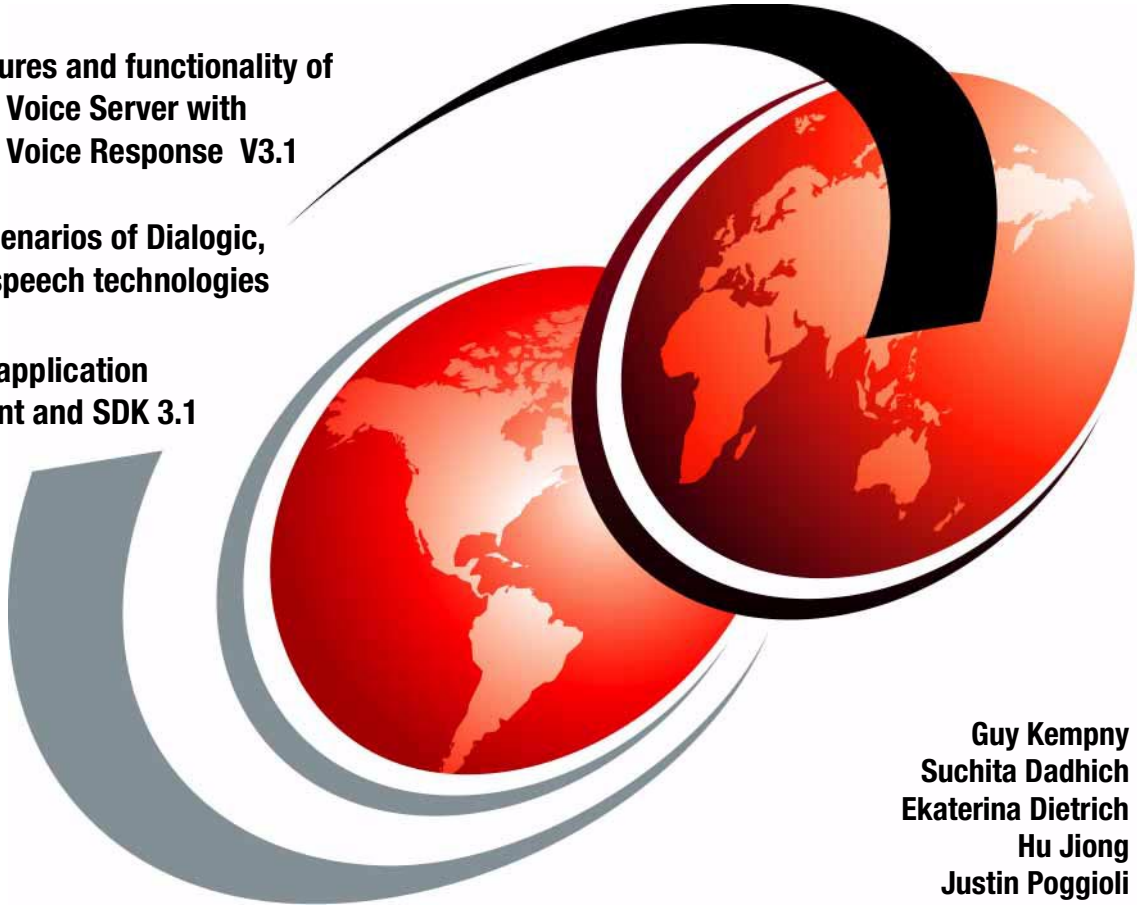
WebSphere software

IBM

IBM WebSphere Voice Systems Solutions

- Covers features and functionality of WebSphere Voice Server with WebSphere Voice Response V3.1
- Includes scenarios of Dialogic, Cisco and speech technologies
- Highlights application development and SDK 3.1



Guy Kempny
Suchita Dadhich
Ekaterina Dietrich
Hu Jiong
Justin Poggioli

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM WebSphere Voice Systems Solutions Implementation Guide

January 2003

SG24-6884-00

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

First Edition (January 2003)

This edition applies to WebSphere Voice Server for Windows 2000 and AIX, V3.1, WebSphere Voice Response for AIX, V3.1, and WebSphere Voice Response for Windows NT and Windows 2000, V3.1.

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:

AIX®
alphaWorks®
DB2®
DB2 Universal Database™
DirectTalk®
IBM @server™
Everyplace™
Home Director™
IBM®
iSeries™
MVS™
Netfinity®
OS/390®
pSeries™
Redbooks™
Redbooks (logo)™ 
RS/6000®
S/390®
SecureWay®
SP™
ThinkPad®
Tivoli®
ViaVoice®
WebSphere®
xSeries™

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

Domino™
Lotus Notes®
Lotus®
Notes®

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, Windows 2000, Windows XP and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

CCIP, the Cisco Arrow logo, the Cisco Powered Network mark, the Cisco Systems Verified logo, Cisco Unity, Follow Me Browsing, FormShare, iQ Breakthrough, iQ Expertise, iQ FastTrack, the iQ logo, iQ Net Readiness Scorecard, Networking Academy, ScriptShare, SMARTnet, TransPath, Voice LAN are trademarks of Cisco Systems, Inc.

ActionMedia, AnyPoint, AppChoice, BoardWatch, BunnyPeople, CablePort, Celeron, Chips, CT Media, Dialogic, DM3, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Create & Share, Intel GigaBlade, Intel InBusiness, Intel Inside, Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel Play, Intel Play logo, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel TeamStation, Intel Xeon, Intel XScale, IPLink, Itanium, LANDesk, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PC Dads, PC Parents, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon, Performance at Your Command, ProShare, RemoteExpress, SmartDie, Solutions960, Sound Mark, StorageExpress, The Computer Inside., The Journey Inside, TokenExpress, Trillium, VoiceBrick, VTune, and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Aculab, the Groomer, Prosody, “Connected with Aculab”, Aculab worldwide digital connectivity and Aculab your connection to the future are registered trademarks of Aculab Plc. TiNG and Whirlwind are trademarks of Aculab Plc. All other product or company references or registered and unregistered trademarks are the sole property of their respective owners.

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

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

Voice-based applications have become increasingly popular as a means to access data and applications. Advances in the interfaces between digital data and voice have led to the increased use of this technology in accessing traditional personal and business applications. Voice-based applications are those that allow the customer to select menu items or enter entry fields using a normal telephone. This allows the customer to interact with the system through traditional voice communication and the familiar telephone keypad, thereby leveraging fundamentals of human communication and increasing ease-of-use and customer satisfaction while decreasing the likelihood of error.

The WebSphere Voice Server product is a member of the IBM WebSphere software family. It provides a platform that enables the creation of voice applications through industry standards such as VoiceXML and Java. The WebSphere Voice Server facilitates the deployment of voice applications by interfacing with voice standards such as Cisco VoIP, IBM WebSphere Voice Response, IBM DirectTalk, and Intel Dialogic platforms. It further aids the development of these applications by providing development tools.

This IBM Redbook discusses the functionality of WebSphere Voice Server in the context of real business environments. We discuss the voice environment and the WebSphere Voice Server. We cover in great detail the various operating platforms supported by WebSphere Voice Server (WebSphere Voice Response, Intel Dialogic, and Cisco). The product has been further enhanced to function on both Intel and AIX systems. The redbook goes into some depth about this new functionality.

Additionally, we discuss both the Software Developers Kit (SDK) and the Voice Toolkit that are available for the development of voice applications. A step-by-step approach was taken to walk through the development of a VoiceXML application utilizing both of these tools, and taking advantage of the new functions provided within them. We also include a section about the different languages supported by WebSphere Voice Server, and how we deployed an actual multi-language system using non-English languages and operating systems in our test environment.

We complete our discussion with a sampling of advanced functionality available to the WebSphere Voice Server such as transcoding, the use of the IBM Voice Portal solution, and mixed operating system environments.

The team that wrote this redbook

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

Guy Kempny is an IT software specialist in IBM Australia. He has 14 years of experience in the IT industry. His expertise covers a broad range of areas that include Intel software support, OS/390, speech software, and the WebSphere product set, as well as project managing IT solutions for IBM's corporate sports sponsorship and large-scale events. Before joining the IBM Marketing team, he was part of the WebSphere software technical team. Prior to this, he was a Program Manager for the IBM Olympic Project Office in Sydney. Guy served as the ITSO Resident Project Leader for this residency.

Suchita Dadhich is an Advisory IT Specialist for Advanced Technical Support organization in Philadelphia, PA, USA. She has several years of experience in delivering technical sales assistance for IBM's WebSphere software portfolio. Her areas of expertise include configuration, architecture design and validation for voice and contact center solutions, including Interactive Voice Response Systems (IVR), as well as application development.

Ekaterina Dietrich is an IT specialist in T-Systems DE-Telekom, department of System Integration in Berlin, Germany. She has 15 years of experience in the IT industry. She holds a PhD degree in Mathematics from Moscow State University. Her areas of expertise include the development of voice applications for different computer telephony platforms of Deutsche Telekom.

Hu Jiong is a software engineer in the Globalization Certification Lab, IBM China. He has four years of experience in the IT industry. Hu Jiong has been focusing on the globalization of the IBM Software and e-Business Solution since he joined IBM two years ago. Hu is a graduate of the East China University of Science and Technology. He is familar with AIX, Linux, Java and WebSphere products. He is currently researching the globalization solution of pervasive devices and voice systems.

Justin Poggioli is a Technical Consultant for Viecore, an IT services firm in Upper Saddle River, NJ specializing in the delivery of complex IT solutions. His areas of expertise include high-level designs for IVR solutions, creation and integration of IVR, CTI and speech applications, and delivering those applications within budgetary and scheduling restraints. He currently works as an engineer and developer in the Java Beans and WebSphere environments.

Thanks to the following people for their contributions to this project:

Tamikia Barrow, Jeanne Tucker, Margaret Ticknor
International Technical Support Organization, Raleigh Center

Rufus Credle, ITSO Project Leader
IBM Research Triangle Park

Alan Wagoner, L2 Defect Support (WebSphere Voice and Translation Servers + DirectTalk for Windows)
IBM Research Triangle Park

Len Zydel, Project Manager
IBM Boca Raton

Debbie Paladino, PVC, Development Manager WebSphere Voice Server, Certified Project Manager/PMP
IBM Boca Raton

Kirk Smith, Manager, Voice Systems Product Support
IBM Research Triangle Park

David Liu, WebSphere Voice Response for AIX Level 2 support
IBM Research Triangle Park

Ethan Merrill, DirectTalk Level 2
IBM Research Triangle Park

Xiaoming Wang, Voice Systems Enterprise Test
IBM Boca Raton

Charlie Sumner, Manager, Voice Systems Enterprise Test
IBM Boca Raton

Charieese Sutton, Manager, Machine Translation
IBM Research Triangle Park

Jeff Kusnitz, Speech Recognition
IBM San Jose

Scott Filor, IBM Field Sales
Intel-Dialogic, Parsippany NJ

John Kozlowski, Regional Manager
Aculab, Wilton CT

Manolo Pedreschi, IT Architect
IBM New Zealand

Neville Richards, Consultant
Viecore Pty Ltd, Australia

Wai-Kong Ho, Senior IT Specialist
IBM Australia

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 Redbooks to be as helpful as possible. Send us your comments about this 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 Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HQ7 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

1

Voice technologies

This chapter provides a description of what voice applications are and the technologies used to enable business applications for voice.

