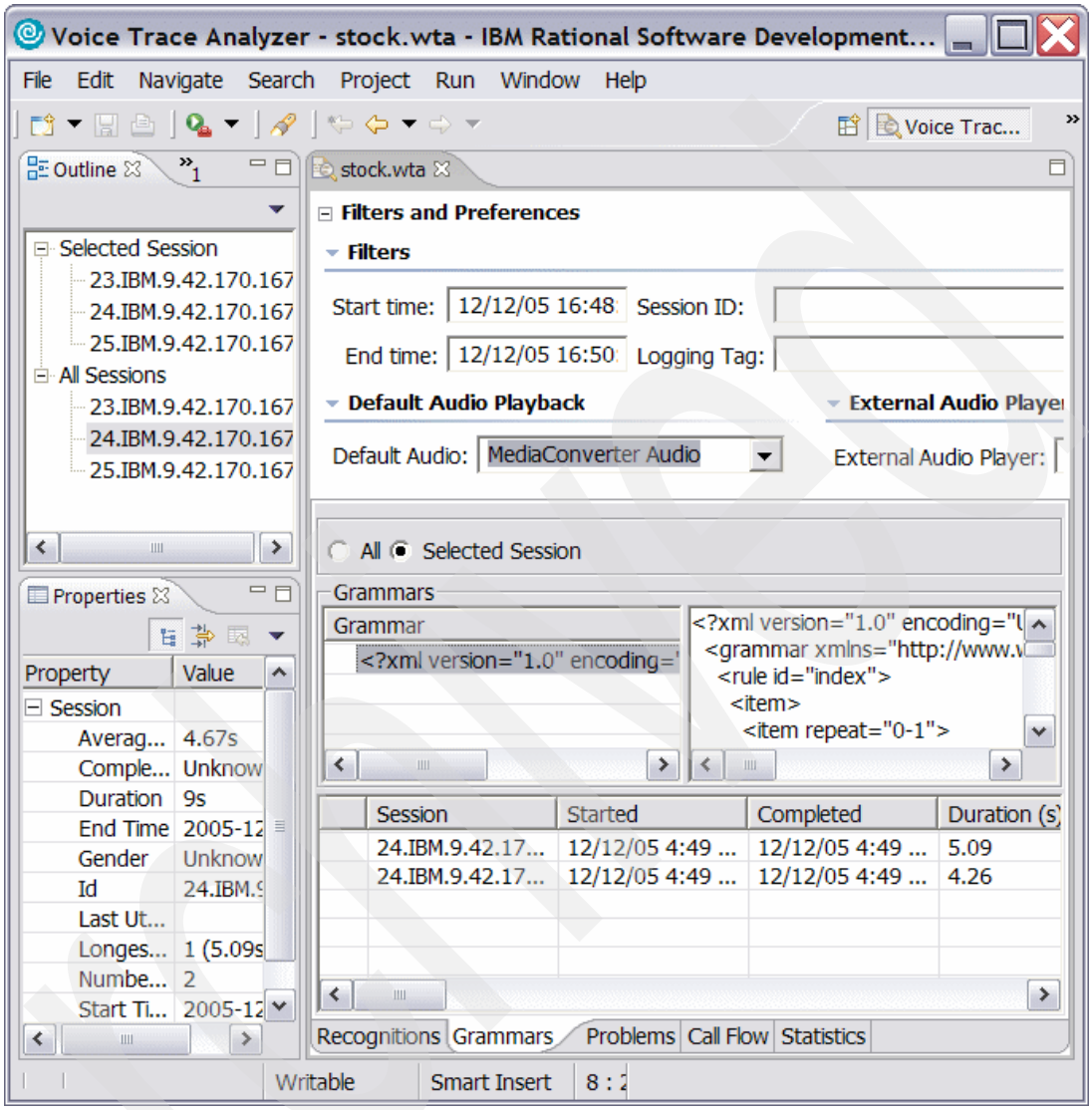



*Figure 7-14 Grammar tab*

## 7.8 Transcription Within Tool

The Voice Toolkit can calculate various accuracy results based on user-supplied transcriptions for each turn.

- In Grammar

  This value indicates whether the user's utterances should match one of the active grammars (In or Out of Vocabulary). Valid values are **Yes**, **No**, or empty. If the value is empty, this stores the calculated value in the database and shows the calculated value in the Turn column.

> **Note:** You must establish a working connection to the MRCP server in order to compute this column.

- Accuracy

  The Voice toolkit displays the accuracy (see Table 7-2) for each turn and saves the accuracy in the database. With this data, along with the Correct Accept and False Accept data, which is already available, the Voice toolkit also computes the other values and populates them in the Statistics tab. Valid values can be empty or one of the following:

  – Correct Accept (CA)

    The utterance matched the active grammars, and the speech recognition system correctly accepted the utterance.

  – Correct Reject (CR)

    The utterance did not match the active grammars, and the speech recognition system did not recognize the utterance.

  – False Accept (FA)

    The speech recognition system incorrectly recognized the utterance. This can happen for the following reasons:

    - FA-In

      The utterance matched the active grammars, but the speech recognition system chose the wrong in-grammar phrase to match it.

    - FA-Out

      The utterance did not match the active grammars, but the speech recognition system chose an in-grammar phrase to match it.

  – False Rejects (FR)

    The utterance matched the active grammars, but the speech recognition system did not recognize it.

*Table 7-2 Accuracy values*

| | **In Grammar** | **Out Grammar** |
|---|---|---|
| **Match** | CA | |
| **No Match** | FR | CR |
| **False Accept** | FA-In (substitution) | FA-Out |

A good speech recognition system tries to minimize the FA and FR percentages. The CA, CR, FA, and FR summary percentages also display on the Statistics page. Set the tracing level for the ASRAPI component on the Voice Server system in order to enable this functionality in the toolkit.

You can manually add transcriptions to the data in the Transcription table:

1. Click the **Recognitions** tab (see Figure 7-15).
2. Scroll to the right to display the Recognized Phrase and Transcription columns. For each turn:
   a. Double-click the speaker icon in the Recognized Phrase column to listen to the utterance.
   b. Click the adjacent field in the Transcription column and enter what was heard.

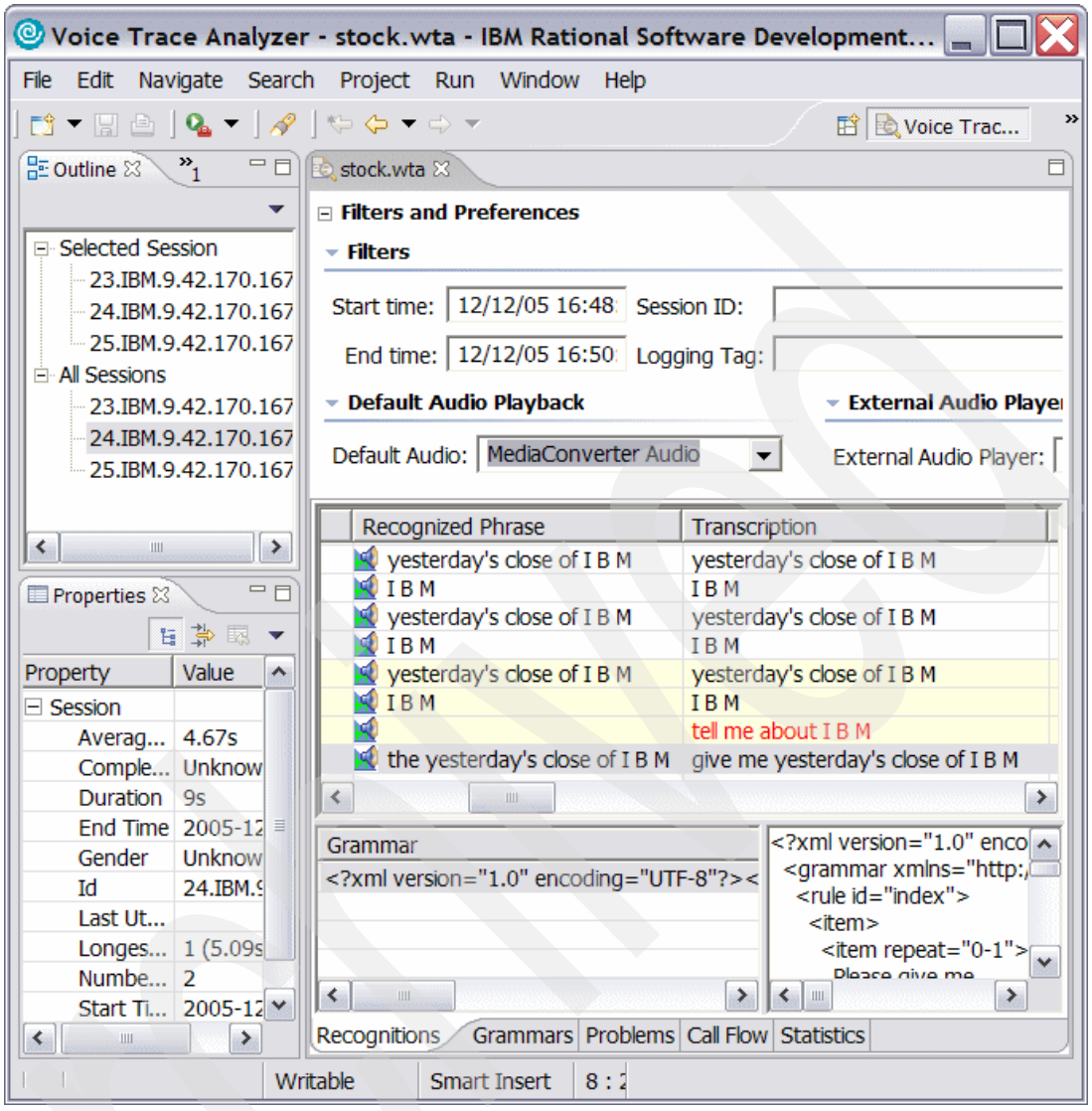



*Figure 7-15 Recognition transcriptions*

3. Right-click any turn in the Recognitions page to display the context menu and click **Compute Accuracy Values**.
4. Scroll right to show the calculated In Grammar and Accuracy results.

**Tip one:** In order to update the Statistics tab with the percentages, the Accuracy column in the Recognition table must be computed. To see the updated Accuracy information displayed on the Statistics tab, select the Refresh button on the Statistics tab.

**Tip two:** If you modify the trace specifications, be aware that both the In Grammar and Transcription columns must be completed in order to compute the Accuracy column. The In Grammar will be computed during Compute Accuracy Values if you use the trace specifications from 7.2, “Setting up files for analysis” on page 66, since this includes the grammar files.

## 7.9 Problems tab

The Problems tab (see Figure 7-16) shows all of the unsuccessful recognitions.

This tab is identical to the Recognitions tab. The only difference is that the Problems page only displays the recognitions with an unsuccessful completion code. All entries are red to indicate they were unsuccessful.

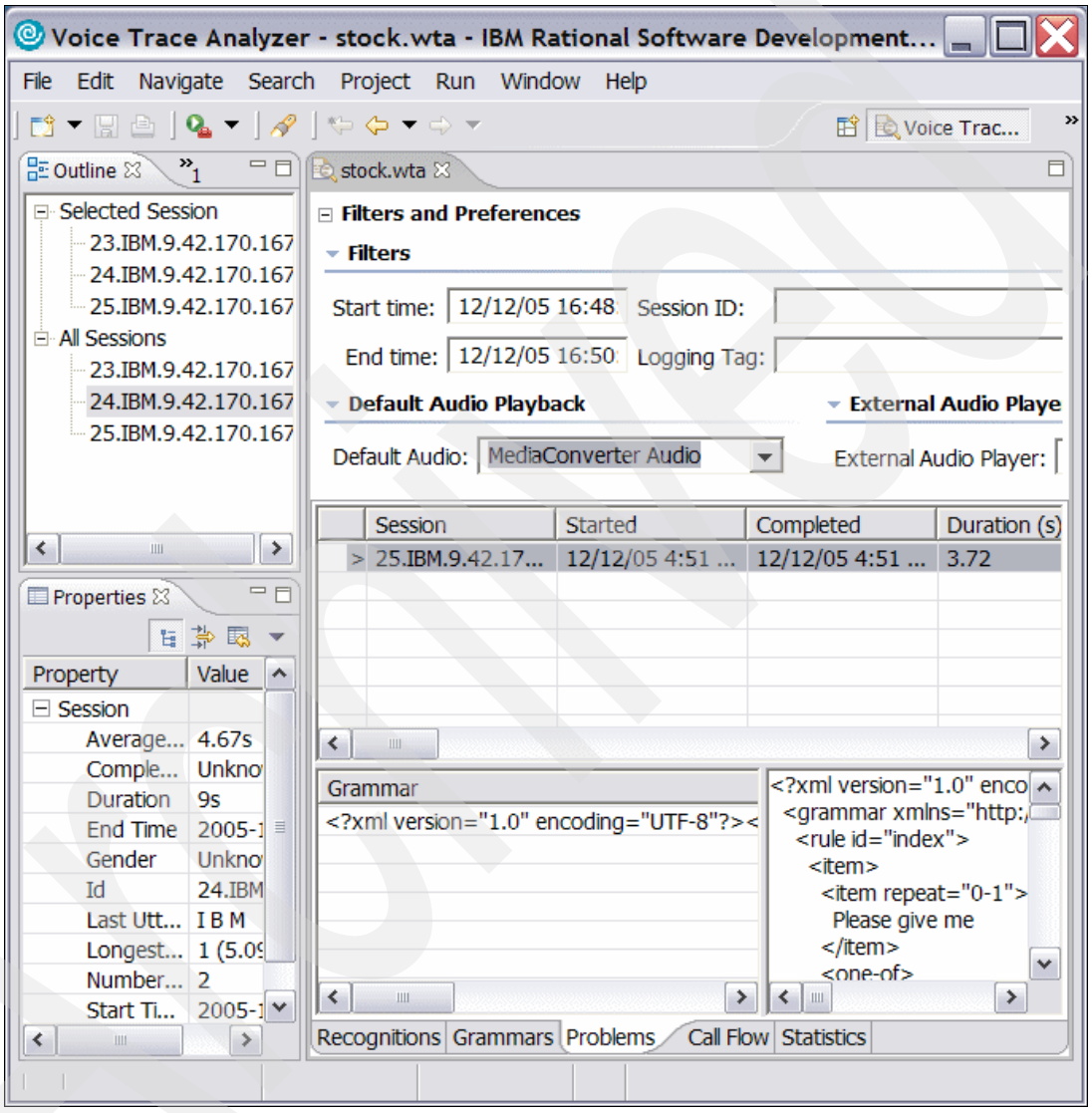



*Figure 7-16 Problems tab*

## 7.10 Call Flow tab

The Call Flow tab (see Figure 7-17) from the Editor View shows a graphical representation of each session.

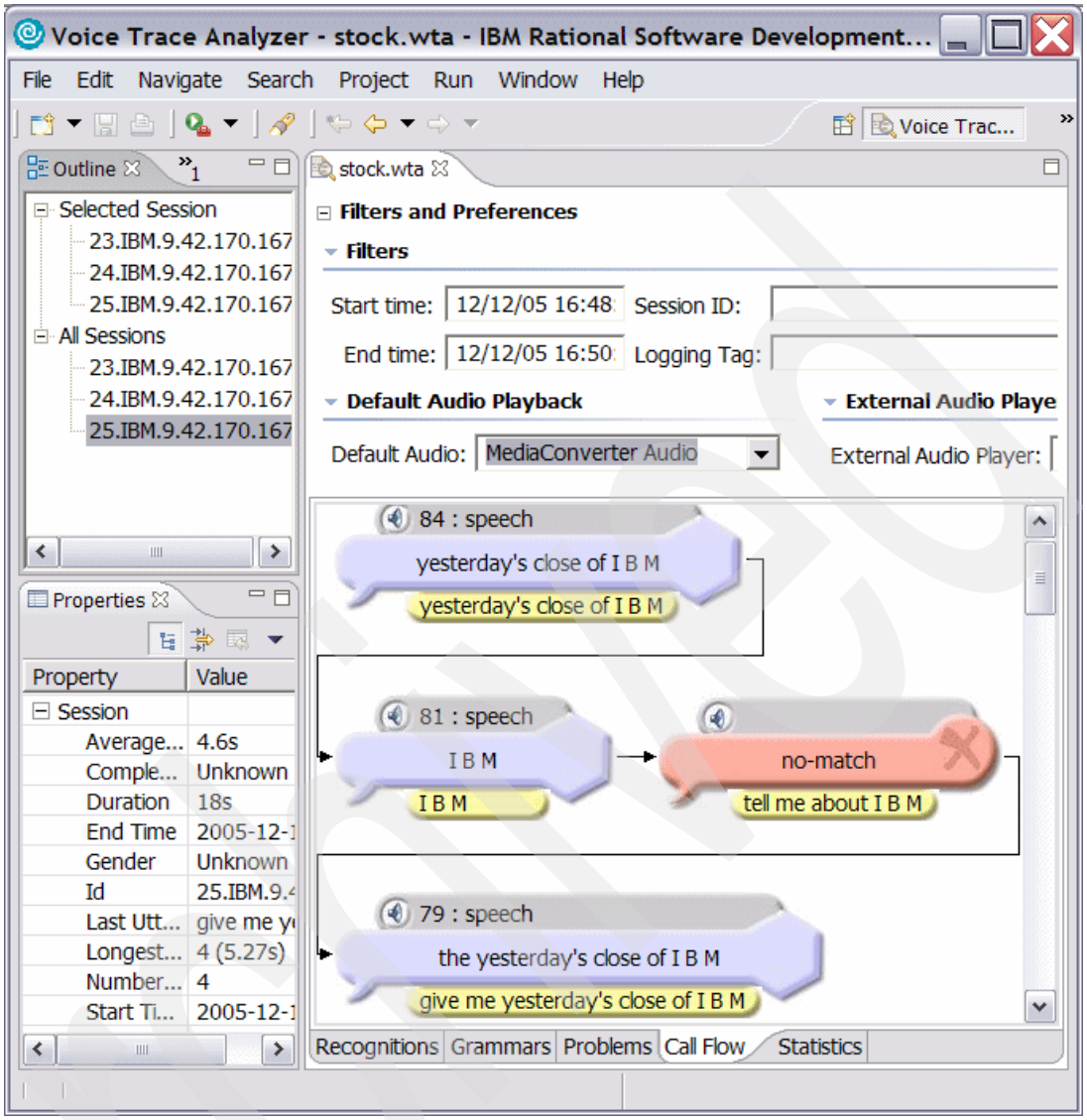



*Figure 7-17 Call Flow tab*

The Call Flow page shows a graphical representation of the call flow. Each figure can have as as many as three parts, but must have at least two. The top part shows the score and confidence of the recognition and the mode in which the input was received (speech or DTMF). An audio icon in the top figure indicates the presence of a recorded utterance. The middle figure shows the input that the server reported. The bottom figure shows the transcription you entered. If no transcription is available, the figure is not shown. You can also enter a transcription from this page.

In Figure 7-17, we selected the third session in the Outline View to show match and no-match conditions.

## 7.11 Statistics tab

This Statistics tab shows basic statistical information, divided into four separate sections.

- Statistics

  This section shows:

  - Total number of detected sessions.
  - Number of detected recognitions.
  - Session having the longest duration, and its duration time.
  - The average session length.
  - The average number of turns per session.
  - This information does not change.

- Accuracy

  This column reflects statistical information derived from the transcription and completion information you provided. Select **Refresh** to update this column.

  **Note one:** In order to update the Statistics tab, you must compute the Accuracy column on the Recognitions tab. This means you must complete both the In Grammar and Transcription columns. Click **Compute Accuracy** to compute the Accuracy column.

  **Note two:** In the properties view,you subjectively define the Completion State for each session.

- Gender

  This section reflects statistical information derived from the gender information you have provided. For each session, you can indicate the gender of the caller as male, female, silence, or unknown. All sessions have a default gender of unknown, until the user changes it. The four fields of this session reflect the respective counts of these values.

- Confidence

  This section has one field that represents the average confidence of all recognitions; this value never changes. A menu list allows you to select a threshold value between 1 and 100. When this value changes, this updates two other fields. The other two fields show the number of recognitions with scores above and below the value specified.

**Tip:** To see the updated Accuracy information displayed on the Statistics tab page, select the Refresh button on the page under the Statistics tab. Refer to 7.8, “Transcription Within Tool” on page 83.

8

# Reusable Dialog Components

In this chapter, we discuss the following Reusable Dialog Components (RDC) topics:

- A short overview explaining Reusable Dialog Components
- RDC usage in the Voice Toolkit
- How to customize RDC
- How to write custom RDC

# 8.1 How to use RDC

Initial development of the first telephony applications using Automatic Speech Recognition (ASR) and Text to Speech (TTS) was extremely complicated and difficult, because each speech vendor had its own set of APIs. It was almost impossible to use a platform from any other vendor than the original speech platform vendor used during development.

The standardization of VoiceXML has made voice application development somewhat easier. You no longer must depend on staying with your speech platform vendor, because now VoiceXML is completely XML-based. Also, you can now use your existing Web infrastructure for serving these VoiceXML applications to the voice browsers.

To speech enable existing Web applications, there is usually a set of JSPs and servlets added to wrap the needed functionality into VoiceXML. However, in terms of reusability of the VoiceXML code and separation of the roles of the different developers, there was still room for improvement.

The advantage of RDC is that it fully adheres to the J2EE programming model. RDCs offer a JSP tag library that application developers can include in their JSP pages. The tag library consists of a set of predefined input objects that application developers can easily include in JSP pages. Also, the tag library is responsible for the necessary dialog management strategy. The use of RDCs allows easy reuse of preconfigured customizable input objects in the J2EE environment, as well as the clear separation of roles during development of a voice application. The Web developer takes care of the proper back-end access to enterprise data. The speech developer is responsible for tuning the used recognition grammars, pronunciations, and for quality of the synthesized prompts. The voice user interface designer gives the application the proper *hear and feel*.

## 8.1.1 Types of RDCs

There are two types of RDCs:

- Atomic RDC
- Composite RDC

The atomic RDC is the basic RDC, which provides the capability to enter digit strings or numbers, for example. In most cases, the atomic RDC consists of only one input field. The composite RDC consists of several other RDCs, which could be either atomic or composite RDCs. A composite RDC, for example, can be a RDC for a money transfer, which consists of a digit RDC for the account number and a number RDC for the amount of money to transfer.

# 8.2 Use existing RDCs in the Voice Toolkit

**Important:** Since RDCs depend on JSP V2.0, you must update the WebSphere Application Server V6.0 runtime that comes with Rational Application Developer to the latest level in order to test RDCs with the Voice Toolkit.

The latest version of the RDC taglib is always available from Apache Jakarta Projects site:

`http://jakarta.apache.org/site/downloads/downloads_taglibs-rdc.cgi`

Here, you can find the source code for the Jakarta Taglibs (which now include the RDCs), as well as the binary distribution of the RDC which resides in the nightly builds section.

### 8.2.1 Set up a dynamic Web application to support RDC

To use the RDC, you need a dynamic Web project within which to deploy the RDC.

The easiest way to get a dynamic Web project that has all the libraries needed by the RDC taglib is to use the project wizard **File** → **New** → **Project**. In the following dialog:

1. You must choose **Web** → **Dynamic Web Project**. After selecting a name for the project and the type of the application server to which to deploy the application, click **Next** to proceed to the dialog shown in Figure 8-1.
2. In this dialog, you create the settings for the dynamic Web project. The RDC requires the **JSP Standard Tag Library** to function properly. In addition, the developer can also choose the **Struts** library, if the developer wants to make use of the Struts features. When you complete the settings, click **Finish** to create a new dynamic Web project in the workspace.

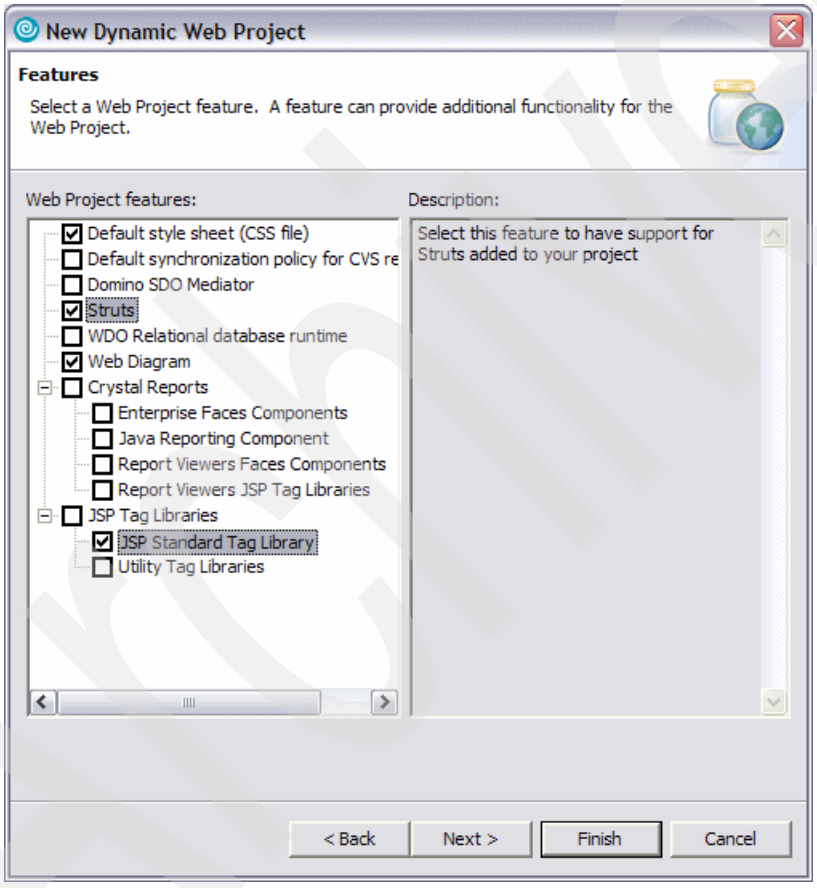



*Figure 8-1 Selecting the features of a dynamic Web project*

3. After you create the new project, copy the RDC taglib into the new Web project.

   The binary distribution of the RDC from Jakarta contains the file taglibs-rdc.tld, which goes into the /WebContent/WEB_INF directory. You also need to copy the file taglibs-rdc.jar into /WebContent/WEB-INF/lib directory.

4. Then you need to load the /WebContent/WEB_INF/web.xml file into the editor.

   Click **Servlets** to add a new servlet that comes with the RDC. First, you need to set the name of the new servlet to GrammarServlet. This servlet requires two parameters. Click

**Add** in the Initialization Parameters section to add each of the parameters. The two parameters that you need are shown in Table 8-1.

*Table 8-1 Servlet initialization parameters*

| Name | Value |
|---|---|
| jar | /WEB-INF/lib/taglibs-rdc.jar |
| grammarDirectory | .grammar |

5. In addition, you also need to modify the URL mappings. Delete the default URL mapping GrammarServlet, that is added automatically, and add a new one with the value:

   `/.grammar/*.`

   Then, select **Use existing Servlet class,** and click **Browse** to select the GrammarServlet from the list box.

   The resulting dialog should look similar to Figure 8-2.

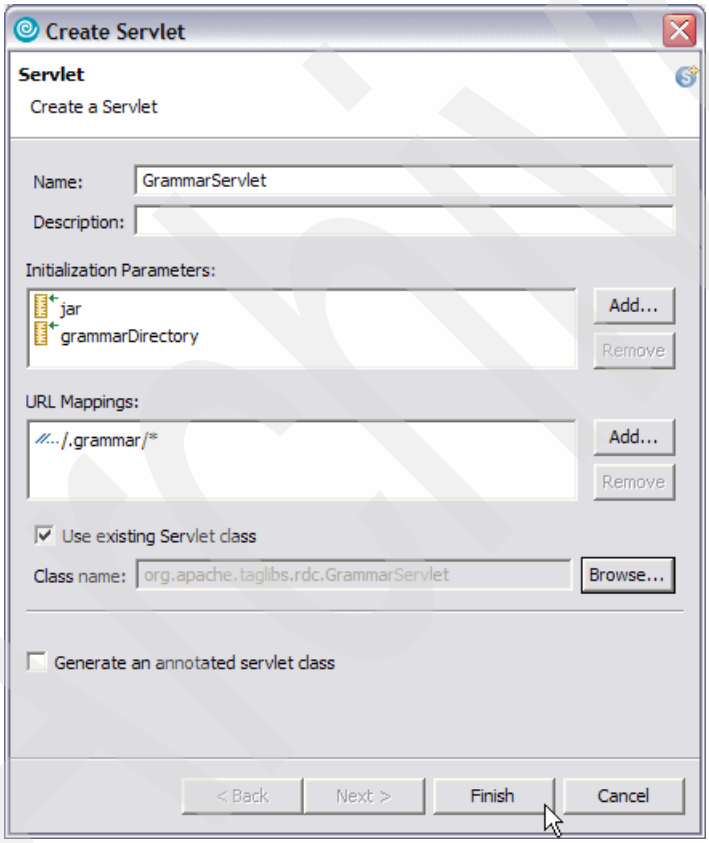



*Figure 8-2 Web.xml settings for the GrammarServlet*

6. If everything is fine, click **Finish** to close the dialog box and save the changed web.xml file.

Now, you can use RDC in your dynamic Web application.

## 8.2.2 Using RDC in the Communication Flow Builder

In order to use RDC in the Communication Flow Builder (CFB), you first need to import the RDC into the CFB.

1. To achieve this, open the context menu in the CFB by right-clicking on the background of the workspace in the CFB perspective. Then, select **Import** from the menu as shown in Figure 8-3.