1.1 Access to information through voice

By increasing the customer's ability to complete a task or request without external human assistance, voice technologies are changing the way customer self-service works. Systems are no longer bound by the structured, telephone keypad-driven dialogs that characterize traditional voice response systems. Voice technologies are aimed at breaking these constraints, enabling customers to interact with an automated, self-service environment in a similar manner to a conversation with another person.

In today's environment, a wide range of business applications can be enabled for voice. Enabling an application for voice does not imply discarding the graphical interface. It means providing a different way of accessing the same information and services, although the same information when provided over the telephone may require some changes from its graphical presentation state (for example, a list box with multiple choices is easily shown in a visual application, but it may not be appropriate to read all options to the customer over the telephone). In summary, only the presentation aspect of the information is changed, not the information itself or how it is generated.

Typically, the business applications to be enabled for voice are either inquiries or transaction-based functions or applications.

For example, a customer can invoke a *query application* to retrieve information from a database. Account balances, weather reports, movie listings or stock quotes obtained from a contact center's Interactive Voice Response system (IVR) are common examples of query applications that can be enabled for voice. Usually, the application guides the customer through a series of menus and forms by playing instructions, prompts, and choices using either prerecorded audio files or text-to-speech, also known as TTS, the technology that converts text into spoken language. The customer can use spoken words to make selections and to provide information. Once the database records are retrieved, the system presents the information by playing back audio files.

On the other hand, a *transaction application* is used for the purpose of performing specific tasks. Transfering funds between accounts and buying/selling stocks are popular examples of this type of application. In the above scenarios, the application starts by authenticating the customer through an identification process, and then uses a series of dialogs to guide the customer to enter the required data (for example, the source and target accounts for a funds transfer transaction). The system plays instructions, prompts, and available options using prerecorded audio files or text-to-speech, and the customer responds using spoken commands. Once the required data is collected, the system performs the transaction and returns the result.

The advantages of enabling existing business applications for voice are manifold:

- *Broader audience* - A larger customer base can be reached with the same infrastructure by allowing access through voice channels. Customers can access information and conduct transactions via more than just the Web. Also, information can be conveyed in spoken words, providing a more natural, user-friendly means of communication.
- *Increased productivity* - Applications and databases can be made readily available to mobile employees, hence improving productivity and overall business efficiency.

Leverage investment - Companies have made significant investment in deploying contact centers and IVR and Web infrastructures. Providing voice access to applications allows businesses to leverage this investment by using the same technology infrastructure, databases, and application development skills.

1.2 What are voice applications?

Voice applications are those in which input and output interactions come through a spoken user interface, as opposed to a graphical one. The shape this spoken data takes is not identical to the form it takes in a visual interface, due to the inherent differences between them.

Business applications enabled for voice can be a powerful tool for users to “browse-by-voice” and to interact with Web-based data, using speech instead of keyboard and mouse. Such applications are capable of exploiting an enterprise architecture comprising servlets, ASPs, JSPs and JavaBeans.

Voice applications can reside on local or remote systems, and customers can access them from a telephony-capable device.

A typical voice application call flow may consist of the following steps:

- The customer rings the contact center and the system answers the call while invoking for execution the appropriate application, such as the one referenced by Dialed Number Identification Service (DNIS) or Automatic Number Identification (ANI) information.
- The application plays a greeting and prompts the customer for the necessary information.
- The application awaits for a response for a set period of time. The caller can respond by speaking or by pressing keys on a telephone keypad.

- The application takes the appropriate action based on the customer's response. For example, it might update information in a database, retrieve information and play it back as audio, or play a help message.
- The customer can terminate the call at any time, simply by hanging up the telephone or by speaking to indicate the interaction is complete (for example, by saying "Stop" or "Exit").

1.3 Speech recognition

Speech recognition, the recognition of human speech by a computer, is a voice technology that translates spoken input into text. Just like using the keyboard or mouse or pressing a key on the telephone keypad to provide input, speech recognition allows for providing input by simply talking.

The speech recognition process is performed by a software component known as the speech recognition engine, whose primary function is to process speech input and translate it into text. The engine relies on statistics and software algorithms to analyze the incoming audio signal and to search for the best match, taking into consideration known words and phrases (the active grammars) and the environment (the acoustic model). Once the most likely match is identified, the engine returns it as a text string.

The application can act upon the returned text string. For example, a customer might say something like "credit card account balance" to which the application may reply "two thousand, seventeen dollars and five cents".

Sometimes the match may fail because the customer said something that the application was not expecting or could not understand. To avoid acting inappropriately as a consequence, the application should have error checking routines built into it, based on the confidence of the recognition result obtained.

How does it work?

Speech recognition translates speech data into a format that voice applications can understand and process. An illustration of this process is shown in Figure 1-1 on page 5.

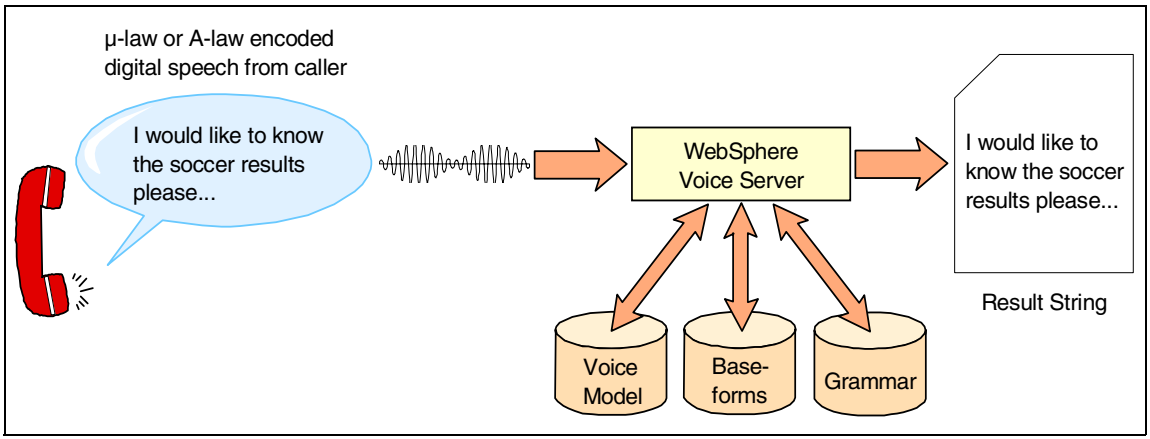


Figure 1-1 The speech recognition process

In Figure 1-1, the *Voice Model* is responsible for keeping a set of parameters for the language to be recognized. The *Baseform* maintains a description of what the words should sound like. Finally, the *Grammar* contains a description of the words and the multiple sequence of words acceptable for the current recognition task.

An utterance of any length can be presented to the system, which then searches the currently loaded grammar for possible word matches. A voice application can contain up to 65,536 words and use more than one grammar.

The overall response time can be affected by the length of the utterance and the size of the grammar.

An application can allow a customer to speak over the voice prompt. With this feature, known as barge-in or cut-through, the recognition algorithm stops when the customer speaks in the middle of a prompt.

A grammar uses a particular set of rules to define the words and phrases that can be recognized by the engine. Each new phrase, or utterance, is compared with the words and phrases in the active grammars, which can define several ways to say the same thing.

Speaker dependency

The recognition engine can be trained to an individual's voice in a “speaker-dependent” system, as in most desktop dictation applications. However, server-based applications are designed to service a wide variety of customers, so they cannot be trained to each individual. They are known as “speaker-independent”.

Dialog types

Three types of dialogs are used in voice recognition applications:

- *Directed.* The application prompts the customer to perform a specific task while controlling the interaction. This menu-driven approach leads the user to complete a function by asking for information at each turn of the dialog, and expecting a specific reply each time.
- *Mixed Initiative.* The customer can control it by providing more input than what is being currently requested, and in a different order than expected. So the caller can anticipate the next prompt and act accordingly.
- *Natural Language Understanding.* Grammars are designed to allow a more conversational interaction with an application, and to mirror the way people talk to each other. This approach requires complex grammars as well as significantly more application logic, because the application needs to extract meaning and context from each spoken utterance.

1.4 Text-to-speech

Text-to-speech is a voice technology that converts text into spoken output. It is performed by a specialized software component known as the TTS engine, whose primary function is to process text input and translate it into spoken output.

Input may consist of text and, optionally, special tags (annotations) that can change the sound of the voice that is ultimately produced.

The major types of TTS currently available in the marketplace include:

- *Formant*, which uses standard computer-generated, synthesized voice sounds.
- *Concatenative*, which is based on joining small samples of actual voice recordings, resulting in a more natural voice sound.

See Figure 1-2 on page 7 for an illustration of the text-to-speech process.

How does it work?

The TTS engine starts by to analyzing the word, phrase, or sentence to vocalize. It expands abbreviations, handles contractions and numbers, and then it *disambiguates* the semantics of the sentence in order to know what words are really intended. For example, “Dr. Patrick lives on Patrick Dr.” must be disambiguated so that it reads “Doctor Patrick lives on Patrick Drive”.

Once the words have been analyzed, the system determines how to synthesize them into speech. Appropriate audio characteristics are applied (such as volume, pitch, and speed), and the speech output is produced.

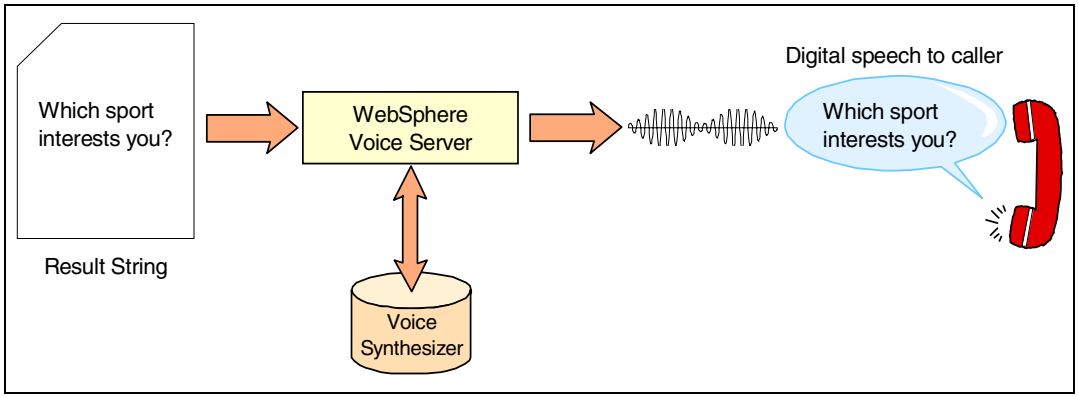


Figure 1-2 The text-to-speech process

Both formant and concatenative TTS technologies use the same front-end system to analyze and process the input; the differences lie in how the output is synthesized. In a formant system, speech output is generated completely algorithmically using knowledge of how the human vocal tract works (it has certain major resonant frequencies). The output produced is highly intelligible, although it may sound robotic or mechanical to some listeners.

On the other hand, in a concatenative system, speech output is produced from recordings of units of actual human speech. These units (sounds, syllables, words, or phrases) are then combined, or concatenated, according to linguistic rules to produce the speech output, resulting in a natural, human-sounding speech.

1.5 Terminology

The following are some of the terms commonly associated with telephony and voice technologies:

- *Acceptance / Rejection* - The two states of recognition results that can be returned by the TTS engine: recognized text and confidence score.
- *Accuracy* - A quantitative measure of a speech recognition system's performance.
- *Annotation* - Alphanumeric tags embedded within the text sent to the TTS engine. They are applied to change some TTS characteristics, for example the gender, speed, and emphasis of the spoken voice.

- *Automatic Number Identification (ANI)* - Often known as the calling number, ANI is an optional feature that the central office (CO) telephony switch or PBX provides at call setup time. ANI can be used to personalize the experience by coding an application to determine whether the customer is using the system for the first time or to greet the customer by name.
- *Barge-in* - The ability to interrupt a playing prompt by saying something or by pressing a key on the telephone keypad.
- *Dictionary* - A collection of pronunciations or phonetic representations of words. The TTS engine uses a dictionary along with the knowledge of the language it is synthesizing to pronounce words from the input text. Typically, there is a default dictionary, and there are also exception dictionaries into which custom pronunciations of words can be added. The search order is exception dictionaries first, and then the default dictionary.
- *Dialed Number Identification Service (DNIS)* - Also referred to as the called number, DNIS is another optional feature that the CO or PBX provides when the call is established. DNIS can be used to distribute incoming calls among different voice applications.
- *Grammar* - A specification of the spoken words, phrases, and sentences that the system can process and convert to text.
- *Phoneme* - The basic unit of speech in a given language. It represents a unique sound, not necessarily equivalent to a letter of the alphabet. For example, there are different phonemes for the “a” sound in the words “place”, “cat”, and “ball”.
- *Prompt* - The audio played during a telephone call.
- *Pronunciation* - The phonetic representation of a word. For example, the word “schedule” can be pronounced as “skehduwl” in US English or as “shehdyuwl” in UK English.
- *Prosody* - How well the output shows the speech rhythm, patterns, and even its irregularities.
- *Utterance* - A single stream of speech between two periods of silence.
- *Vocabulary* - Total set of words that the TTS engine can pronounce. Theoretically, the vocabulary of TTS is unlimited; that is, the engine should be able to pronounce any given text string.

1.6 VoiceXML

The Voice eXtensible Markup Language (VoiceXML) is an XML-based, industry-standard language for creating voice applications, much as HTML is a language for developing visual applications.

VoiceXML is defined and promoted by an industry forum, the VoiceXML Forum, founded by AT&T, IBM, Lucent and Motorola, and currently supported by 575 member companies.

VoiceXML was designed to create audio dialogs that feature text-to-speech, digitized as well as prerecorded audio, recognition of both spoken and dual-tone multi-frequency (DTMF) key input, recording of spoken input, telephony, and mixed-initiative conversations. Its goal is to provide voice access to Web-based content and applications. It enables the development of voice applications via the use of a familiar markup style and Web server-side logic to deliver applications over telephone lines. The resulting applications allow conversational access to Web-based data, and can also interact with existing back-end business data and logic.

VoiceXML supports dialogs that feature:

- Recognition and scoping of spoken input
- Dual-tone multi-frequency (DTMF) input
- Recording of spoken input
- Synthesized speech output
- Prerecorded digitized audio output
- Dialog flow control
- ANI and DNIS

A VoiceXML application is capable of retrieving information from a Web server and, by making use of scripts and appropriate grammars, the application can interact with the customer through spoken words. The role VoiceXML plays in accessing and presenting customer information and a parallel with HTML are illustrated in Figure 1-3.

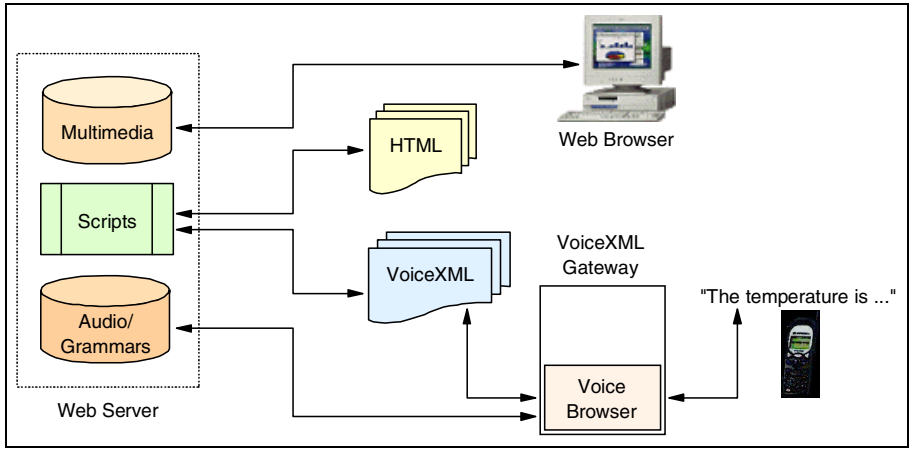


Figure 1-3 The role of VoiceXML

Advantages of VoiceXML

VoiceXML offers several important capabilities:

- Makes building voice applications easier, in the same way that HTML simplifies building visual applications. It also reduces the amount of speech expertise that developers must have.
- Uses the same existing back-end business logic as visual applications, enabling voice solutions to be introduced faster.
- Minimizes development and maintenance costs by leveraging the Web design skills and infrastructure already present. On the other hand, customers can benefit from a consistency of experience between voice and visual applications.
- Implements a client/server paradigm, where a Web server provides VoiceXML documents that contain dialogs to be interpreted and presented to the customer and whose responses are submitted to the Web server.
- Allows you to request documents and submit data to server scripts using Universal Resource Indicators. VoiceXML documents can be either static or dynamically generated by CGI scripts, JavaBeans, ASPs, JSPs, Java servlets, or other server-side logic.
- Offers an open application development environment that generates portable applications.
- VoiceXML applications are designed predominantly to accept spoken input, but can also accept DTMF input if desired. As a result, VoiceXML helps speed up customer interactions by providing a more natural interface.
- Supports networked and Web-based applications, meaning that a customer at one location can access information provided by a server at a geographically remote location.
- Allows local processing and validation of user input.
- Supports playback of prerecorded audio files. It also allows the recording of user input, where the resulting audio can be played back locally or uploaded to the server for storage or later processing.
- Defines events corresponding to activities such as a request for help, the failure of a user to respond within a timeout period, and an unrecognized user response. An application can provide catch elements that respond appropriately to a given event for a particular context.
- Accepts help requests using a system of events and catch elements. Help can be customized by specifying an event count, so that different logic blocks are executed depending on the number of times the event has occurred, and to provide increasingly detailed messages each time the user asks for help.

- Supports the use of subdialogs, the equivalent of function calls, to provide a disambiguation or confirmation dialog, and to create reusable components.

For additional information on this subject, visit the following URL: http://w3.hursley.ibm.com/~walterja/Books/Java/html/Application_Development_Using_Java_&_VoiceXML/wvraj017.htm.

1.7 Application development

This section deals with application development topics and the tools available to the developer to create voice applications.

1.7.1 Available tools

In this section, we discuss the available tools.

Voice Toolkit

The Voice Toolkit provides an integrated development environment to develop voice applications on a desktop workstation before deploying them. It offers an integrated development environment, a VoiceXML editor, editors for creating grammars, a pronunciation builder, a basic audio recorder, and a set of VoiceXML Reusable Dialog Components.

Some of its features include:

- Graphical VoiceXML application development environment.
- Source mode VoiceXML editor with support for VoiceXML 1.0 plus IBM extensions.
- Content assist in the VoiceXML editor, to check syntax and list valid elements and attributes within the source code.
- Custom pronunciation generator.
- National language support (NLS) to develop applications in any of the supported languages.
- Source mode grammar editor for JSGF and SRCL formats.
- Colorizing feature in the VoiceXML and grammar editors to highlight element tags, attributes and comments.
- Ability to generate and tune pronunciations for speech recognition and TTS.
- A basic audio recorder for creating and playing audio files.
- Reusable Dialog Components to allow adding common functions to VoiceXML applications.

Reusable dialogs

Reusable dialogs are ready-to-use VoiceXML code objects that provide basic dialog functions needed to facilitate application development. Current examples include credit-card information, e-mail addresses, and home addresses (that is, street type, US major cities and postal codes).

The dialogs receive parameters and can return values to and from other VoiceXML programs. They constitute some of the building blocks that provide the core functionality needed by VoiceXML applications.

Software Developers Kit (SDK)

The SDK is a tool that brings support for VoiceXML to Web application development activities—in essence, providing a spoken equivalent to visual browsing.

It uses the workstation's speakers to play audio output. Input can be entered using the workstation's microphone, prerecorded audio files, or the DTMF simulator, which simulates any telephone keypad input.

The SDK includes the following:

- Speech browser that interprets VoiceXML markup. This specialized browser includes a DTMF simulator.
- Speech recognition and TTS engines for accepting voice input and generating synthesized speech output.
- Telephony acoustic models to approximate the speech recognition behavior of applications deployed in a production environment.
- Tools for desktop testing of speech applications.
- A program to set up and configure microphone and speakers.
- Sample VoiceXML files and user documentation.

Voice Server speech technologies

A set of C APIs for integrating speech recognition and TTS into other telephony platforms is provided on a per-language basis.

1.7.2 Creating and deploying an application

A brief list of the typical steps involved in creating a VoiceXML application includes:

1. An application developer uses the SDK and optionally a Web development tool to create a voice application written in VoiceXML. The VoiceXML pages can be static or dynamically generated.

2. The developer publishes the VoiceXML application (including VoiceXML pages, grammar files, prerecorded audio files, and server-side logic) to the Web server.
3. The developer uses a workstation and the SDK to test the application, pointing the VoiceXML browser to the appropriate starting VoiceXML page.
4. A telephony expert configures the telephony infrastructure for the applicable deployment platform.
5. The administrator uses the deployment platform to configure, deploy, monitor, and manage a dedicated system.
6. The developer makes a call using a real telephone to test the VoiceXML application running on the Voice Server.

1.7.3 Integrating speech recognition and TTS

Speech recognition and TTS technologies can be integrated into voice applications using three different methodologies: VoiceXML, JavaBeans, and WebSphere voice response state tables.

VoiceXML

VoiceXML is a newer programming environment allowing easy integration with other XML and HTTP-type applications. VoiceXML is the ideal choice for enabling business applications for voice and to allow customer access to these applications over multiple channels.

JavaBeans

JavaBeans are an object-oriented presentation logic environment that uses common industry tools for Java development. JavaBeans can be integrated with business logic applications on an application server, and also with other JavaBeans-enabled products for database access and transaction processing.