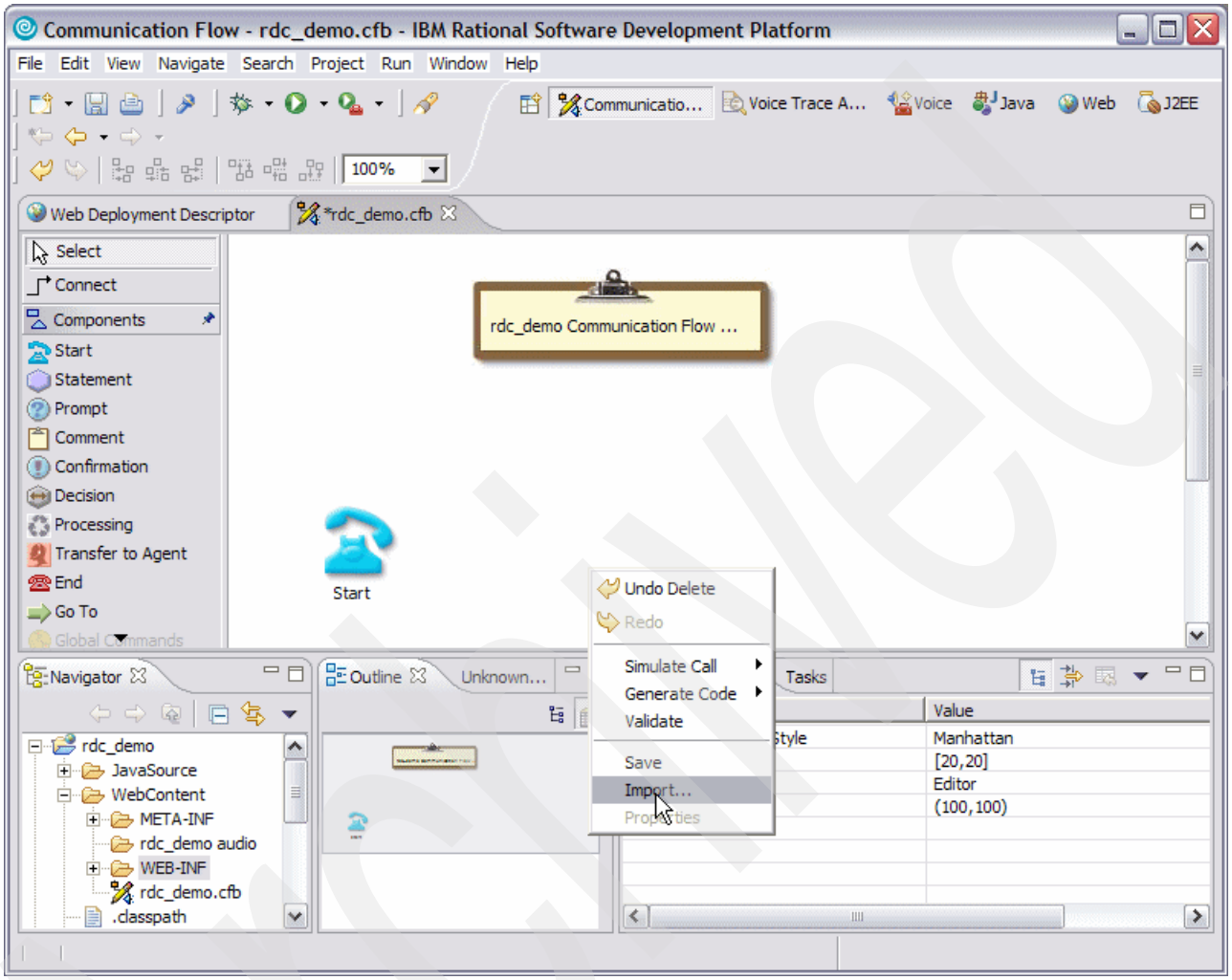



*Figure 8-3 Importing an RDC library into the Communication Flow Builder*

In the dialog window that opens, select the JAR file which contains the RDC as shown in Figure 8-4.

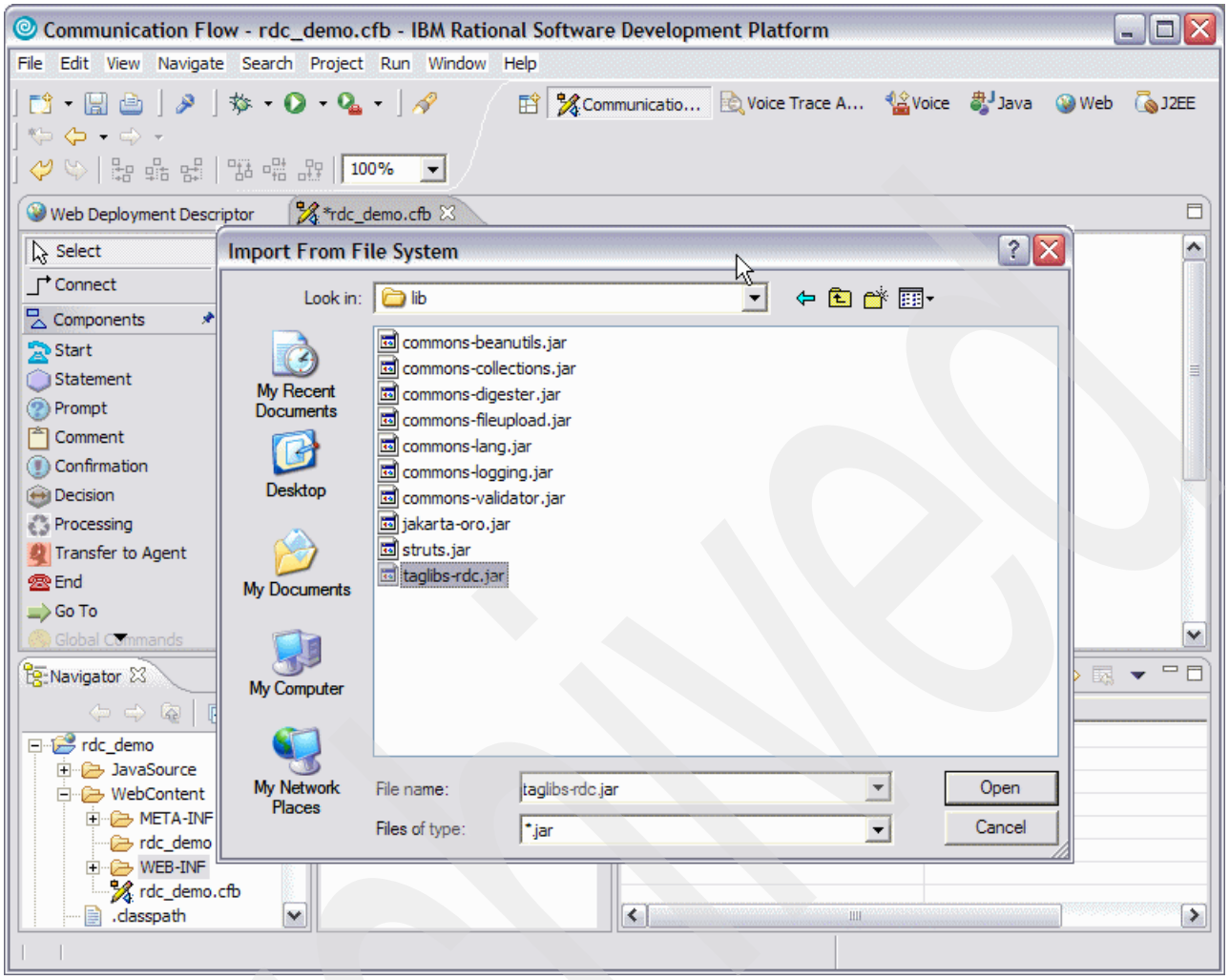



*Figure 8-4 Choosing the taglibs-rdc.jar to import*

Importing the JAR file creates a new folder in /WebContent with the name of the currently imported taglib that contained the RDC, which in our case is taglibs-rdc. This newly created folder contains a set of .rdc files. Each .rdc file represents one of the RDCs, which is contained in the JAR file (for example, digits.rdc is the RDC for digits).

2. Once you have imported the RDC taglib, use drag and drop to add a new RDC from the list into the Communication Flow Builder.

   When you have dropped an RDC into the Communication Flow Builder, a dialog as shown in Figure 8-5 appears.

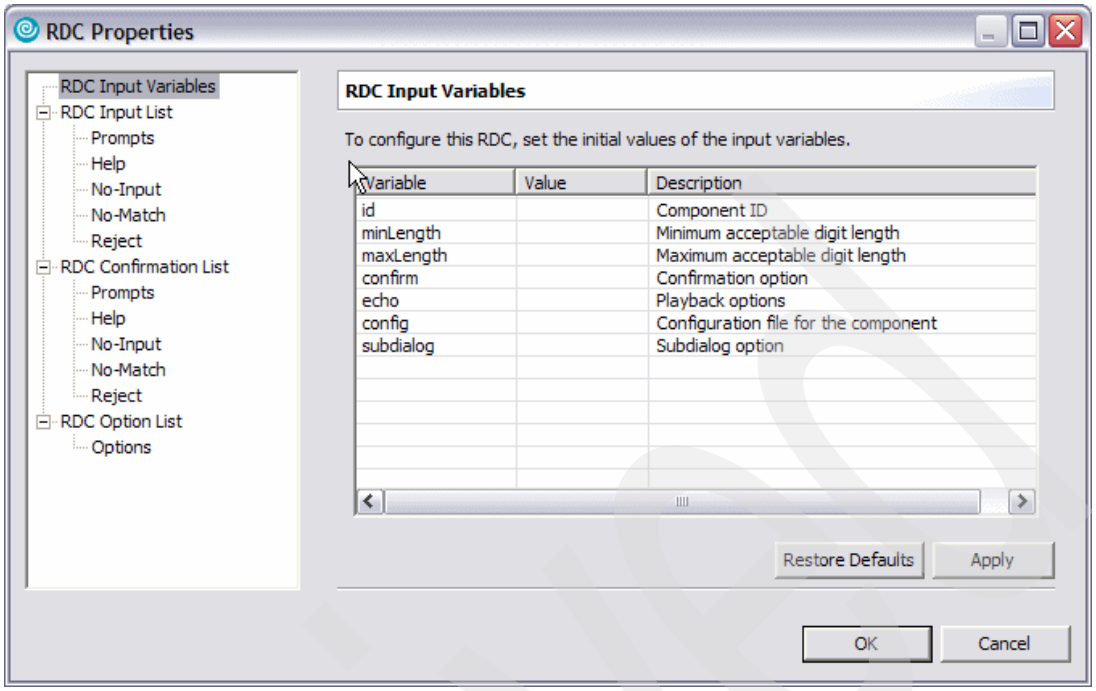



*Figure 8-5 Setting the RDC properties in the Voice Toolkit*

3. Here, you can set the prompts that RDC uses, as well as other parameters, such as the length of the accepted digit string for the digit.rdc, or the maximum number of No-Matches that are allowed before a particular action takes place.

   The newly added RDC is then available in the Communication Flow Builder, and you can link it to other elements of the Communication Flow Builder.

4. After you have set all the necessary dialog objects properly and connected them with branches, you can generate a JSP file to use in the Web application to generate VoiceXML.

**Important:** The configuration that the CFB currently produces is incompatible with the RDC; therefore, in Example A-6, you find an XSL stylesheet to convert the configuration files to the proper format.

## 8.2.3 Customizing existing RDC

After generating code with the CFB, and especially if you made any changes to the generated code, we do not recommend that you to change any properties for the behavior of the RDC in the CFB. *A new code export overwrites all manual changes*. Therefore, you *must* modify the config files of your RDC manually. The config files generated by CFB typically reside in the directory /WebContent/RDCConfigurationFiles. The naming scheme for the config files is *cfb-name.component-id*-Config.xml. Example 8-1 is an example of the config file for a digit RDC.

*Example 8-1 Config file for a digit RDC*

```
<?xml version="1.0" encoding="utf-8"?>
<config>
   <input>
      <property-list>
         <property name="incompletetimeout" value="1s" />
```

```
      <property name="completetimeout" value="1s" />
    </property-list>
    <prompt-list>
      <prompt>
        <audio
          src="rdc_demo%20audio/R00010_input_prompt_1.au">
          Please specify the digits.
        </audio>
      </prompt>
    </prompt-list>
    <help-list>
      <prompt>
        <audio src="rdc_demo%20audio/R00010_input_help_1.au">
          You need to specify digits like 1 2 3 4.
        </audio>
      </prompt>
    </help-list>
    <noinput-list>
      <noinput count="1">
        <prompt>
          <audio
            src="rdc_demo%20audio/R00010_input_noinput_1.au">
            I did not hear you speak the digits.
          </audio>
        </prompt>
      </noinput>
      <noinput count="2">
        <prompt>
          <audio
            src="rdc_demo%20audio/R00010_input_noinput_2.au">
            Could you please repeat the digits?
          </audio>
        </prompt>
      </noinput>
      <noinput count="3">
        <prompt>
          <audio
            src="rdc_demo%20audio/R00010_input_noinput_3.au">
            I appear to be having trouble hearing you.
            Waiting for you to speak the digits.
          </audio>
        </prompt>
      </noinput>
    </noinput-list>
    <nomatch-list>
      <nomatch>
        <prompt>
          <audio
            src="rdc_demo%20audio/R00010_input_nomatch_1.au">
            I am sorry. I didn't understand you. Please
            repeat the digits.
          </audio>
        </prompt>
      </nomatch>
    </nomatch-list>
  </input>
  <confirm>
    <property-list>
      <property name="incompletetimeout" value="1s" />
      <property name="completetimeout" value="1s" />
```

```
    </property-list>
    <prompt-list>
       <prompt>
          <audio
             src="rdc_demo%20audio/R00010_confirm_prompt_1.au">
             I think you said #{model.utterance}. Is that right?
          </audio>
       </prompt>
    </prompt-list>
    <help-list>
       <help>
          <prompt>
             <audio
                src="rdc_demo%20audio/R00010_confirm_help_1.au">
                To accept digits, say yes. To change your mind,
                say no.
             </audio>
          </prompt>
       </help>
    </help-list>
    <noinput-list>
       <noinput>
       <prompt>
          <audio
             src="rdc_demo%20audio/R00010_confirm_noinput_1.au">
             Is #{model.utterance} acceptable ? Please say yes or
             no.
          </audio>
       </prompt>
       </noinput>
    </noinput-list>
    <nomatch-list>
       <nomatch>
       <prompt>
          <audio
             src="rdc_demo%20audio/R00010_confirm_nomatch_1.au">
             If #{model.utterance} is acceptable, say yes.
             Otherwise, say no.
          </audio>
       </prompt>
       </nomatch>
    </nomatch-list>
    <reject>
       <prompt>
          <audio
             src="rdc_demo%20audio/R00010_confirm_reject_1.au">
             OK, lets try again.
          </audio>
       </prompt>
    </reject>
 </confirm>
 <validate>
    <handler count="1">
       <prompt>
          I'm sorry, but there is no default or initial value
          available.
       </prompt>
    </handler>
    <handler count="2">
       <prompt>Please specify shorter digits</prompt>
```

```
      </handler>
      <handler count="3">
         <prompt>Please specify longer digits</prompt>
      </handler>
      <handler count="4">
         <prompt>That is an invalid digit.</prompt>
      </handler>
   </validate>
   <echo>
      <property-list>
         <property name="universals" value="all" />
      </property-list>
      <prompt-list>
         <prompt>OK, #{model.utterance}. Got it.</prompt>
      </prompt-list>
   </echo>
</config>
```

To modify any of your prompts, simply go to one of the four sections:

- Input
- Confirm
- Validate
- Echo

There, you find sections for the typical prompt, but, you also find sections for the help, No-Input, and the No-Match events. In addition, you can set the properties and also additional event handlers for each section.

You can choose to customize different prompts in your RDC when you add it into the CFB. In order to do this, you must modify the .rdc files, which are imported in the taglibs-rdc directory.