WebSphere Voice Response state tables

State tables are a programming environment that offers low-level access to DTMF, prerecorded speech, speech recognition, fax, and Computer-Telephony Integration (CTI) products such as Genesys T-Server or CallPath.

State table actions pass commands to a supplied custom server that manages all interactions. No understanding of speech recognition technology is required, since it is all handled by the software components supplied. However, some specialized knowledge is required to develop new grammars.

1.8 Hardware technology

In this section, we briefly discuss the hardware technology (voice cards) used by WebSphere Voice Server and WebSphere Voice Response.

1.8.1 IBM

In configurations that include the use of WebSphere Voice Response for AIX V3.1 and WebSphere Voice Server for AIX, V3.1, IBM has its own hardware solution. IBM RS/6000 Systems or pSeries Systems that support the use of digital trunk adapters for connectivity to the telephony networks use the ARTIC 960RxD DTXA FC6310 adapter card.

1.8.2 Intel Dialogic

Intel acquired Dialogic Corporation in July 1999, assimilating the leading supplier of computer telephony (CT) products used by OEMs, application developers, and service providers into its communications building-block offerings.

Intel offers a broad range of Telecom hardware and software products and services - ranging from boards to server software. These help converge voice and data technologies to answer communications needs for a range of environments from enterprise organizations to service providers. Intel provides voice, fax, conferencing, and speech technologies, telephone and IP network interfaces, PBX integration products, and carrier-class board systems-level products.

For more information on Intel Dialogic products and solutions, please visit the following URL: http://www.intel.com/network/csp/trans/dialogic.htm

In configurations that include the use of WebSphere Voice Response for Windows NT and Windows 2000 and WebSphere Voice Server for Windows, refer to the General Information and Planning book GC34-5480-06 to locate the supported Intel Dialogic cards.

1.8.3 Aculab

Aculab was founded in the UK in 1978 and this remains the location of the head office. It was from this base that Aculab's commitment to global customer satisfaction was developed and spurred the opening of further offices in the US, Germany and Australia.

Through a customer-focused approach to development, Aculab has produced a computer telephony (CT) product portfolio that satisfies the DSP resource and

global digital connectivity requirements of developers and system integrators. CT applications utilizing Aculab's components can handle real-time telephony through an extensive range of resources and signaling systems.

The evolution of Aculab's generic API to provide a consistent programming interface reduces the integration time of a combination of technologies including text-to-speech (TTS), automatic speech recognition (ASR), conferencing, and fax. Collectively through supporting a broad range of industry standards and operating systems, Aculab can provide the products needed when quality and performance cannot be compromised.

For more information on Aculab products and solutions, please visit the following URL: http://www.aculab.com/company_main/company_main.htm.

In configurations that include the use of WebSphere Voice Response for Windows NT and Windows 2000 and WebSphere Voice Server for Windows, refer to the General Information and Planning book GC34-5480-06 to locate the supported Aculab cards.

2

IBM voice solutions

In this chapter, we discuss the latest voice solutions available from IBM.

2.1 WebSphere Voice Server

WebSphere Voice Server for AIX and Windows, V3.1 is a software product that can be integrated with other software and hardware telephony products to speech-enable these products.

A customer using an application developed with WebSphere Voice Server places a telephone call from either a landline or mobile phone.

- A telephony server answers this telephone call, which is a PC or UNIX server with special telephony hardware that allows it to answer telephone calls.
- The telephony server is also running WebSphere Voice Server, which listens to the user speaking and recognizes words.
- These recognized words are then passed to the application, running on WebSphere Application Server, which performs the desired transaction and sends the resulting text back to WebSphere Voice Server.
- WebSphere Voice Server then uses the text to synthesize speech, which the telephony server routes back to the handset where the caller can hear it.

The WebSphere Voice Server, V3.1 can be used for the purpose of developing software server applications built on open industry standards that enables businesses to voice-enable existing Web applications. This can allow customers access to voice-enabled Web applications by using a telephone.

Figure 2-1 on page 19 illustrates the connection environment for WebSphere Voice Server.

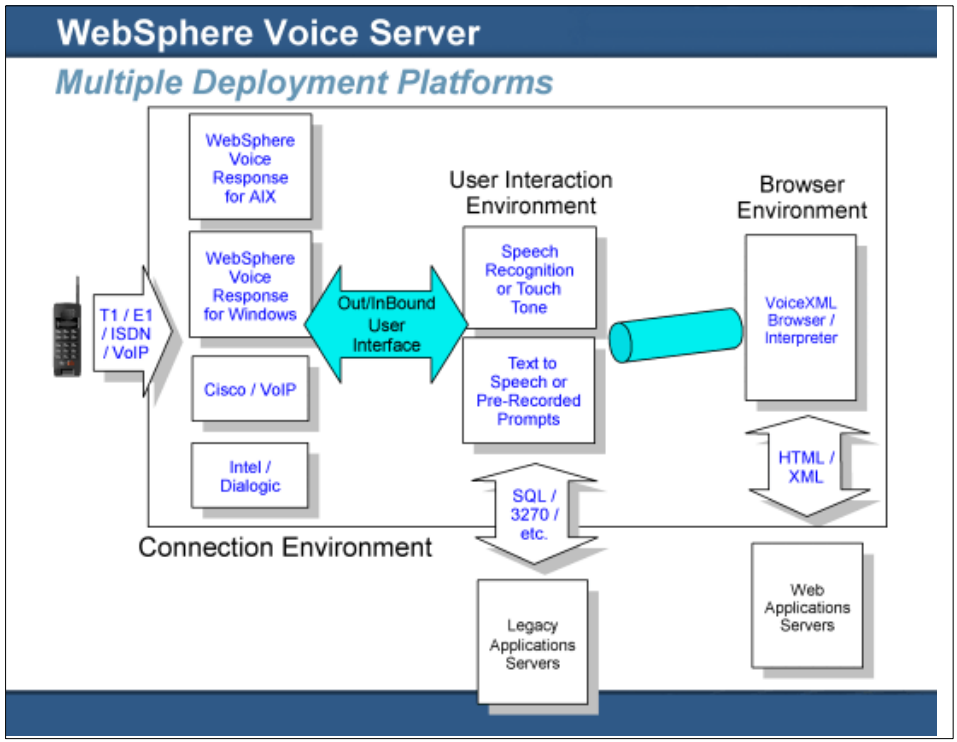


Figure 2-1 WebSphere Voice Server connection environment

2.1.1 Voice-enabling Web applications

WebSphere Voice Server provides the following features and service to voice-enable Web applications:

- A combination of products that includes:
  - A VoiceXML browser that interprets VoiceXML markup
  - IBM speech recognition and TTS engines for accepting voice input and generating synthesized speech output
  - *WebSphere Voice Server Administrator's Guide*
  - A sample VoiceXML application for testing your installation and configuration
  - A support for system management functions, not available on WebSphere Voice Response for AIX
  - Shipment with an SDK for application development

- Support software for integration with the following telephony platforms:
  - WebSphere Voice Response for AIX, V3.1 with PTF U483599 installed (formerly known as IBM DirectTalk for AIX)
  - WebSphere Voice Response for Windows V3.1 with CSD IP22517 installed
  - Cisco VoIP gateways using H.323 V2 VoIP software
  - Intel Dialogic PC with Dialogic T1 telephony adapters

2.1.2 Speech-enabling IVR applications

WebSphere Voice Server can speech-enable IVR applications via the following means:

- WebSphere Voice Server includes:
  - IBM speech recognition and TTS engines for accepting voice input and generating synthesized speech output
  - Appropriate Programming APIs to access the speech technologies from the supported platforms
  - *WebSphere Voice Server Administrator's Guide*
  - A sample application for testing your installation and configuration in the appropriate programming language
  - Support for system management functions
- Support software for integration with the WebSphere Voice Response for AIX (formerly known as DirectTalk for AIX), V3.1 (with PTF U483599 installed) telephony platform
  - Provides support for channel associated signaling (CAS), Integrated Services Digital Network (ISDN), and Signaling System 7 (SS7) signaling connections
- Support software for integration with the WebSphere Voice Response for Windows V3.1 (with CSD IP22517 installed)

For more information on WebSphere Voice Server, visit the following URL:

http://www2.ibmlink.ibm.com/cgi-bin/master?xh=TsNuzMkJtlp2*G2USenGnN9332&request=announcements&parms=H%5F202%2D224&xfr=N.

2.2 WebSphere Voice Response

WebSphere Voice Response for AIX, WebSphere Voice Response for Windows, and WebSphere Voice Server are part of the IBM WebSphere Voice family portfolio of voice and call processing products, which enable business enterprises and service providers to give callers access to their business applications and data, anytime, anyplace, from any telephone.

WebSphere Voice Response for AIX enables the development and operation of automated customer service solutions for medium and large enterprises, Telcos, and service providers. Small to medium business enterprises can obtain the same benefits with WebSphere Voice Response for Windows. Clients, customers, employees, and other users can interact directly with business applications using telephones connected via public or private networks.

WebSphere Voice Response-based solutions can reduce operating costs and improve productivity with powerful, scalable solutions from 12 to thousands of voice communication ports operating in customer premises or within telecommunication networks.

WebSphere Voice Server provides speech recognition and speech synthesis, text-to-speech software for developing and deploying applications that can be used over the telephone. Leveraging existing telephony and Web infrastructures, it enables easy voice access to WebSphere e-business applications. WebSphere Voice Response products may either be used as stand-alone Interactive Voice Response (IVR) systems or with WebSphere Voice Server to provide a fully speech-enabled solution.

2.2.1 WebSphere Voice Response for AIX, V3.1

IBM WebSphere Voice Response for AIX, V3.1 is well-suited for large enterprises or telecommunications businesses. It is scalable, robust and designed for continuous 24/7 operation. A WebSphere Voice Response for AIX system can support between 12 and 480 concurrent telephone channels on a single system. Multiple systems can be networked together to provide larger configurations. Some of the features that IBM WebSphere Voice Response for AIX offers are:

- Supports from 1 to 16 E1 or T1 digital trunks on a single pSeries server
- Supports up to 1,500 ports on a single system image (SSI) clusters
- Up to 2304 telephony channels using T1 connections or 2880 telephony channels using E1 connections can be supported in a 19” rack
- Programming models: Java, VoiceXML, Native

- Includes DirectTalk simulator for application development using Java/VoiceXML
- Requires AIX V4.3
- Support for WebSphere Voice Server V2 for speech input and Text-to-Speech output
- Multiple network connectivity: public switched telephone network (PSTN), ISDN, CAS, SSu, Voice over IP (VoIP)
- WebSphere Voice Response for AIX is the base platform for the IBM unified messaging application, IBM Message Center

2.2.2 WebSphere Voice Response for Windows NT and Windows 2000, Version 3.1

WebSphere Voice Response for Windows NT and Windows 2000 is an Interactive Voice Response (IVR) product that is for users who prefer a Windows-based operating environment to run self-service applications. WebSphere Voice Response is capable of supporting simple to complex applications and can scale to thousands of lines in a networked configuration. Applications can be developed using the native development environment, or using standards-based development for Java or VoiceXML.