The parameters in the component section of the .rdc file are predominantly for the GUI configuration of the RDC, but the parameters in the config section are almost the same as those used in the RDC config file. Therefore, if you want your own set of customized RDCs to use in CFB, modify the existing .rdc files according to your needs, and then, replace each of the imported RDCs with your modified RDCs.

## 8.3 Develop your own RDC

An RDC consists of the following components:

- A tag file that is referenced when you use the JSP tag with the RDC name
- A config file that contains the VoiceXML properties, prompts, and event handlers
- A Java bean which represents the data model of the RDC
- Grammars for speech input and DTMF input

There are two ways to develop a new RDC:

- Add the new RDC into the Jakarta taglibs project.
- Write a completely separate JSP taglib that uses the RDC taglib.

### 8.3.1 Compiling the Jakarta taglibs

You need an installation of Jakarta Tomcat V5.x in order to compile the Jakarta taglibs RDC, because the taglibs are dependent on some of the Tomcat libraries. All other libraries come with a Rational Application Developer installation; therefore, the easiest way to get those

libraries is to create a new dynamic Web application and get the required libraries from the /WebContent/WEB-INF/lib directory.

To compile the taglibs in Rational Application Developer, start with an empty project and extract the jarkarta-taglibs archive into that directory. The resulting window looks like Figure 8-6.

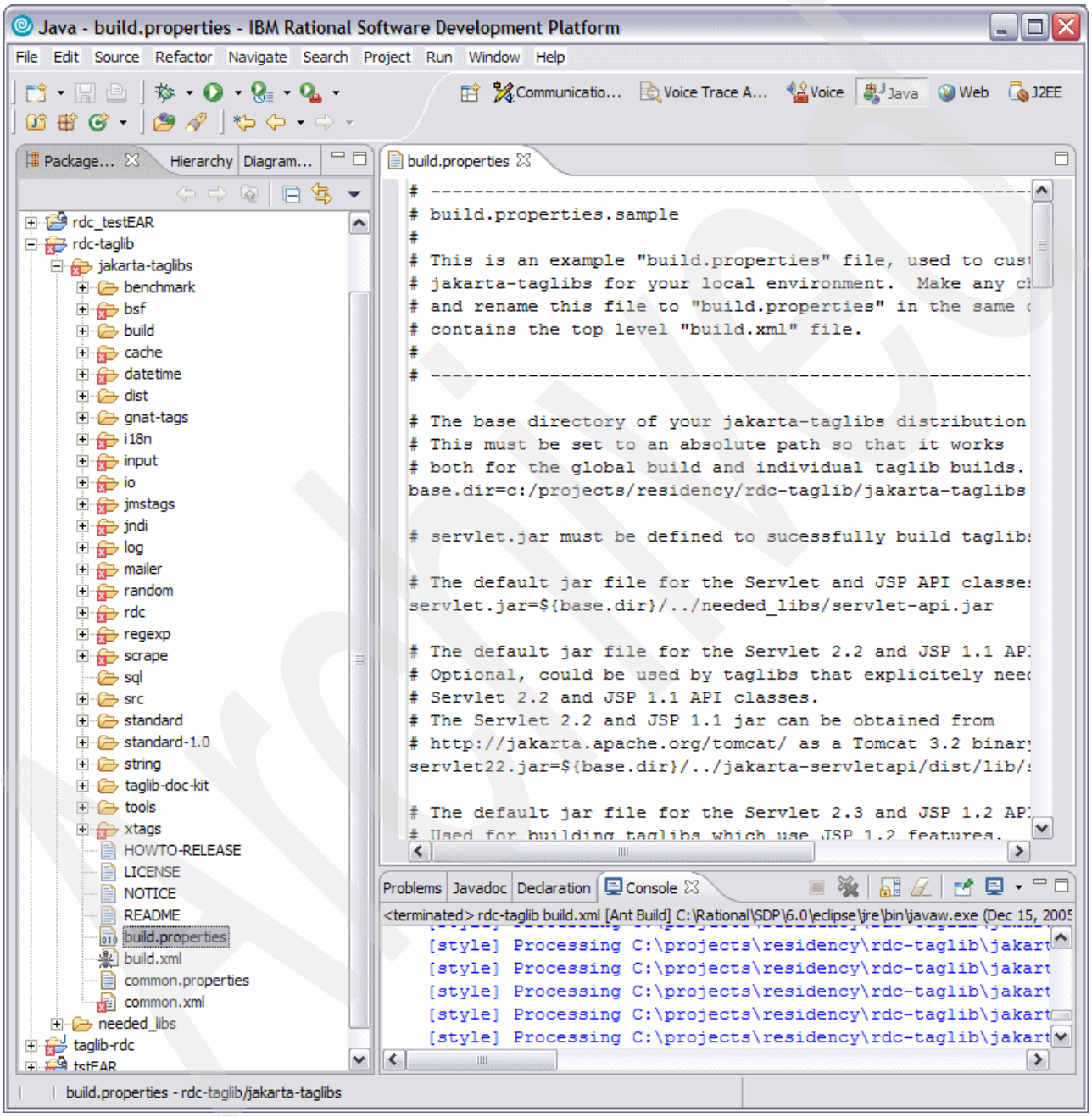



*Figure 8-6 RDC taglibs project after importing the sources*

You must rename the build.properties.sample, which is located in the top level directory, to build.properties. After that, you must set the proper paths to the necessary libraries in build.properties, as shown in Example 8-2 on page 100, where only the changes to build.properties show.

*Example 8-2 Sample build.properties*

```
# ----------------------------------------------------------------------
# build.properties.sample
## This is an example "build.properties" file, used to customize building
# jakarta-taglibs for your local environment.  Make any changes you need,
# and rename this file to "build.properties" in the same directory that
# contains the top level "build.xml" file.
#
# ----------------------------------------------------------------------

# The base directory of your jakarta-taglibs distribution
# This must be set to an absolute path so that it works
# both for the global build and individual taglib builds.
base.dir=c:/projects/residency/rdc-taglib/jakarta-taglibs

# servlet.jar must be defined to sucessfully build taglibs

.
.
.

# The default jar file for the Servlet 2.4 and JSP 2.0 API classes.
# Used for building taglibs which use JSP 2.0 features.
# The Servlet 2.4 and JSP 2.0 jar can be obtained from
# http://jakarta.apache.org/tomcat/ as a Tomcat 5.x binary download.
servlet24.jar=${base.dir}/../needed_libs/servlet-api.jar

jsp-api.jar=${base.dir}/../needed_libs/jsp-api.jar
jsp20.jar=${base.dir}/../needed_libs/jsp-api.jar
jstl.jar=${base.dir}/dist/standard/lib/jstl.jar

.
.
.

# jmstags requires the commons-digester API
commons-digester.jar=${base.dir}/../needed_libs/commons-digester.jar

# jmstags requires the commons-beanutils API
commons-beanutils.jar=${base.dir}/../needed_libs/commons-beanutils.jar

.
.
.

# RDC requires the commons-el API
commons-el.jar=${base.dir}/../needed_libs/commons-el.jar

# RDC requires the commons-logging API
commons-logging.jar=${base.dir}/../needed_libs/commons-logging.jar

# Taglibs such as the RDC Taglib and Cache Taglib require
# the Standard Taglib's standard.jar distribution
# You can obtain the Jakarta Taglibs Standard distribution from
# http://jakarta.apache.org/taglibs/
standard.jar=${base.dir}/dist/standard/lib/standard.jar

# The RDC Taglib requires Struts 1.2.x for sample applications
# You can download Struts 1.2 from
# http://struts.apache.org/
```

```
struts12.jar=${base.dir}/../needed_libs/struts.jar
.
.
.
#   build.dir          Base directory for build targets
#   dist.dir           Base directory for distribution targets
build.dir = ${base.dir}/build
dist.dir = ${base.dir}/dist
```

In Example 8-2 on page 100, all the necessary libraries were put into the needed_libs directory, which resides at the same level as the jakarta-taglibs directory.

To build the RDC taglibs, you need the following libraries:

- servlet24.jar is taken from a Tomcat 5.x installation and usually resides in *tomcat_dir*/common/lib/servlet-api.jar
- jsp-api.jar resides in the same directory as servlet-api.jar
- commons-beanutils.jar, commons-el.jar, commons-digester.jar, and commons-logging.jar are available from:

  http://jakarta.apache.org/site/downloads/downloads_commons.html
- struts.jar can be downloaded from the struts Web page:

  http://struts.apache.org
- standard.jar and jstl.jar are bootstrapped from the jakarta-taglibs source.

You need to compile the standard taglib first, in order to compile the RDC taglib. Do this by selecting the context menu of the build.xml file in the standard directory. Select **Run** → **3 Ant Build** as shown in Figure 8-7.

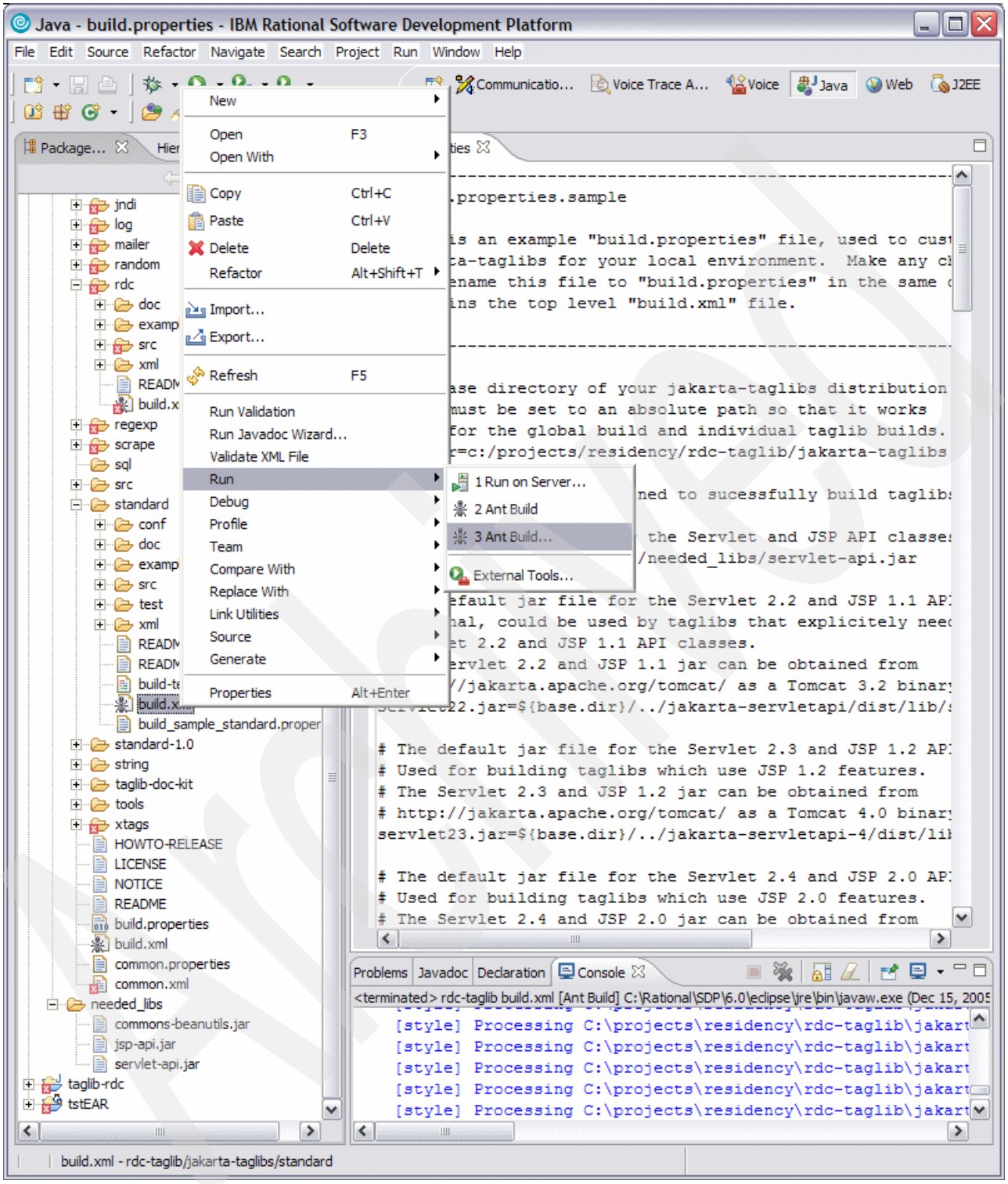



*Figure 8-7 Invoking the build of standard.jar and jstl.jar*

A dialog opens where you can specify additional build information. In the Targets tab, you should select the target dist, in addition to the already selected targets, as shown in Figure 8-8.

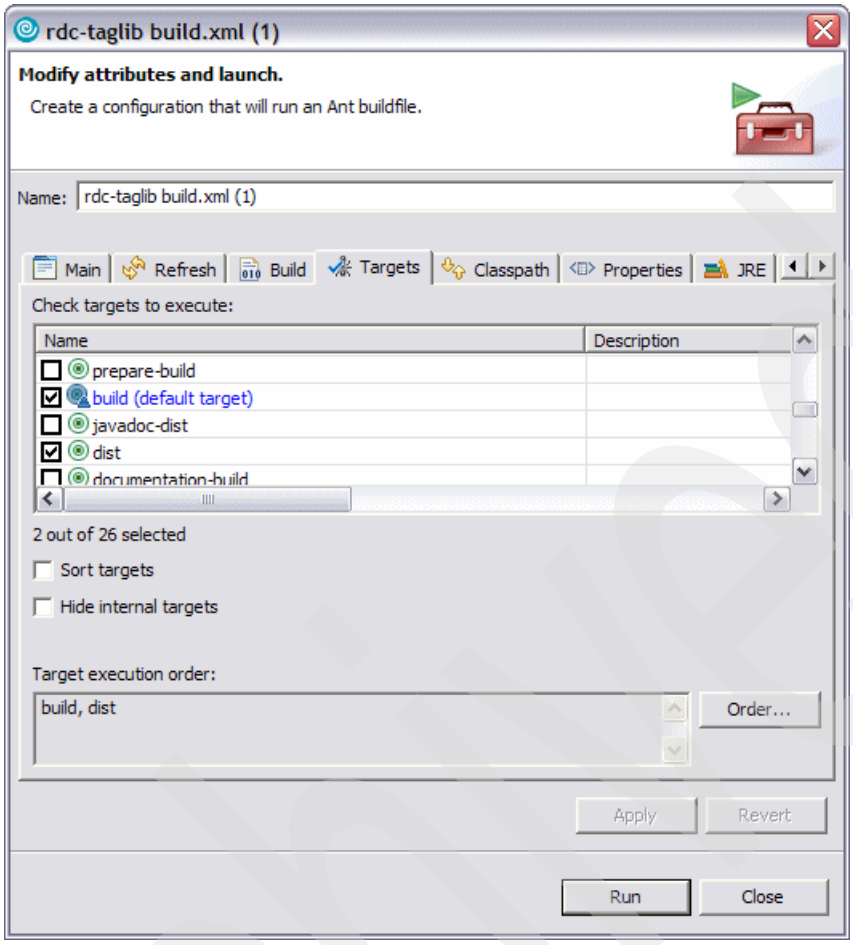



*Figure 8-8 Targets dialog for the standard taglib*

To build the standard taglib, we recommend you use the normal J2SE™ Runtime from Sun, instead of the runtime included with Rational Application Developer. To select a different JDK™ to compile the standard taglib, select **JRE** and choose a proper JDK from the list box, as shown in Figure 8-9. After you complete this, click **Run** for the build process to compile the standard taglib.

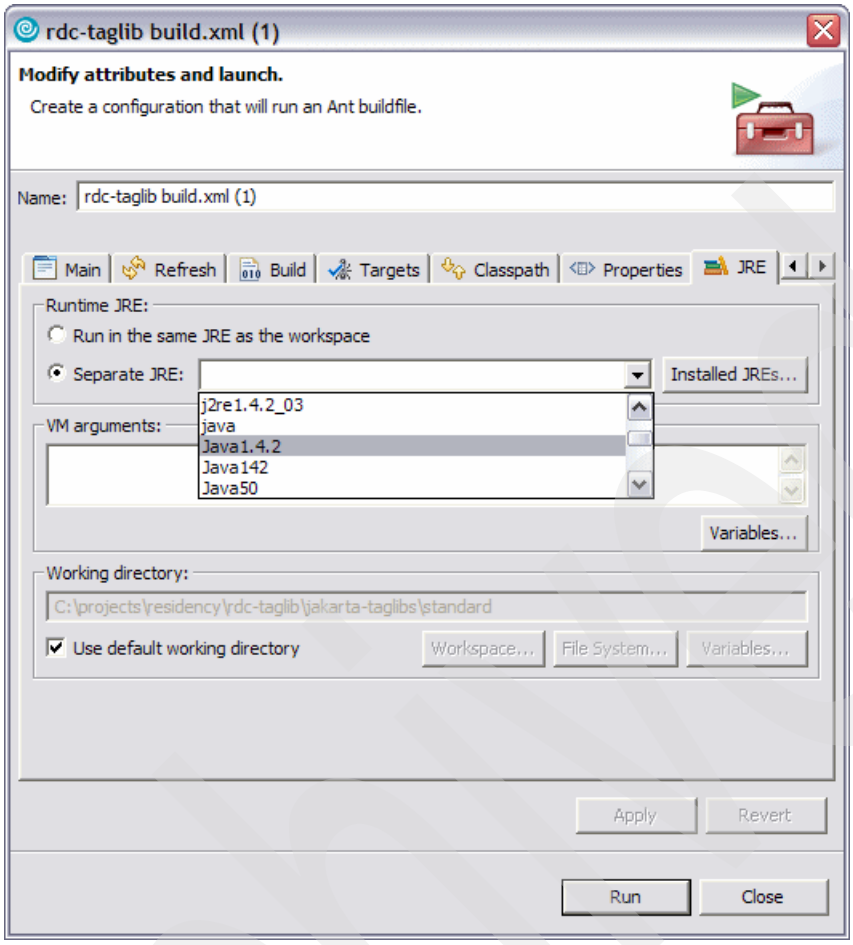



*Figure 8-9 Select a different JDK to build the standard taglib*

When the build process finishes, you find standard.jar and jstl.jar in the directory dist/standard/lib, where they are then picked up to build the RDC taglib. Now, you can start the build process for the RDC taglib, since there are no additional targets necessary for the RDC taglib build.

1. Click **Run** → **2 Ant Build** from the context menu of rdc/build.xml. The build process then produces taglibs-rdc.jar and taglibs-rdc.tld in dist/rdc.

You can now use these files as described in 8.2.1, “Set up a dynamic Web application to support RDC” on page 91.

Since it is now possible to compile the RDC taglib directly in Rational Application Developer, the next section shows you how to add support for other languages in the existing RDC.

## Add support for other languages into the RDC taglib

To add support for a new language to an RDC, you need the following things:

- Localized grammar files
- A localized config files with the default prompts
- A localized RDC file that can be dropped in the CFB
- Adding or modifying the language properties file for the RDC

The grammar files for all RDCs are located in rdc/src/.grammar. If you want to add support for a new language, these grammars should go into a subdirectory beneath the grammar

directory (for example, rdc/src/.grammar/de-DE for German or rdc/src/.grammar/fr-CA for Canadian French). The non en-US grammar file names should have the same file name as the en-US grammar file name but will be located in the respective language subdirectory.

You should apply the same procedure to the config files, which normally reside in rdc/src/META-INF/tags/rdc/config. Therefore, the German config files should go into rdc/src/META-INF/tags/rdc/config/de-DE, and Canadian French config files reside in rdc/src/META-INF/tags/rdc/config/fr-CA. The non en-US config file names should have the same file name as the en-US config file name but will be located in the respective language subdirectory.

For the RDC files, a different approach is necessary because the CFB needs the RDC files. The CFB needs the RDC files to reside in rdc/src/META-INF/tags. To add language support, code the language into the filename (for example, the digit.rdc file is the en-US file for recognition of digit strings). For example, a localized RDC for Castilian Spanish would have the name digits_es-ES.rdc and the Italian RDC for the input of numbers should have the name number_it-IT.rdc.

After all the resource files are at their proper places, you need property files for each language. The property files must be located in rdc/src/org/apache/taglibs/rdc/resources. If there is already a generic property file named RDCBundle.properties and one for en_US named RDCBundle_en_US.properties, the property file for the new resource bundle has to follow the standard naming conventions for resource bundles. Therefore, a German resource bundle must be named RDCBundle_de_DE.properties, and a Dutch resource bundle would be named RDCBundel_nl_NL.properties.

## 8.4 Submit new RDCs to the Jakarta taglibs

The RDCs from Jakarta are a open source project, and, therefore, they benefit from contributions from developers to the project. Contributions do not need to be completely new RDCs. Contributions can be fixes to “bugs”, patches, and enhancements. The open source project developers appreciate localization of existing RDCs into different languages.

To contribute to the Jakarta taglibs project, submit your patches and enhancements to Apache’s bugzilla, which you can access through the following Web site:

```
http://issues.apache.org/bugzilla
```

9

# Tuning voice applications

This chapter describes some tuning methods available in WebSphere Voice Server and the Voice Toolkit. The tuning is part of the iterative process defined in 1.3, “SUI Design methodology” on page 4.

Application developers should try these methods in a defined order. The following sections follow a typical tuning progression, although application developers can switch some parts off or skip parts, depending on the application peculiarities.

In any case, developers must first use the default values, keep a record of the application behavior, and then play with parameters in order to enhance the results. In this chapter, we assume that application developers followed the best practice guidelines in 1.4, “Best practices” on page 5.

# 9.1 Modifying prompts to improve recognition

The way you formulate the prompt request may impact the user answer. We observed in 1.4.1, “Prompt design” on page 5 how we can encourage the user to say “yes” or “no” instead of alternatives, such as “yes I would”.

Modifying prompts in the way the request is phrased can avoid speech recognition errors or user misunderstanding.

## 9.1.1 Timeout

The *timeout attribute* specifies the interval of silence allowed after the end of the last prompt while waiting for the user input. (It is a different timeout than the speech recognizer timeouts that we discuss in 9.5.2, “Timeouts” on page 114).

If the user exceeds this interval, the platform will throw a No-Input event.

The reason for allowing you to specify timeouts as prompt attributes is to support *tapered* timeouts. For example, the user may get five seconds for the first input attempt, and ten seconds for the next input attempt.

# 9.2 Active grammars

The first thing to do when investigating a speech recognition behavior is to understand which grammars are active.

VoiceXML browsers have three always active grammars: cancel, exit, and help. In addition to those, your own grammar must be active, but you may also have other active grammars, which may interfere.

Generally speaking, you should determine which grammars are active at a given time. You must enable or disable them carefully. Reducing the number of active grammars to the minimum amount can lead to better performance. The more active grammars you have, the more valid utterances you get, and the more potential acoustic confusabilities you generate. This is especially true when you handle multiple large grammars.

The Voice Trace Analyzer (see 7.7, “Grammars tab” on page 82) tells you which grammar was active.

In Figure 9-1, you can see the three always active grammars and the two built-in (DTMF and speech) grammars.

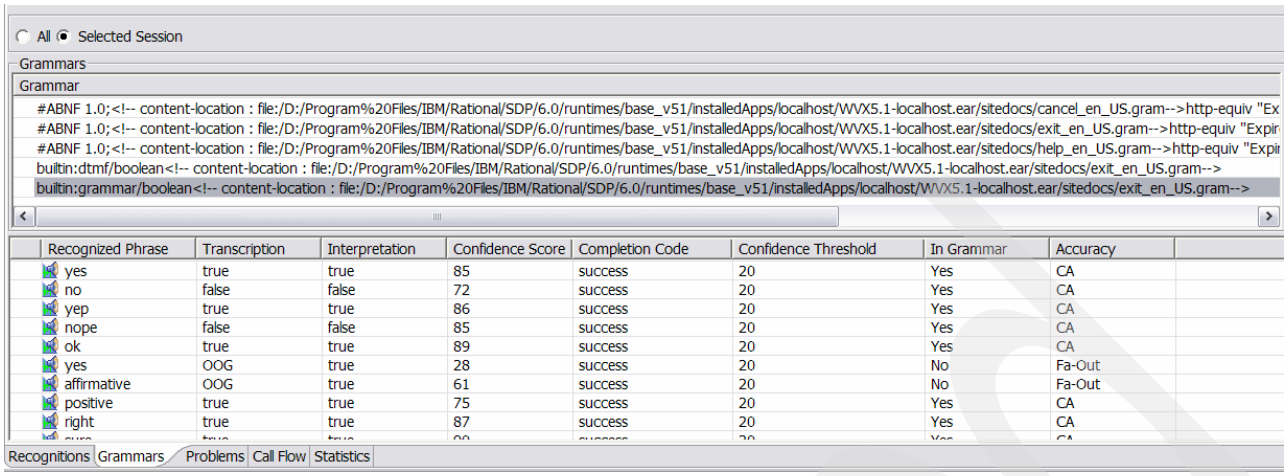



*Figure 9-1 You can check active grammars with the Voice Trace Analyzer*

You should exercise care to avoid using the same word or phrase in multiple grammars that can be active concurrently; the VoiceXML browser resolves any ambiguities by using the first matching value according to the grammar hierarchy defined in the VoiceXML Programmer's Guide "Hierarchy of active grammars".

You can temporarily disable active grammars, including the VoiceXML browser's grammars for built-in commands, by using the modal attribute on various form items. When the modal attribute is set to true, this disables all grammars temporarily, except the grammar for the current form item.

## 9.3 Using lexicons for pronunciations

In 6.4, "Add words and pronunciations with Pronunciation Builder" on page 61, we describe how to fill in a lexicon file. We gave the example of a names.lxml where we added our first and last names with the default pronunciation (see Example 6-2 on page 62). For better performance, now we try to enrich these pronunciations with legitimate alternate pronunciations, as shown in Example 9-1.

*Example 9-1 names2.lxml: Modified version of names.lxml with alternate pronunciations*

```
<?xml version="1.0" encoding ="UTF-8"?>
<!DOCTYPE lexicon PUBLIC "-//com/ibm/speech/grammar/lexicon//DTD Lexicon 1.0//EN"
"ibmlexiconml.dtd">

<lexicon version="1.0" xml:lang="en-US" alphabet="x-ibmasr" case-sensitive="false">
   <lexeme>
      <spelling>gary elliott</spelling>
      <phoneme>G EY R IY   EH L IY IX TD</phoneme>
   </lexeme>
   <lexeme>
      <spelling>james chamberlain</spelling>
      <phoneme>JH EY M Z   CH EY M B AXR L IX N</phoneme>
   </lexeme>
   <lexeme>
      <spelling>markus klehr</spelling>
      <phoneme>M AA R K AX S   K L EH R</phoneme>
      <phoneme>M AA R K UW S   K L EH R</phoneme>
      <phoneme>M AA R K UW S   K L EY R</phoneme>
      <phoneme>M AA R K AX S   K L EY R</phoneme>
```

```
   </lexeme>
   <lexeme>
      <spelling>jerome baude</spelling>
      <phoneme>JH AX R OW M   B AO DD</phoneme>
      <phoneme>JH AX R OW M   B OW DD</phoneme>
   </lexeme>
</lexicon>
```

Adding new pronunciations does not always provide improvements. Indeed, while adding new pronunciations covers a wider range of inputs, it can also introduce confusion. In some cases, reducing the number of alternates helps. This tuning requires expertise. In the end, you will certainly want to test your lexicon together with your grammar using the Test Grammar on MRCP defined in Chapter 5, “Testing Grammars on MRCP” on page 45.

## 9.4 Weighting grammars

Weighting alternatives of your grammars can improve the overall system performance.

*Weights* are simple positive floating point values without exponentials. Legal formats are "n", "n.", ".n", and "n.n" where "n" is a sequence of one or many digits. A *weight* is nominally a multiplying factor in the likelihood domain of a speech recognition search. A weight of 1.0 is equivalent to providing no weight at all. A weight greater than "1.0" positively biases the alternative, and a weight less than "1.0" negatively biases the alternative.

Grammar authors and speech recognizer developers should be aware of the following limitations upon the definition and application of weights as outlined above.

- Guessing weights does not always improve speech recognition performance. In some cases, it can even degrade results.
- The best way to obtain effective weights is to study real speech input to a grammar.
- Adding weights to your grammar will likely modify your confidence score tuning. See 9.5.1, “Confidence levels” on page 111.
- Weight tuning is highly sensitive. You may have to do the tuning again, if you have modified your grammar by adding or deleting alternatives.
- Tuning weights for a particular grammar and recognizer does not guarantee improved recognition performance on other speech recognizers.

A reasonable technique for developing portable weights is to use weights that are correlated with the occurrence counts of a set of alternatives.

A sensible case of study is the boolean grammar. During a confirmation step, most of the people will answer by “Yes” or “No”. Some will say “Yes I do” or “No I do not” and a few will say something like “Yep”, “Nope”, or “Ok”. You may want to include this information into your grammar as shown in Example 9-2.

*Example 9-2 One example of a weighted grammar*

```
<?xml version="1.0" encoding ="iso-8859-1"?>
<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
"http://www.w3.org/TR/speech-grammar/grammar.dtd">
<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar" xml:lang="en-US"
root="main_rule">
<rule id="main_rule" scope="public">
   <ruleref uri="#yes"/>
   <ruleref uri="#no"/>
```

```
</rule>
<rule id="yes" scope="private">
   <one-of>
      <item weight="10">yes</item>
      <item weight="2">yes I do</item>
      <item>yep</item>
      <item>right</item>
      <item>ok</item>
      <item>sure</item>
   </one-of>
</rule>
<rule id="no" scope="private">
   <one-of>
      <item weight="10">no</item>
      <item weight="2">no I do not</item>
      <item>wrong</item>
      <item>nope</item>
      <item></item>
   </one-of>
</rule>
</grammar>
```

## 9.5 Tuning VoiceXML properties

Most of the following parameters are sensitive and they can damage your performance if you do not handle them carefully. If you already have good settings in a WebSphere Voice Server V4.2, refer to the document, *Migrating Applications from IBM WebSphere Voice Server V4.2 to V5.1.x for Use with IBM WebSphere Voice Response*.

http://www.ibm.com/support/docview.wss?rs=761&context=SSKNG6&dc=DA420&dc=DA410&dc=DA440&dc=DA430&uid=swg27006260&loc=en_US&cs=utf-8&lang=en

### 9.5.1 Confidence levels

#### Basics

When decoding a speech signal, the speech recognition engine searches for the best hypothesis according to the loaded grammars. The speech recognition engine returns a set of the best candidates together with their scores, which are derived from the decoding likelihood.