Some of the features that IBM WebSphere Voice Response for Windows NT and Windows 2000, V3.1 offers are:

Features:

- Scalability: supports from 4 to 120 telephony ports per server
- Systems can be networked together to provide larger configurations
- Programming models: Java, VoiceXML, Native
- Multiple network connectivity: Analog, T1, E1, and ISDN

2.3 WebSphere Voice Toolkit

The IBM WebSphere Voice Toolkit can help developers to create voice applications in less time, using a VoiceXML application development environment. WebSphere Voice Toolkit features grammar and VoiceXML editors as well as a pronunciation builder so that application developers do not need to know the internals of voice technology. It is built on Eclipse and part of the WebSphere Studio family.

The WebSphere Voice Toolkit includes:

- An integrated development environment (IDE)
- VoiceXML editor
- VoiceXML debugger
- Grammar editor
- Grammar test tool
- Pronunciation Builder
- Built-in audio recorder
- VoiceXML Reusable Dialog Components
- Audio analysis tool
- Server analysis tool

2.4 WebSphere Voice Server SDK

The IBM WebSphere Voice Server SDK (Software Developers Kit) uses VoiceXML technology to enable developers to create voice-enabled applications and test them on a desktop workstation. With these tools and the IBM WebSphere Voice Server, developers can make applications accessible with a flexible voice interface.

The WebSphere Voice Server SDK includes the following:

- VoiceXML speech browser
- IBM speech recognition engine
- IBM text-to-speech engine
- Sample applications
- Tools for developing and testing VoiceXML applications
- Support for Brazilian Portuguese, Canadian French, French, German, Italian, Japanese, Simplified Chinese, Spanish, UK English, and US English
- Optionally, download and use the IBM WebSphere Voice Server Concatenative Text-to-Speech (CTTS) Development Edition in the language you install

3

WebSphere Voice Server with WebSphere Voice Response for Windows, V3.1

In this chapter, we discussed the preparations and process we followed to successfully install and configure WebSphere Voice Server with WebSphere Voice Response for Windows, V3.1.

3.1 Setup and install process

In this section, we discuss the setup and install process.

1. Install telephony card.
   a. In the ITSO lab, we used the following voice cards:
      i. Dialogic D/120JCT-LS (analog)
      ii. Dialogic D/41JCT-LS (analog)
      iii. Dialogic D/320SC (echo cancellation)
      iv. D/240SC-T1 (digital)
      v. D/480SC-2T1, (digital - two T1 ports)
2. Install IBM Cross Platform Technologies for Windows V2.0
3. Install License Use Management software
4. Install Telephony software.

   The ITSO used Intel Dialogic software and SDK 5.1.1.
5. Configure hardware
6. Run sample program, Voice Horoscope, which comes with the Intel Dialogic software
7. Install WebSphere Voice Response V3.1
8. Install WebSphere Voice Response CSD 3.1.1
9. Configure licensing software
10. Run WebSphere Voice Response and test DTMF functionality
11. Install Language Support
12. Install WebSphere Voice Server
13. Reboot
14. Configure Voice Server

3.2 WebSphere Voice Server V3.1 prerequisites

The following are the minimum hardware requirements when used with the WebSphere Voice Response for Windows V.31 telephony platform.

- Intel Pentium 1 GHz processor, or equivalent
- CD-ROM drive
- 512 MB RAM minimum required
- 500 MB of disk space minimum required per language install

- Local network: 10/100 Mb Fast Ethernet

Note: Requires the separately orderable WebSphere Voice Response for Windows, V3.1 (with CSD IP22517 installed) product. The general WebSphere Voice Response for Windows hardware requirements are identified in the *WebSphere Voice Response for Windows General Information and Planning Guide*.

The following are the minimum software requirements when used with the WebSphere Voice Response for Windows V3.1 telephony platform.

- Microsoft Windows 2000 Server with Service Pack 2 applied
- Adobe Acrobat Reader V4.0.5, or later (included with package)
- WebSphere Voice Response for Windows, V3.1 (with CSD IP22517 installed)

Resource estimates for WebSphere Voice Server connecting network

If WebSphere Voice Server and WebSphere Voice Response are installed in different systems, connect them using a dedicated local area network (LAN). The minimum requirement is for a 100 Mbps Ethernet network or equivalent. Given that uncompressed voice represents 64 Kbps, around 480 simultaneous sessions of speech recognition or text-to-speech will use about 75% of the usable capacity of a 100 Mbps Ethernet network for voice traffic, the remaining 25% being used for responses, headers, and control information. This may be influenced by other traffic in the system, for example, the internal communication of the external recognition or synthesis system itself. If the limit of 480 simultaneous recognitions or text-to-speech syntheses possible over a single 100 Mbps Ethernet network is likely to be exceeded, use a switched fast Ethernet or switched ATM local area network for optimum performance.

Table 3-1 provides more details on TTS disk space requirements.

Table 3-1 Disk space requirements for text-to-speech

Language	Base Product Client Files (per client)	Speech Recognition and Formant Text-To-Speech (per server)	Concatenative Text-To-Speech (per server)
System Requirement (temporary files)	120 MB	120 MB	300 MB
First language	100 MB	200 MB	450 MB

Language	Base Product Client Files (per client)	Speech Recognition and Formant Text-To-Speech (per server)	Concatenative Text-To-Speech (per server)
Each additional language	100 MB	110 MB	450 MB

You can install both speech recognition and text-to-speech on the same server system. When determining the system requirements, you can use the figures given in the previous sections. However, you should calculate the memory requirements based on the total number of engines configured (for either speech recognition or text-to-speech). Similarly, you must allow sufficient processor resource for the expected number of concurrent speech recognition and text-to-speech engines.

Note: It is difficult to be precise in estimating the saturation point for a specific configuration, because it is very sensitive to the type of any external recognition system used and the nature of the voice application. If you are in doubt about these issues, contact your IBM representative who can give you advice on capacity planning.

Restriction: WebSphere Voice Server has a maximum limit of 48 concurrent lines, while WebSphere Voice Response has a maximum limit of 96 concurrent lines for T-1 and 120 for E-1.

3.3 Installation of IBM Cross Platform Technologies for Windows

Step 1: Choose the setup language

In the following steps, we install the IBM Cross Platform Technologies, Version 2.0. Once downloaded and unzipped, double-click the install.exe program, which can be found in the Java folder. The language selection window is shown in Figure 3-1 on page 29. We have chosen English for the purposes of this redbook, although there are several languages to choose from. Select your language for this installation and click **OK**.

The following languages are available:

- Chinese (Traditional)
- Chinese (Simplified)

- English
- French (Standard)
- German
- Italian

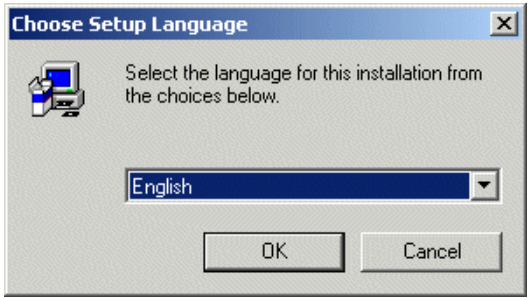


Figure 3-1 Language selection window

Step 2: Welcome

The Welcome window appears as shown in Figure 3-2. Click **Next**.

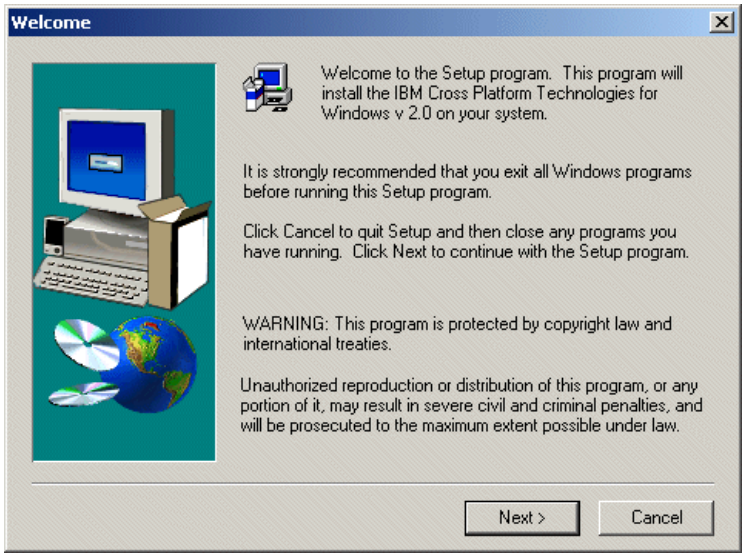


Figure 3-2 Welcome window

Step 3: License information

The License Agreement window appears, as shown in Figure 3-3 on page 30. You are required to read this. When done, click **Yes**.

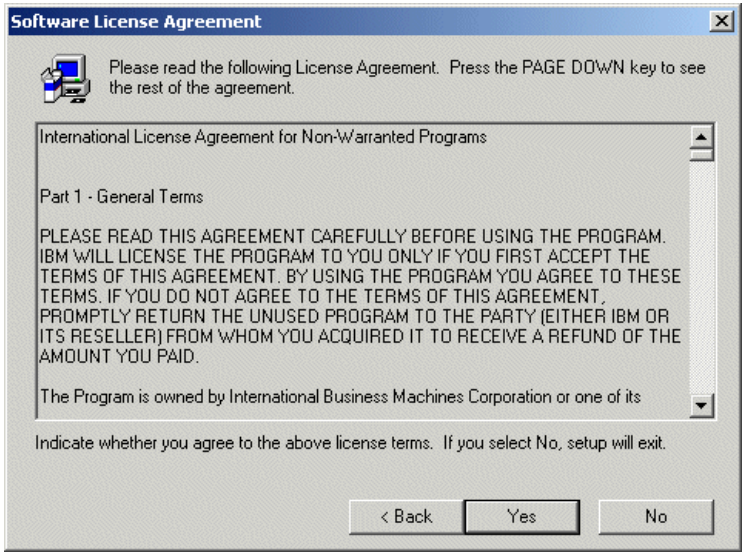


Figure 3-3 License Agreement window

Step 4: Choose destination location

You can install the Cross Platform Technologies in any preferred directory, although it is recommended that you use the default location as shown in Figure 3-4 on page 31. After you have designated the location where you would like your software installed, click **Next**.

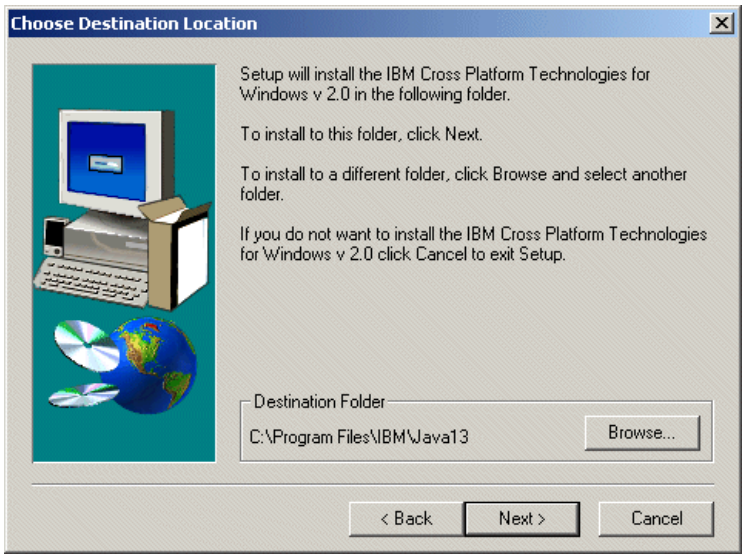


Figure 3-4 Destination selection window

Step 5: Select components

Here you will select the components you wish to install. For our purposes, we left the default selected as in Figure 3-5. Click **Next**.