A rejection mechanism rejects the recognition output if the score is below the threshold (default threshold equal 0.5) and sends a No-Match back to the application.

If the score is above the threshold, the speech recognition engine accepts the recognition result and sends it back to the application.

This rejection mechanism is essential for a good speech recognition system. You expect the system to recognize “yes” when you say “yes”. In the meantime, you do not want it to recognize anything if your friend was talking to you when you were about to answer the application.

In the above example, your friend’s speech may confuse the recognition engine which returns some hypothesis. The incoming signal made of your friend’s speech and some background noises should lead to very low scores and reject your friend’s speech.

The score of an incoming speech signal depends on numerous factors. The most intuitive parameter is the speech utteration quality. A speaker swallowing words likely gets a worse score, than a speaker clearly but naturally uttering words. However, the speech utteration quality may not always be the parameter with the greatest impact.

Background noises or channel distortions can severely impact scores. The grammar size can also have an impact in some cases.

The default value is often not optimal.

We consider the five Accuracy values defined in 7.8, “Transcription Within Tool” on page 83 shown in Table 9-1.

*Table 9-1 Accuracy values*

| | **MATCH** | **NO-MATCH** |
|---|---|---|
| **In Grammar** | CA or FA-In* | FR |
| **Out of Grammar** | FA-Out | CR |

* A correct recognition generates a "CA". A false recognition generates a "FA-In"

## Trade-off

If asked to confirm a money transfer from one account to another one, you do not want the recognition engine to understand “yes” if you said “no” or “yes” if you were not talking but your friend was talking in the background. (Note that this behavior does not necessary mean that your system quality is bad, but incorrectly tuned instead).

For a critical task, such as a money transfer, you want to lower the FA-In and FA-Out by increasing the threshold, even if this has the drawback of increasing the FR (see Figure 9-2). You should tune a less critical task differently.

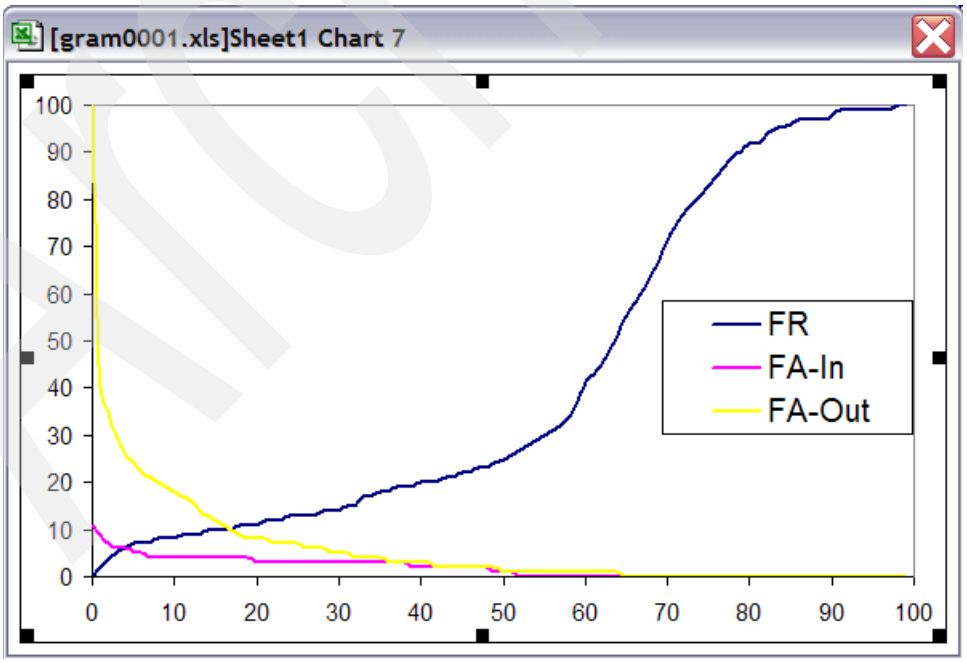



*Figure 9-2 FA-In, FA-Out, and FR curves depending on thresholds*

In Figure 9-2 on page 112, you may want to decrease the threshold to get better performances.

VoiceXML V2.0 provides a way to modify the default threshold on a grammar by grammar basis. VoiceXML scores and thresholds range from 0 to 1 with a step size of 0.1; IBM internal scores vary from 0 to 100.

## Voice Toolkit

The Voice Toolkit can calculate the above accuracy results based on user-supplied transcriptions for each turn (see Chapter 7, “Voice Trace Analyzer” on page 65).

Figure 9-3 displays the recognition tab from the Voice Trace Analyzer. After filling out the Transcription column, the toolkit compares it with the Interpretation column and determines the Accuracy value turn by turn.

| | Transcription | Interpretation | Confidence Score | Completio... | Confiden... | In Grammar | Accuracy |
|---|---|---|---|---|---|---|---|
| > | true | true | 85 | success | 20 | Yes | CA |
| > | false | false | 72 | success | 20 | Yes | CA |
| > | false | | | no-match | 20 | Yes | FR |
| > | true | true | 86 | success | 20 | Yes | CA |
| > | false | false | 85 | success | 20 | Yes | CA |
| > | true | true | 89 | success | 20 | Yes | CA |
| > | OOG | true | 28 | success | 20 | No | Fa-Out |
| > | OOG | true | 61 | success | 20 | No | Fa-Out |
| > | true | true | 75 | success | 20 | Yes | CA |
| > | true | true | 87 | success | 20 | Yes | CA |
| > | true | true | 90 | success | 20 | Yes | CA |
| > | | | | error | 20 | | |
| | true | true | 82 | success | 20 | Yes | CA |
| | false | false | 81 | success | 20 | Yes | CA |
| | true | true | 92 | success | 20 | Yes | CA |
| | true | true | 87 | success | 20 | Yes | CA |
| | OG | false | 32 | success | 20 | No | Fa-Out |
| | OOG | true | 20 | success | 20 | No | Fa-Out |
| | false | | | no-match | 20 | Yes | FR |
| | OOG | true | 28 | success | 20 | No | Fa-Out |
| | true | true | 88 | success | 20 | Yes | CA |
| | true | true | 89 | success | 20 | Yes | CA |
| | true | true | 86 | success | 20 | Yes | CA |
| | false | false | 90 | success | 20 | Yes | CA |
| | false | false | 95 | success | 20 | Yes | CA |
| | false | false | 81 | success | 20 | Yes | CA |
| | | | | error | 20 | | |

*Figure 9-3 CA, CR, FA-In, FA-Out, and FR for a boolean test set*

The choice of a correct threshold determines how frequently the system asks the speaker to repeat, and how tolerant or how severe the system is. For accurate and relevant tunings, conduct this analysis with a representative test set. You may want to play with the parameters in the first phases of the application development. However, you must choose the final threshold in an environment that is as close as possible to the final deployment.

## Advanced algorithm

You may find that the above filtering algorithm is not fully satisfying for your specific application. If so, you may want your system to look at your confidence scores, but also look at the confidence score distance between the first result and the second result of your N-best list.

Indeed, if two results roughly have the same confidence scores, the first one may not be the right one. You may want the user to repeat in such circumstances, or you may want to execute a special dialog to handle difficult cases, such as when you know that two valid entries are highly confusable.

“GetNbestList.vxml” on page 129 is an example that uses the variable application.lastresult$.confidence.

## 9.5.2 Timeouts

Here, we speak about speech recognizer timeouts which are:

- Completetimeout
- Incompletetimeout
- Maxspeechtimeout

They are different from the prompt timeout related to the No-Input event (see 9.2, “Active grammars” on page 108).

- completetimeout:

  This is the length of silence required following user speech before the speech recognizer finalizes a result (either accepting it or throwing a No-Match event). The speech recognizer uses complete timeout when the speech is a complete match of an active grammar. By contrast, the speech recognizer uses the incomplete timeout when the speech is an incomplete match to an active grammar.

- incompletetimeout

  The required length of silence following user speech after which a recognizer finalizes a result. The incomplete timeout applies when the speech prior to the silence is an incomplete match of all active grammars. In this case, once the timeout is triggered, the speech recognizer rejects the partial result (with a No-Match event).

  The incomplete timeout also applies when the speech prior to the silence is a complete match of an active grammar, but where it is possible to speak further and still match the grammar. By contrast, the speech recognizer uses the complete timeout when the speech is a complete match to an active grammar, and no further words can be spoken.

- maxspeechtimeout

  The maximum duration of user speech is maxspeechtimeout. If this time elapses before the user stops speaking, the event "maxspeechtimeout" is thrown.

  Long complete and incomplete timeout values delay the result completion, and, therefore, make the computer's response slow. Short complete and incomplete timeouts may lead to utterances getting broken up inappropriately.

  The incomplete timeout is usually longer than the complete timeout to allow users to pause mid-utterance (for example, to breathe).

Although default values work fine in most cases, you may need to adjust them for your grammars. Listen to the audio files listed on the page under the Recognition tab of the Voice Trace Analyzer, and modify the timeouts if the system encounters systematic errors.

## 9.5.3 Speedvsaccuracy

This is a hint specifying the desired balance between speed and accuracy. Table 9-2 on page 115 defines the various values and their meanings.

*Table 9-2 Speedvsaccuracy values and meanings*

| Value | Meaning |
|---|---|
| 0.0 | Fastest recognition |
| 0.5 | Default |
| 1.0 | Best accuracy |

Unless you use old machines, we recommend to try the highest value 1.0, and reduce the value if you face latency issues.

### 9.5.4 Sensitivity

This parameter sets the sensitivity level of the speech detector. A value of 1.0 means that the speech detector is highly sensitive to quiet input. A value of 0.0 means the speech detector is the least sensitive to noise. The default value is 0.5.

When the prompt is playing is the important issue in speech detection. A false barge-in is considered bad, since the caller cannot hear the response (of the system) to which the caller should confirm or reply.

During prompt play, you want to keep the false barge-in rate low, while also keeping the false reject rate as low as possible. In addition to an energy-based mechanism (sensitive to a different noise environment), WebSphere Voice Server V5.1 has a feature-based speech detector that detects speech by looking at the incoming features. This makes overall speech detection more robust to different energy levels.

You may want to carefully modify the sensitivity value after listening to audio files badly endpointed; but for an accurate tuning, you need help from IBM lab services.

## 9.6 Acoustic model adaptation

The following sections provide an overview of Acoustic Model adaptation, as well as how to set up WebSphere Voice Server to collect the adaptation data.

### 9.6.1 Overview

The acoustic models that ship with WebSphere Voice Server are trained on large amounts of data, which were wisely chosen to cover many different environments, different kinds of applications, and also regional differences of the speakers of this language.

You might in some cases find it useful to modify the acoustic model toward the actual use environment, the application domain, or the actual user group.

IBM offers adaptation of the acoustic model as a lab-based service, and WebSphere Voice Server V5.1.3 delivers all the options to collect the necessary data to adapt the acoustic model. For adaptation of the acoustic model, you need large amounts of data. You should have recorded at least the adaptation data of 5000 user utterances, the more data you have the better.

Necessary steps to perform adaptation:

1. Set up the WebSphere Voice Server to collect the adaptation data.
2. Run a data collection for the adaptation data, and, also, for test data.

3. Run collector.sh to package the adaptation data.
4. Engage lab-based services to perform the adaptation of the acoustic model.
5. Deploy adapted acoustic model on the WebSphere Voice Server.

### 9.6.2 Setting up WebSphere Voice Server to collect the adaptation data

To save all the data necessary for adaptation, you must enable ASRTRN=entryExit=enabled in the Diagnostic Trace Service of WebSphere Voice Server as described in 7.2, “Setting up files for analysis” on page 66.

After the data collection of the adaptation data, which you can also do by just activating the necessary traces on the production system, initiate a collector run on the WebSphere Voice Server box. On Linux, you typically do this by issuing the following command:

```
/opt/WebSphere/AppServer/bin/collector.sh
```

You can then send the JAR file that is produced to lab-based services so that they can actually adapt the acoustic model.

## 9.7 TTS tuning

The following sections are a guide to TTS tuning.

### 9.7.1 Lexicon

In 9.3, “Using lexicons for pronunciations” on page 109, we describe how to customize pronunciations for ASR purposes.

We can also modify the default TTS pronunciations by using our own TTS lexicon. You can find details about all required steps in Chapter 6, “Editor and Pronunciation Builder for lexicons” on page 57.

1. You first have to create your LXML file and select **Synthesizer (Text to Speech)** as shown in Figure 9-4 on page 117.

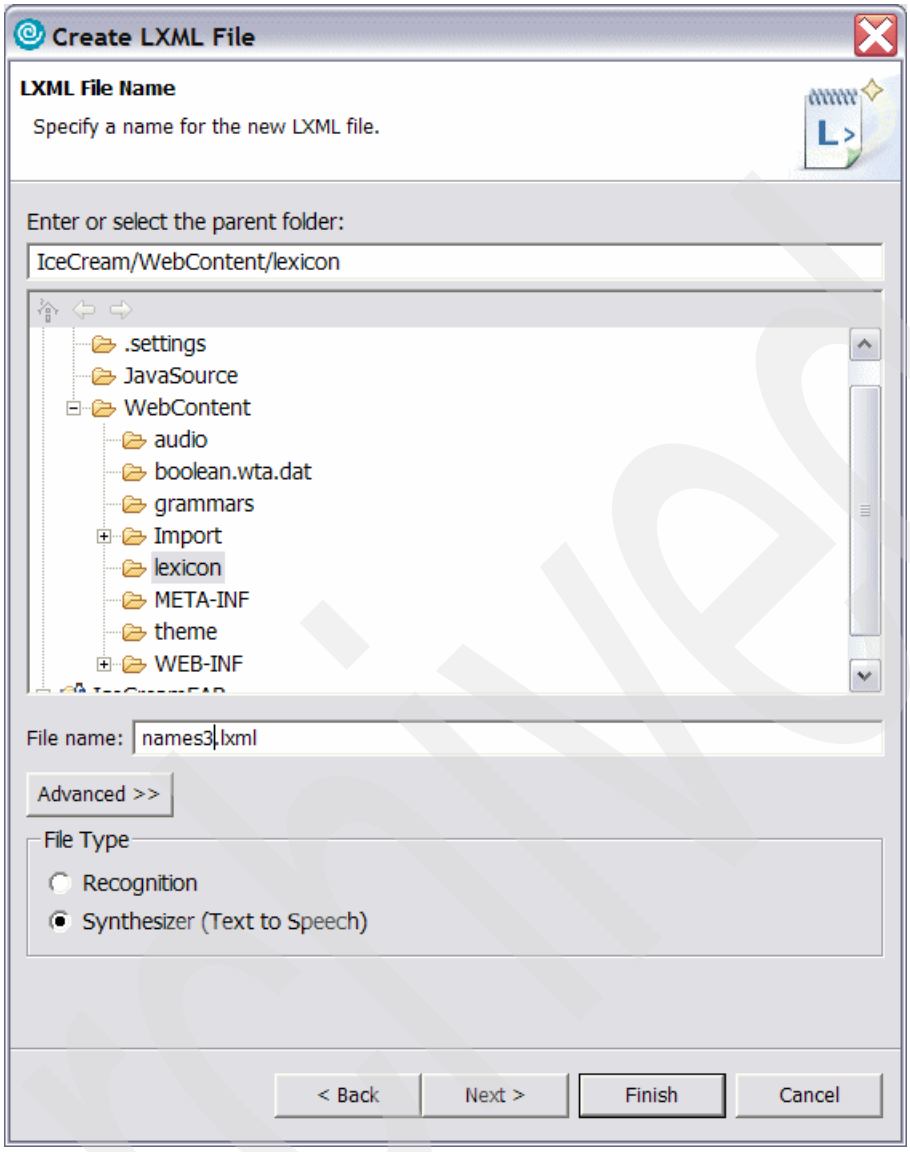



*Figure 9-4 Create LXML File*

2. Then you fill in the desired pronunciations using the Pronunciation Builder.

The result is your own TTS lexicon file as shown in Example 9-3.