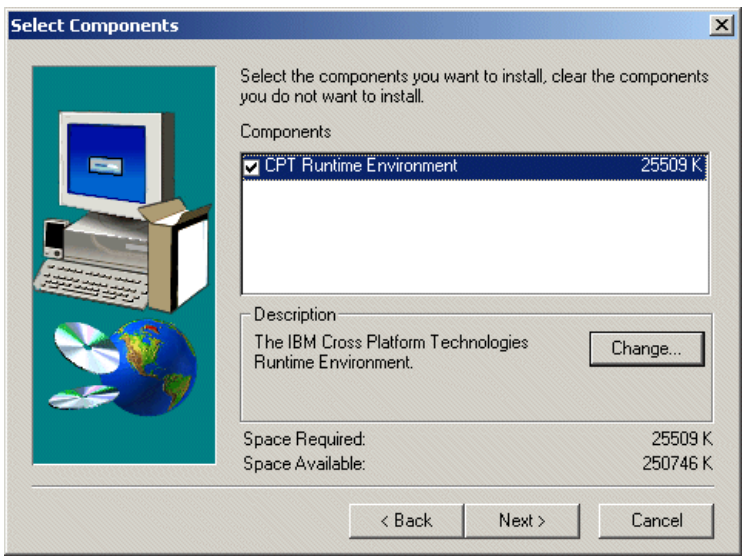


Figure 3-5 Select Components window

Step 6: Question

You will be asked if you would like to install this Java Runtime Environment as the System JVM, as shown in Figure 3-6. Click **Yes**.

Figure 3-6 Question

Step 7: Verification for your install

Before the installation occurs, you are asked to verify the options you have chosen as shown in Figure 3-7. After you have reviewed your selections, click **Next**.

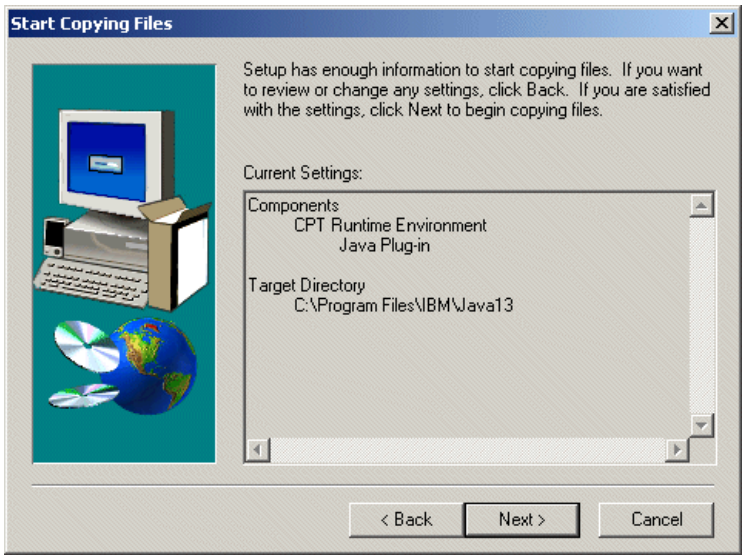


Figure 3-7 Verification

Step 8: Installation and finish

A progress bar displays the install process. Upon completion, you will see the Setup Complete window, shown in Figure 3-8 on page 33. Click **Finish**.

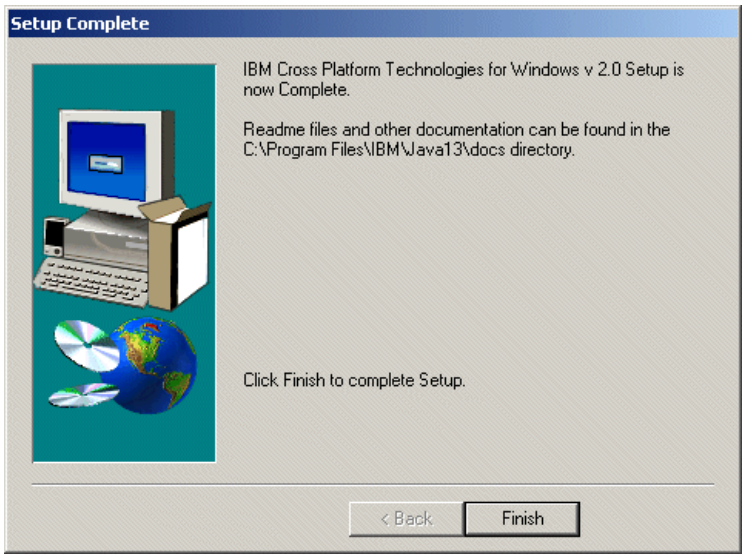


Figure 3-8 Finish

You should have successfully installed the IBM Cross Platform Technologies. Now you can install the License User Management Tool.

3.4 Installation of LUM

The License Use Management tool is necessary for the WebSphere Voice Server to communicate with the WebSphere Voice Response. (If the number of phone lines you will be using doesn't match the number of licenses distributed, errors will be thrown.)

Step 1:

Once downloaded and unzipped, run the arkwin462.exe program by double-clicking it. You will see the Welcome window as shown in Figure 3-9 on page 34. Click **Next**.

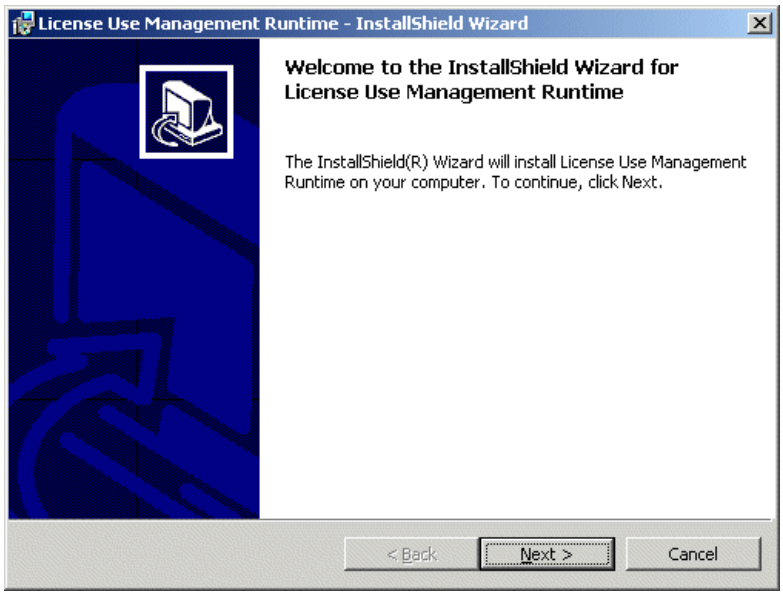


Figure 3-9 Welcome

Step 2: License agreement

The license agreement window appears, as shown in Figure 3-10 on page 35. You are required to read this. When done, select the **I accept the terms in the license agreement** and click **Next**.

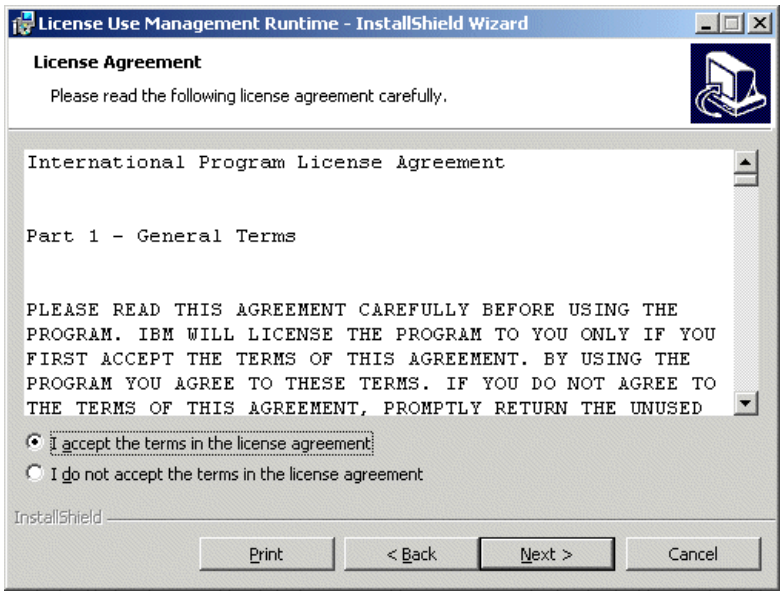


Figure 3-10 License Agreement window

Step 3: License Use Management Runtime

The window describing LUM appears, as shown in Figure 3-11 on page 36. It is strongly advised that you read this. When finished, click **Next**.

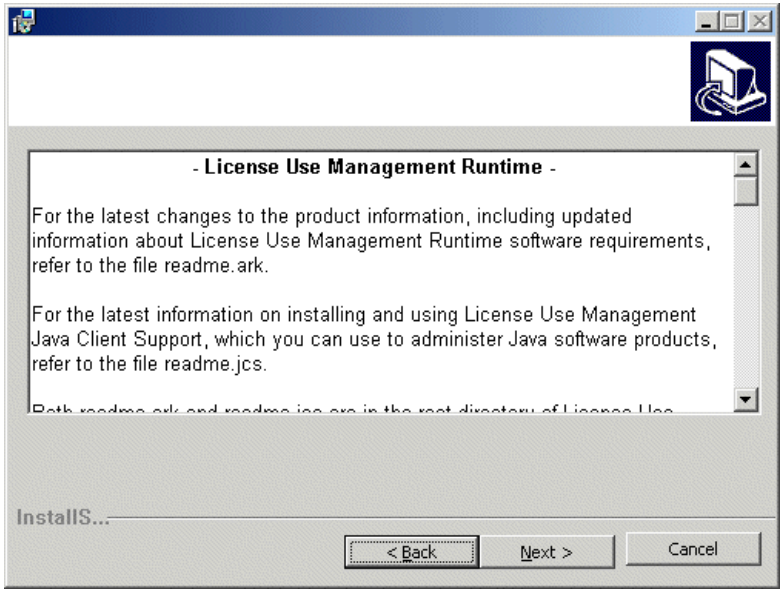


Figure 3-11 License Use Management Runtime

Step 4: Customer information

The Customer Information window appears, as shown in Figure 3-12 on page 37. Fill in the User Name and Organization test boxes, then click **Next**.

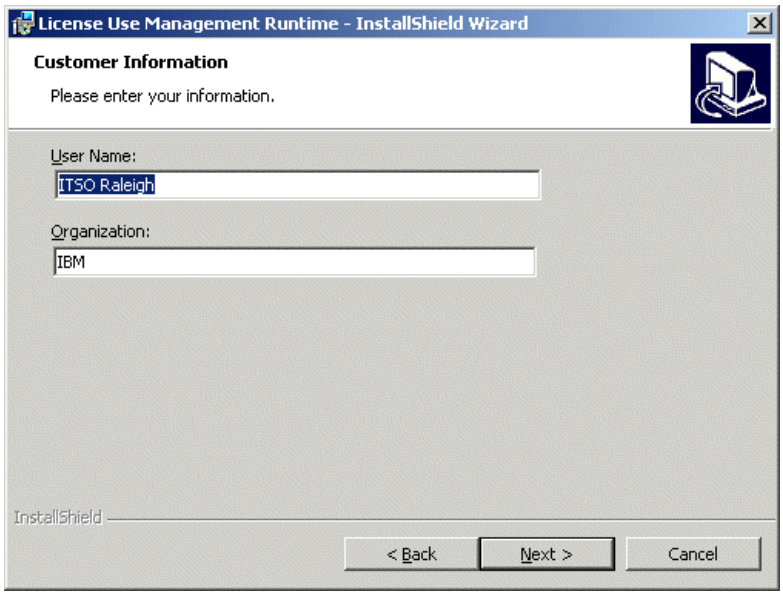


Figure 3-12 Customer Information window

Step 5: Setup type

The Setup Type window appears (shown in Figure 3-13 on page 38). For the purposes of this redbook, we have selected a complete install, and it is suggested that you do as well, then click **Next**.

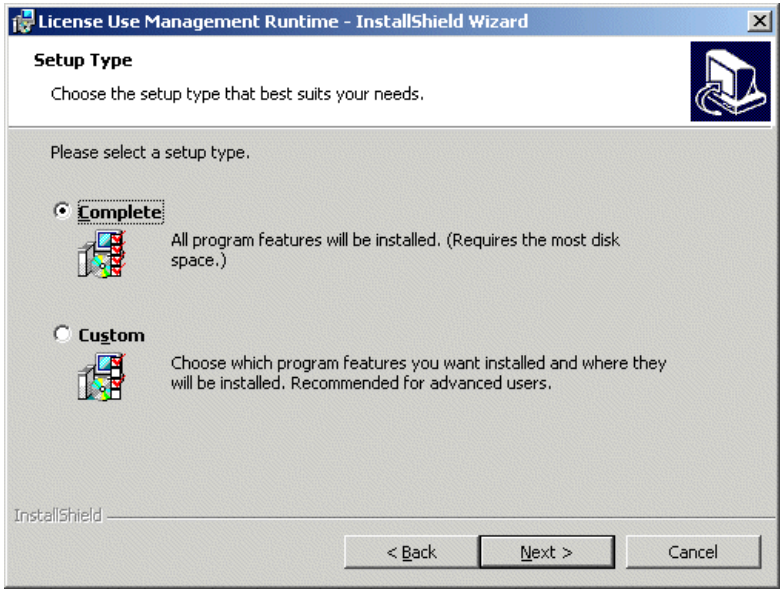


Figure 3-13 Setup Type window

Step 6: Ready to install

The Ready to install window appears (Figure 3-14 on page 39). This is the last window before the installation begins. Click **Install** to begin the installation.

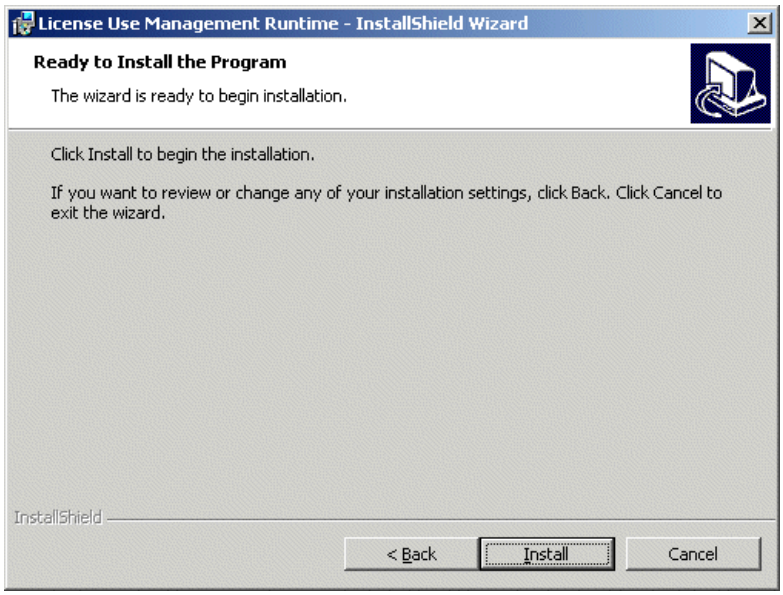


Figure 3-14 Ready to Install the Program window

Step 7: Finish

You will see Figure 3-15 when installation has completed. Click **Finish**.

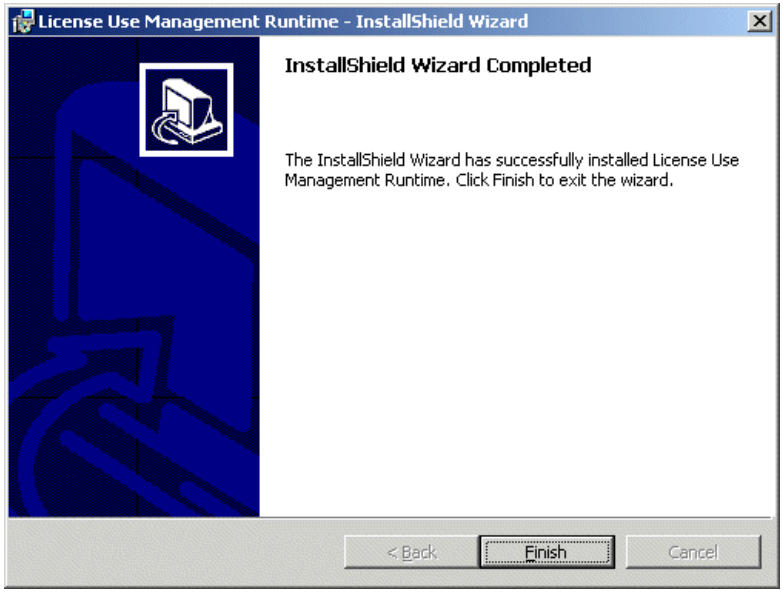


Figure 3-15 Completed window

You are now ready for the installation of your telephony software and hardware.

3.5 Installing Dialogic System Release 5.1

In these steps the Dialogic software is installed. In our document, Version 5.1 will be installed. Some of the windows may vary slightly if you install a different version of the Dialogic software.

Note: Do not use any software that may have been supplied with your Intel Dialogic cards or drivers downloaded from the Intel Dialogic Web site, as they may not be compatible with WebSphere Voice Response for Windows, Version 3.1. Use only the software supplied on the WebSphere Voice Response for Windows CD or the WebSphere Voice Response for Windows CSD.

Step 1: Starting the Dialogic software installation

From the WebSphere Voice Response for Windows CD, run the setup.exe program by double-clicking it. Click **Next**. The setup window is shown in Figure 3-16.

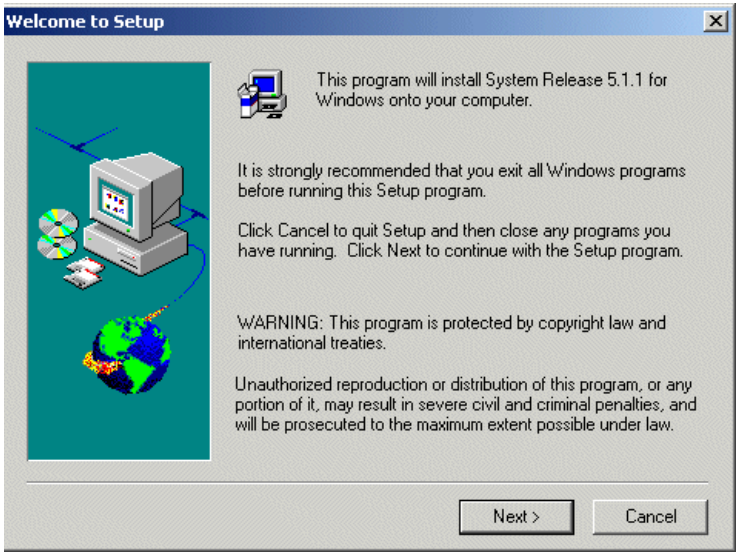


Figure 3-16 Setup window

Step 2: Release Guide

The *Release Guide* can be viewed at this point, as shown in Figure 3-17 on page 41. Unless you wish to read it now, click **No**.

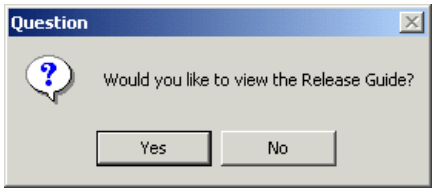


Figure 3-17 Release Guide window

Step 3: License

The license agreement appears as shown in Figure 3-18. You are required to read this. Click **Yes** when completed.

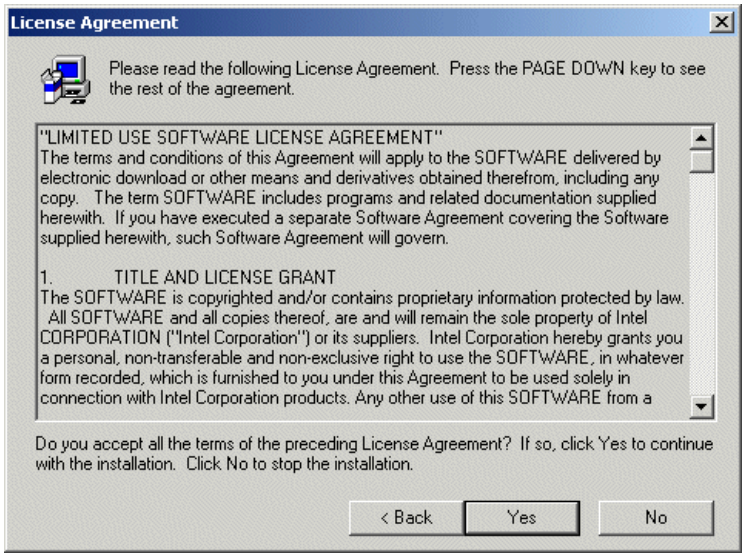


Figure 3-18 license Agreement window

Step 4: Registration

You are required to enter a name and company as shown in Figure 3-19 on page 42. Click **Next** when completed.