*Example 9-3 names3.lxml: A customized version of our last names*

```
<?xml version="1.0" encoding ="UTF-8"?>
<!DOCTYPE lexicon PUBLIC "-//com/ibm/speech/grammar/lexicon//DTD Lexicon 1.0//EN"
"ibmlexiconml.dtd">

<lexicon version="1.0" xml:lang="en-US" alphabet="x-ibmtts" case-sensitive="true">
   <lexeme>
      <spelling>chamberlain</spelling>
      <phoneme>.1Cem.0bR.0lXn</phoneme>
   </lexeme>
   <lexeme>
      <spelling>elliott</spelling>
      <phoneme>.1E.0li.0Xt</phoneme>
```

```
   </lexeme>
   <lexeme>
      <spelling>klehr</spelling>
      <phoneme>.1klEr</phoneme>
   </lexeme>
   <lexeme>
      <spelling>baude</spelling>
      <phoneme>.1bod</phoneme>
   </lexeme>
</lexicon>
```

In Example 9-3 on page 117, we only changed the default pronunciation for “baude”.

## 9.7.2 Speech Synthesis Markup Language usage

The VoiceXML V2.0 specification adopted Speech Synthesis Markup Language (SSML) as the standard markup language for speech synthesis. SSML provides a standard way in which to control speech synthesis and text processing parameters for developers of speech applications. SSML enables developers to specify pronunciations, volume, pitch, speed, and so on.

For complete documentation about using IBM Text to Speech Technology and Speech Synthesis Markup, go to:

http://www.ibm.com/support/docview.wss?rs=426&context=SSMQSV&dc=DA400&uid=swg27005819&loc=en_US&cs=utf-8&lang=en

Although there is not a tool dedicated to SSML tuning in the current version of the Voice Toolkit, there is an easy way to test your SSML tags using the Pronunciation Builder described in 6.4, “Add words and pronunciations with Pronunciation Builder” on page 61.

1. You first need to create a lexicon file as described above.
2. Then you open the Pronunciation Builder (see Figure 9-5).

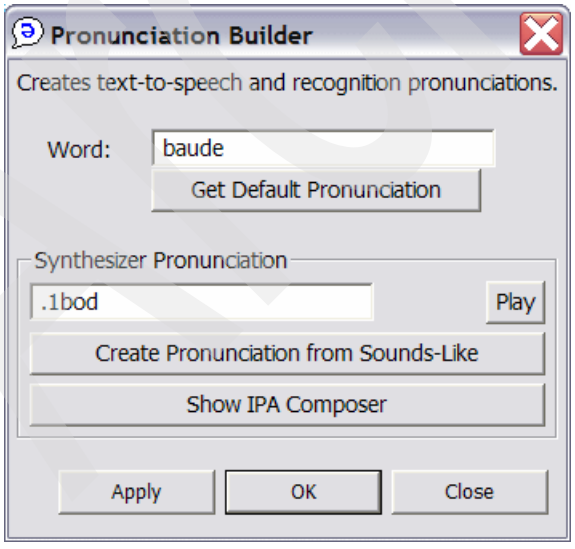



*Figure 9-5 The Pronunciation Builder for TTS*

3. Click **Create Pronunciation from Sounds-Like** (see Figure 9-6).

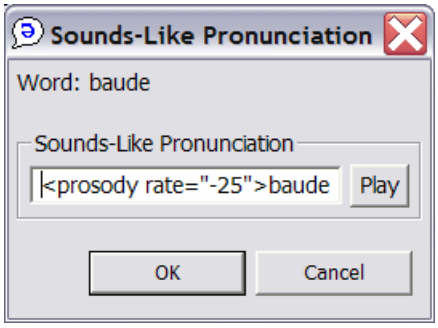



*Figure 9-6 Sounds-Like Pronunciation*

4. You can enter your SSML tags in the Sounds-Like Pronunciation field to listen to it. Refer to Example 9-4.

*Example 9-4 SSML tags applied on our last names*

```
<?xml version="1.0" encoding ="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.1//EN"
"http://www.w3.org/TR/voicexml21/vxml.dtd">

<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml"
   xml:lang="en-US">
   <meta name="GENERATOR" content="IBM WebSphere Voice Toolkit" />
   <form>
      <block>
         <prompt>
         jerome <prosody rate="-25" volume="loud">baude</prosody>
         markus <prosody rate="-25" volume="loud">klehr</prosody>
         gary <prosody rate="-25" volume="loud">elliott</prosody>
         james <prosody rate="-25" volume="loud">chamberlain</prosody>
         </prompt>
      </block>
   </form>
</vxml>
```

## 9.7.3 Phonetics and phone sequences in VoiceXML

The following example shows the US English phonetic pronunciation of “baude” using the IBM TTS phonetic alphabet. Refer to Example 9-5.

*Example 9-5 Use of IBM TTS phonetic alphabet*

```
...
         <prompt>
         jerome <phoneme alphabet="ibm" ph=".1bod">baude</phoneme>
         </prompt>
...
```

The pronunciation of “baude” is given in a notation called *Symbolic Representation (SPR)*. For more information about SPRs, see “Using IBM Text-to-Speech Technology and Speech Synthesis Markup” at the following Web site:

http://www.ibm.com/support/docview.wss?rs=426&context=SSMQSV&dc=DA400&uid=swg27005819&loc=en_US&cs=utf-8&lang=en

### 9.7.4 Lab service engagements

We have described what you can tune using the Voice Toolkit and SSML tags.

In addition to this, IBM labs can go further for specific customer needs. IBM TTS experts have tools for prompt optimization, and can also perform complete voice customizations.

Some companies want their own voice using their voice talent. You can get Corporate voices built on a lab service engagement. We can also enrich TTS voices with the brand new technique called *expressiveness*.

A

# Sample code

This appendix provides code samples used in the Redpaper.

## wines.txt

We used Example A-1 to create the wines.grxml file in 4.1, “SRGS XML Grammar Builder” on page 31.

*Example: A-1 wines.txt*

```
Bordeaux
Cabernet Sauvignon
Chardonnay
Chenin Blanc
Chianti
Gewurztraminer
Grenache
Grigio
Italian Red
Italian White
Malbec
Merlot
Mourvedre
Muscat
Nebbiolo
Petite Sirah
Pinot Blanc
Pinot Gris
Pinot Noir
Primitivo
Rhone
Riesling
Sangiovese
Sauvignon Blanc
Semillon
Shiraz
Syrah
```

```
Tempranillo
Viognier
Zinfandel
```

## wines.grxml

Example A-2 is the resulting output of the Grammar Editor. It was created in 4.1, “SRGS XML Grammar Builder” on page 31.

*Example: A-2* wines.grxml

```
<?xml version="1.0" encoding ="iso-8859-1"?>
<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
"http://www.w3.org/TR/speech-grammar/grammar.dtd">

<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar" xml:lang="en-US"
root="main_rule">
   <!-- Place Content Here -->


   <rule id="main_rule" scope="public">
      <one-of>
         <item>Bordeaux</item>
         <item>Cabernet Sauvignon</item>
         <item>Chardonnay</item>
         <item>Chenin Blanc</item>
         <item>Chianti</item>
         <item>Gewurztraminer</item>
         <item>Grenache</item>
         <item>Grigio</item>
         <item>Italian Red</item>
         <item>Italian White</item>
         <item>Malbec</item>
         <item>Merlot</item>
         <item>Mourvedre</item>
         <item>Muscat</item>
         <item>Nebbiolo</item>
         <item>Petite Sirah</item>
         <item>Pinot Blanc</item>
         <item>Pinot Gris</item>
         <item>Pinot Noir</item>
         <item>Primitivo</item>
         <item>Rhone</item>
         <item>Riesling</item>
         <item>Sangiovese</item>
         <item>Sauvignon Blanc</item>
         <item>Semillon</item>
         <item>Shiraz</item>
         <item>Syrah</item>
         <item>Tempranillo</item>
         <item>Viognier</item>
         <item>Zinfandel</item>
      </one-of>
   </rule>
</grammar>
```

## wines.gram

Example A-3 is the resulting output of the Grammar Editor. It was created in 4.4, “SRGS-ABNF Grammar Editor” on page 39.

*Example: A-3 Output from Grammar Editor*

```
#ABNF 1.0 iso-8859-1;
language en-US;

mode voice;
root $main_rule;
tag-format <semantics/1.0>;

public $main_rule = (Bordeaux
                   | Cabernet Sauvignon
                   | Chardonnay
                   | Chenin Blanc
                   | Chianti
                   | Gewurztraminer
                   | Grenache
                   | Grigio
                   | Italian Red
                   | Italian White
                   | Malbec
                   | Merlot
                   | Mourvedre
                   | Muscat
                   | Nebbiolo
                   | Petite Sirah
                   | Pinot Blanc
                   | Pinot Gris
                   | Pinot Noir
                   | Primitivo
                   | Rhone
                   | Riesling
                   | Sangiovese
                   | Sauvignon Blanc
                   | Semillon
                   | Shiraz
                   | Syrah
                   | Tempranillo
                   | Viognier
                   | Zinfandel);
```

## stock.grxml

Example A-4 was used in 4.2, “Graphics tab” on page 35 and Chapter 5, “Testing Grammars on MRCP” on page 45.

*Example: A-4 stock.grxml*

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" root="index"
   version="1.0" xml:lang="en-US">
   <rule id="index">
      <item>
      <item repeat="0-1">Please give me</item>
```

```
<one-of>
      <item repeat="0-1">the</item>
</one-of>
<one-of>
   <item>
      <one-of>
         <item repeat="0-1">current</item>
      </one-of>
      <item>price</item>
      <item repeat="0-1">of</item>
      <tag>$.requestType="intro";</tag>
   </item>
   <item>
      <item>change</item>
      <item repeat="0-1">in</item>
      <tag>$.requestType="change";</tag>
   </item>
   <item>
      <one-of>
         <item>high</item>
      </one-of>
      <item repeat="0-1">of</item>
      <tag>$.requestType="high";</tag>
   </item>
   <item>
      <one-of>
         <item>low</item>
      </one-of>
      <item repeat="0-1">of</item>
      <tag>$.requestType="low";</tag>
   </item>
   <item>
      <one-of>
         <item>opening price</item>
      </one-of>
      <item repeat="0-1">of</item>
      <tag>$.requestType="open";</tag>
   </item>
   <item>
      <item repeat="0-1">yesterday's</item>
      <one-of>
         <item>close</item>
         <item>closing price</item>
      </one-of>
      <item repeat="0-1">of</item>
      <tag>$.requestType="prevClose";</tag>
   </item>
   <item>
      <item>overview</item>
      <item repeat="0-1">on</item>
      <tag>$.requestType="overview";</tag>
   </item>
   <item>
      <one-of>
         <item>news</item>
         <item>rumors</item>
      </one-of>
      <item repeat="0-1">about</item>
      <tag>$.requestType="news";</tag>
   </item>
```

```
      <item repeat="0-1">the</item>
   </one-of>
</item>
<one-of>
   <item>
      <item>
         <one-of>
            <item>I B M</item>
            <item>
               International Business Machines
            </item>
         </one-of>
      </item>
      <tag>
         $.isin="US4592001014"; $.index="DE000DJII9";
      </tag>
   </item>
   <item>
      <item>
         I T S O Blue
         <item repeat="0-1">
            <ruleref uri="#co" />
         </item>
         <item repeat="0-1">
            <ruleref uri="#inc" />
         </item>
      </item>
      <tag>
         $.isin="US5893311077"; $.index="DE000DJII9";
      </tag>
   </item>
   <item>
      <item>
         Lotus
         <item repeat="0-1">Software</item>
      </item>
      <tag>
         $.isin="US9545682201"; $.index="DE000DJII9";
      </tag>
   </item>
   <item>
      <item>
         Tivoli
         <item repeat="0-1">Software</item>
      </item>
      <tag>
         $.isin="US4282361033"; $.index="DE000DJII9";
      </tag>
   </item>
   <item>
      <item>
         WebSphere
         <item repeat="0-1">Development</item>
         <item repeat="0-1">
            <ruleref uri="#inc" />
         </item>
      </item>
      <tag>
         $.isin="US7427181091"; $.index="DE000DJII9";
      </tag>
```

```
         </item>
      </one-of>
      <item repeat="0-1">please</item>
   </rule>
   <rule id="inc">
      <one-of>
         <item>Incorporated</item>
      </one-of>
   </rule>
   <rule id="co">
      <one-of>
         <item>Corporation</item>
      </one-of>
   </rule>
</grammar>
```

## stock.gram

Example A-5 is the conversion of stock.grxml to ABNF format in 4.6, “Grammar format converters” on page 41.

*Example: A-5 Stock.gram*

```
#ABNF 1.0 UTF-8;
language en-US;
root $index;
 $index =

  (
  (Please give me)  <0-1>
( (the)
 <0-1>)
( (
( (current)
 <0-1>)
  (price)
  (of)  <0-1> {$.requestType="intro";} )
      |
 (
  (change)
  (in)  <0-1> {$.requestType="change";} )
      |
 (
( (high)
)
  (of)  <0-1> {$.requestType="high";} )
      |
 (
( (low)
)
  (of)  <0-1> {$.requestType="low";} )
      |
 (
( (opening price)
)
  (of)  <0-1> {$.requestType="open";} )
      |
 (
```

```
  (yesterday's)  <0-1>
( (close)
      |
 (closing price)
)
  (of)  <0-1> {$.requestType="prevClose";} )
      |
 (
  (overview)
  (on)  <0-1> {$.requestType="overview";} )
      |
 (
( (news)
      |
 (rumors)
)
  (about)  <0-1> {$.requestType="news";} )
      |
 (the)
 <0-1>) )
( (
  (
( (I B M)
      |
 (
                      International Business Machines
                   )
) )  {
               $.isin="US4592001014"; $.index="DE000DJII9";
            } )
      |
 (
  (
               I T S O Blue

  (
  $co)  <0-1>
  (
  $inc)  <0-1>)  {
               $.isin="US5893311077"; $.index="DE000DJII9";
            } )
      |
 (
  (
               Lotus

  (Software)  <0-1>)  {
               $.isin="US9545682201"; $.index="DE000DJII9";
            } )
      |
 (
  (
               Tivoli

  (Software)  <0-1>)  {
               $.isin="US4282361033"; $.index="DE000DJII9";
            } )
      |
 (
  (
```

```
                WebSphere

  (Development)  <0-1>
  (
  $inc)  <0-1>)  {
                $.isin="US7427181091"; $.index="DE000DJII9";
             } )
)
  (please)  <0-1>;

 $inc =

( (Incorporated)
) ;

 $co =

( (Corporation)
) ;
```

## clean_config.xls

Example A-6 shows the XSL stylesheet used to convert config files created by CFB.

*Example: A-6 XSL stylesheet to convert config files created by CFB*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

   <xsl:output method="xml" encoding="utf-8" />

   <xsl:template match="@*|node()">
      <xsl:copy>
         <xsl:apply-templates select="@*|node()" />
      </xsl:copy>
   </xsl:template>

   <xsl:template match="input|confirm|echo">
      <xsl:element name="{name()}">
         <xsl:apply-templates select="property-list" />
         <xsl:if test="count(prompt-list)>0">
            <prompt-list>
               <xsl:for-each select="prompt-list">
                  <prompt>
                     <xsl:call-template name="get_prompts" />
                  </prompt>
               </xsl:for-each>
            </prompt-list>
         </xsl:if>
         <xsl:if test="count(help-list)>0">
            <help-list>
               <xsl:for-each select="help-list">
                  <help>
                     <xsl:call-template name="get_prompts" />
                  </help>
               </xsl:for-each>
            </help-list>
```

```
            </xsl:if>
            <xsl:if test="count(noinput-list)>0">
               <noinput-list>
                  <xsl:for-each select="noinput-list">
                     <noinput>
                        <xsl:call-template name="get_prompts" />
                     </noinput>
                  </xsl:for-each>
               </noinput-list>
            </xsl:if>
            <xsl:if test="count(nomatch-list)>0">
               <nomatch-list>
                  <xsl:for-each select="nomatch-list">
                     <nomatch>
                        <xsl:call-template name="get_prompts" />
                     </nomatch>
                  </xsl:for-each>
               </nomatch-list>
            </xsl:if>
         </xsl:element>
      </xsl:template>

      <xsl:template match="validate">
         <xsl:copy-of select="." />
      </xsl:template>

      <xsl:template match="property-list">
         <xsl:copy-of select="property" />
      </xsl:template>

      <xsl:template name="get_prompts">
         <xsl:attribute name="count">
            <xsl:value-of select="@count" />
         </xsl:attribute>
         <xsl:copy-of select="prompt" />
      </xsl:template>

   </xsl:stylesheet>
```

## GetNbestList.vxml

Example A-7 shows how to get the Nbest list together with the associated confidence scores. You could take advantage of this information to implement different algorithms for score filtering.

*Example: A-7 GetNbestList.vxml*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE vxml PUBLIC "-//W3C//DTD VOICEXML 2.0//EN"
"http://www.w3.org/TR/voicexml20/vxml.dtd">
<vxml version="2.0" xml:lang="en-US" xmlns="http://www.w3.org/2001/vxml">
<meta name="GENERATOR" content="Voice Toolkit for WebSphere Studio" />

<property name="confidencelevel" value="0.2" />
<form id="test">
<field name="testgram">
<property name="maxnbest" value="3" />
<prompt>Say character</prompt>
```

```
            <grammar mode="voice" type="application/srgs">
            #ABNF 1.0;
            language en-US;
            mode voice;
            root $character;
            public $character= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P
| Q | R | S | T | U | V | W | X | Y | Z | zero | one | two | three | four | five | six |
seven | eight | nine | goodbye;
            </grammar>
            <filled>
               <log>
                  UTTERANCE:
                  <value expr="testgram$.utterance" />
               </log>
               <log>
                  INTERPRETATION:
                  <value expr="testgram$.interpretation" />
               </log>
               <log>
                  CONFIDENCE:
                  <value expr="testgram$.confidence" />
               </log>
               <if cond="application.lastresult$.length &gt; 1">
                  <log>
                     U1:<value expr="application.lastresult$[1].utterance" />
                  </log>
                  <log>
                     I1:<value expr="application.lastresult$[1].interpretation" />
                  </log>
                  <log>
                     C1:<value expr="application.lastresult$[1].confidence" />
                  </log>
               </if>
               <if cond="application.lastresult$.length &gt; 2">
                  <log>
                     U2:<value expr="application.lastresult$[2].utterance" />
                  </log>
                  <log>
                     I2:<value expr="application.lastresult$[2].interpretation" />
                  </log>
                  <log>
                     C2:<value expr="application.lastresult$[2].confidence" />
                  </log>
               </if>
               <goto next="#test" />
            </filled>
         </field>
      </form>
</vxml>
```

# Abbreviations and acronyms

| | |
|---|---|
| **ABNF** | Augmented Backus Naur Form |
| **AH** | Authentication Header |
| **ALW** | headerless, 8kHz 8-bit a-law single channel audio format |
| **API** | application programming interface |
| **AS** | Advanced Server |
| **ASR** | Automatic Speech Recognition |
| **AU** | Australian |
| **BNF** | Backus-Naur Form |
| **CA** | Correct Accept |
| **CCXML** | Call Control Extended Markup Language |
| **CD** | compact disc |
| **CFB** | Communication Flow Builder |
| **CPU** | Central Processing Unit |
| **CR** | Correct Reject |
| **CSV** | comma separated value |
| **DTD** | Document Type Definition |
| **DTMF** | Dual Tone Multi-Frequency |
| **ER** | exception response |
| **ESD** | electrostatic discharge |
| **FA** | False Accept |
| **FR** | False Rejects |
| **GRXML** | GRammar eXtensible Markup Language |
| **GUI** | Graphical User Interface |
| **HTML** | Hypertext Markup Language |
| **HTTP** | Hyper Text Transport Protocol |
| **IBM** | International Business Machines Corporation |
| **IBM** | International Business Machines Corporation |
| **ID** | identifier |
| **IP** | Internet Protocol |
| **IPA** | International Phonetic Alphabet |
| **ISV** | independent software vendor |
| **ITSO** | International Technical Support Organization |
| **IVR** | interactive voice response |
| **JDK** | Java Development Kit |
| **JRE** | Java Runtime Environment |
| **JSGF** | Java Speech Grammar Format |
| **JSP** | JavaServer Pages |
| **LXML** | Lexicon Extensible Markup Language |
| **MRCP** | Media Resource Control Protocol |
| **NLU** | Natural Language Understanding |
| **RAD** | Rational Application Developer |
| **RDC** | Reusable Dialog Component |
| **ROI** | return on investment |
| **RSA** | Rational Software Architect |
| **RWD** | Rational Web Developer |
| **SISR** | Semantic Interpretation for Speech Recognition |
| **SPR** | Symbolic Phonetic Representations |
| **SRGS** | Speech Recognition Grammar Specification |
| **SRGXML** | Speech Recognition Grammar Extended Markup Language |
| **SSML** | Speech Synthesis Markup Language |
| **SUI** | Speech User Interface |
| **TTS** | Text to Speech |
| **ULW** | mu-law or µ-law |
| **URI** | Uniform Resource Identifier |
| **URL** | Uniform Resource Locator |
| **US** | United States |
| **VoiceXML** | Voice eXtended Markup Language |
| **VOX** | VoxWare MetaVoice Encoded Audio format |
| **VTK** | Voice Toolkit |
| **VXML** | Voice eXtended Markup Language |
| **WAV** | PCM Waveform Audio. A format to store digitally recorded sound. |
| **WVS** | WebSphere Voice Server |
| **XML** | eXtensible Markup Language |
| **XSL** | eXtensible Stylesheet Language |

# Glossary

**Acoustic model.** This represents the variability of the acoustic profile of a phoneme under different circumstances.

**Automatic Speech Recognition (ASR).** Speech recognition technologies allow computers equipped with a source of sound input, such as a telephone handset or microphone, to interpret human speech.

**Barge-in.** This refers to the ability of a caller to interrupt a prompt as it is playing, either by saying something or by pressing a key on the telephone keypad.

**Confidence score.** This refers to the likelihood that the text returned by a speech recognition engine is correct. The higher the confidence score, the more likely the speech recognition engine correctly identified what was said.

**Confusability.** A concept in which words with similar phonetic sequences can be confused with each other, for example, can't elope, can elope, and cantaloupe. In other words, there may be similar-sounding words or phrases in the same grammar used at the same point in the dialog; for example, a voice mail system where the system is expecting the user to say "replay" or "reply" in response to a voice message. You can resolve this by changing "replay" to "play message again" to avoid the confusion.

**cut-thru.** See Barge-in.

**eXtensible Markup Language (XML).** A standard metalanguage for defining markup languages that is based on Standard Generalized Markup Language (SGML). XML simplifies the process of authoring and managing structured information and transmitting and sharing structured information across diverse computing systems.

**Grammar.** This uses a particular syntax, or set of rules, to define the words and phrases that can be recognized by the speech recognition engine. A grammar can be as simple as a list of words, or it can be flexible enough to allow so much variability in what can be said that it approaches natural language capability.

**Lombard Speech.** This is related to barge-in and refers to the tendency of people to speak louder in noisy environments, in an attempt to be heard over the noise. Callers barging-in tend to speak louder than necessary, which can be problematic in speech recognition systems. Speaking louder does not help the speech recognition process. On the contrary, it distorts the voice and hinders the speech recognition process.

**Natural Language Speech Recognition (NLSR).** An advanced type of speech recognition. NLSR can recognize particular words and phrases spoken by the user.

**Phoneme.** This is the basic acoustic unit of spoken language. For example, the "oo" sound in "hoop" is a phoneme. The systems that perform Automatic Speech Recognition must match speech sounds to phonemes by using acoustic models.

**Phonology.** This is the study of phonemes and their usage. In other words, to study how sounds are organized and used in natural languages.

**Recognition Timeout.** This is the maximum duration of user speech. Each VoiceXML Browser has its own default.

**Silence.** This delineates the start and the end of an utterance.

**Speech Synthesis Markup Language (SSML).** A markup language to control speech synthesis and text processing defined by the World Wide Web Consortium (W3C).

**SSML.** See Speech Synthesis Markup Language.

**Stuttering Effect.** If there is a noticeable delay, greater than 300 MS, from when the user says something and when the prompt ends, then, quite often, the caller does not think the system heard what was said, and will most likely repeat what was said, and both the caller and the system get into a confusing situation.

**Text to Speech (TTS).** An optional feature that lets an application play language speech directly from ASCII text by converting that text to synthesized speech. You can use the text for prompts or for text retrieved from a database or host, and TTS can be spoken in an application with prerecorded speech.

**Turn.** In the Voice Trace Analyzer tool, a turn is a MRCP RECOGNIZE message that is assigned a unique number starting at 1 and increments by 1.

**Utterance.** This is any stream of speech between two periods of silence. In other words, this is what the user says. An utterance can be a single word, or it can contain multiple words (a phrase or sentence). For example, "checking", "checking account", or "I would like to know the balance of my checking account please" are all examples of possible utterances that a caller might say to a banking application written in VoiceXML.

**Voice eXtensible Markup Language (VoiceXML).** The W3C's standard XML format for specifying interactive voice dialogues between a human being and a computer. It is fully analogous to HTML, and brings the same advantages of Web application development and deployment to voice applications that HTML brings to visual applications.

**Voice Trace Analyzer.** A tool shipped with the WebSphere Voice Toolkit V6.0 product that helps to validate your system's speech recognition performance and to troubleshoot problems.

**VoiceXML.** See Voice eXtensible Markup Language.

**XML.** See eXtensible Markup Language.

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 136. Note that some of the documents referenced here may be available in softcopy only.

- *IBM WebSphere Voice Server for Multiplatforms V5.1.1/V5.1.2 Handbook*, SG24-6447

  http://www.redbooks.ibm.com/abstracts/sg246447.html

## Online resources

These Web sites and URLs are also relevant as further information sources:

- World Wide Web Consortium (W3C) Home Page:

  http://www.w3c.org
- W3C recommendation of Speech Recognition Grammar Specification (SRGS) V1.0:

  http://www.w3.org/TR/2004/REC-speech-grammar-20040316
- W3C recommendation of Speech Synthesis Markup Language (SSML) V1.0:

  http://www.w3.org/TR/2004/REC-speech-synthesis-20040907
- W3C working draft of Semantic Interpretation for Speech Recognition (SISR):

  http://www.w3.org/TR/2003/WD-semantic-interpretation-20030401
- W3C recommendation for Voice Extensible Markup Language (VoiceXML) V2.0:

  http://www.w3.org/TR/voicexml20
- WebSphere Application Server Zone:

  http://www.ibm.com/developerworks/websphere/zones/was
- WebSphere Application Server V5.1.x Information Center:

  http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp?topic=/com.ibm.websphere.base.doc/info/welcome_base.html
- WebSphere Voice family product documentation library:

  http://www.ibm.com/developerworks/websphere/zones/voice/proddoc.html#wvs
- WebSphere Voice Server Education and Class Information:

  http://www.ibm.com/developerworks/websphere/education/enablement
- WebSphere Voice Server for Multiplatforms V5.1.x Information Center:

  http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/index.jsp
- WebSphere Voice Server Support Page:

  http://www-306.ibm.com/software/pervasive/voice_server/support
- WebSphere Voice Server System Requirements:

  http://www.ibm.com/software/pervasive/voice_server/system_requirements

- WebSphere Voice Server IVR and gateway compatibility web page:

  http://www.ibm.com/software/pervasive/voice_server/ivrgateway.html

- WebSphere Voice Toolkit:

  http://www.ibm.com/software/pervasive/voice_toolkit

- WebSphere Voice Zone:

  http://www.ibm.com/developerworks/websphere/zones/voice

- IBM VoiceXML Programmer's Guide

  http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/topic/com.ibm.voicetools.reldocs.doc/pgmguide.pdf

- Using IBM Text to Speech Technology and Speech Synthesis Markup Language

  http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/topic/com.ibm.voicetools.reldocs.doc/tts_ssml.pdf

- Prototyping SUI with VoiceXML and Voice Toolkit

  http://publib.boulder.ibm.com/infocenter/pvcvoice/51x/topic/com.ibm.voicetools.reldocs.doc/vxmluiproto.pdf

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

**ibm.com**/redbooks

## Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

# Speech User Interface Guide

Redpaper

**Leverage the efficiency of your voice applications**

**Develop voice applications using Reusable Dialog Components (RDCs)**

**Tune voice applications using the Voice Toolkit**

To acquire new customers and to retain existing customers, companies must give credence to customer satisfaction. Call centers with well crafted speech-enabled applications significantly improve customer satisfaction, as well as provide customers with additional services and 24x7 support.

These speech-enabled applications rely on the development of pleasant and efficient Speech User Interfaces (SUIs).

If the Speech User Interface is improperly designed, constructed, and tuned pre- and post-deployment, this incurs unreasonable and unnecessary expenses that can lead to critical situations, increased problem activity, and decreased customer satisfaction.

In order to craft an effective SUI, you should follow proper and proven methodology (best practices). This Redpaper details an effective methodology that you can use to create and deliver high quality Speech User Interfaces to meet business needs. We use IBM WebSphere Voice Toolkit V6.0.1 throughout to test and tune WebSphere Voice Server to ensure it is optimally configured for your SUIs. However, it is beyond the scope of this Redpaper for specifics about programming languages used, such as VoiceXML.

We assume a basic knowledge of VoiceXML development using WebSphere Voice Server and Interactive Voice Response.

**INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

**BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**
**ibm.com**/redbooks