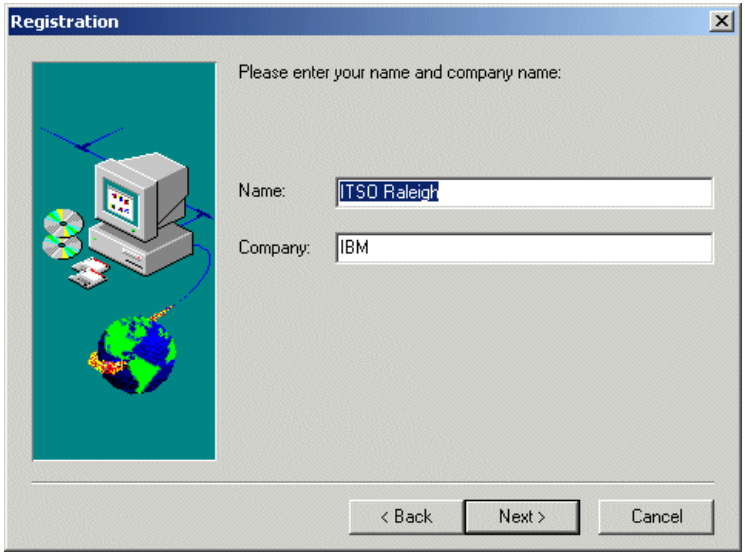


Figure 3-19 Registration window

Step 5: Setup options

The setup options give you four choices. The WebSphere Voice Response for Windows install manual Chapter 4, pp.29-32 covers the Dialogic installation. It recommends:

Choose Custom install. We recommend that you install only the following components:

- Intel Dialogic drivers, firmware and configuration files
- Intel Dialogic Development SDK
- Sample programs
- Springware TAPI Service Provider
- Online documentation
- Performance Counters for Windows NT Performance Monitor
- Antares Software (if you have a legacy Antares card).
- Third party support (if you are using an Aculab card).

This will use around 600 MB of hard drive space. Alternatively, we successfully installed using the minimum custom install. Click **Custom** on the options window shown in Figure 3-20 on page 43, then click **Next**.

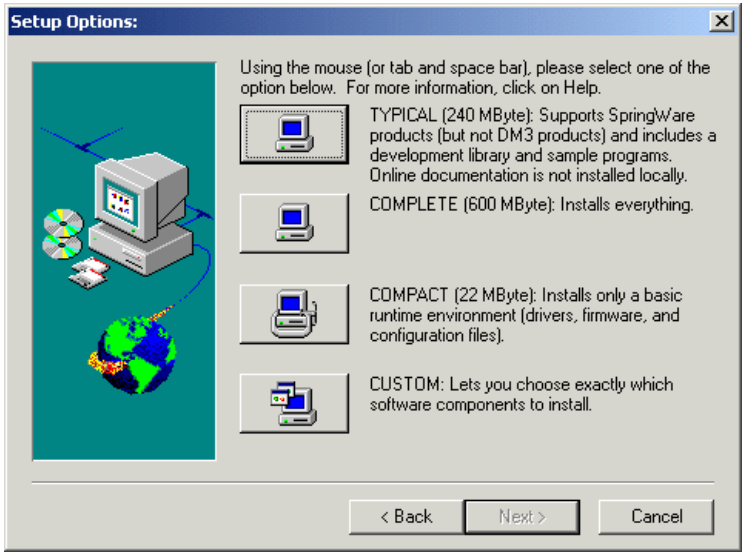


Figure 3-20 Options window

Step 6: Selection options

We chose to install the following options:

- Dialogic drivers, firmware and configuration files
- Development SDK
- Sample programs
- Springware TAPI Service Provider
- Online documentation
- Performance Counters for Win NT Perf. Monitor

Select these as illustrated in Figure 3-21 on page 44, and then click **Next**.

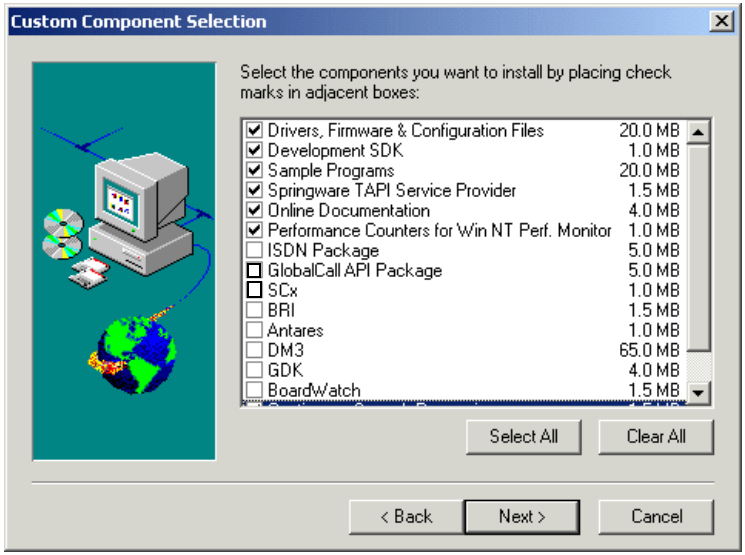


Figure 3-21 Custom Component Selection window

Step 7: Online documentation

For better use of the documentation, select **Install Documentation Locally** as shown in Figure 3-22. Click **Next**.

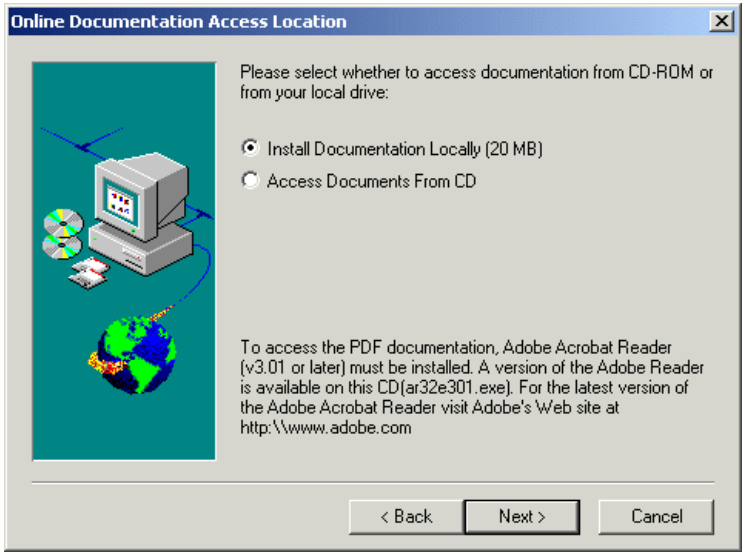


Figure 3-22 Documentation window

Step 8: Installation destination

The default directory path is displayed, as demonstrated in Figure 3-23. You may change this. We used the default. Click **Next**.

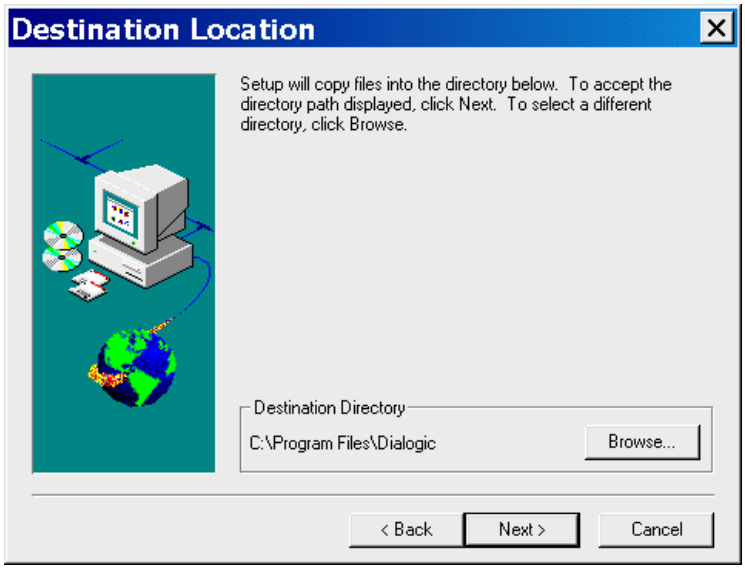


Figure 3-23 Destination window

Step 9: Program folder

The default program group folder is displayed in Figure 3-24 on page 46. We used the default. Click **Next**.

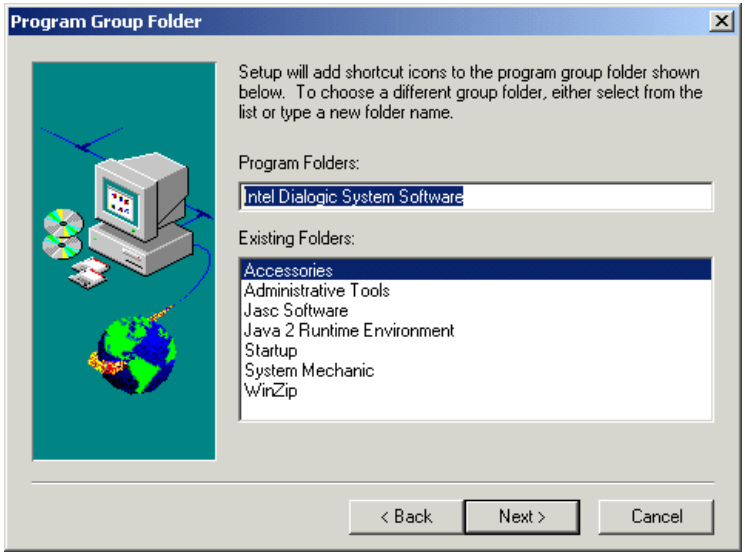


Figure 3-24 Program folder window

Step 10: Setup summary

Before the final installation, a summarization of the options is presented as shown in Figure 3-25. If these are correct, continue by clicking **Next**.

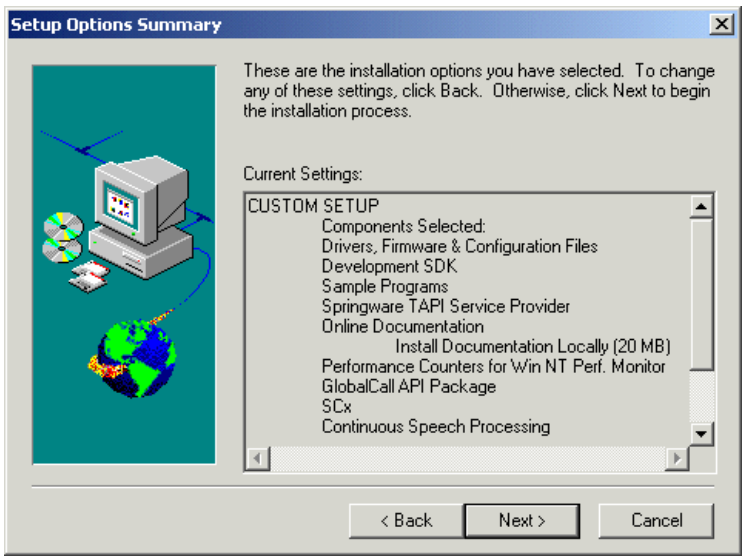


Figure 3-25 Summary window

Step 11: Dialogic Configuration Manager (DCM)

Once installation is complete, you are able to read the *Release Guide* and run the Dialogic Configuration Manager (DCM). Since we have not installed the card yet, deselect this option as shown in Figure 3-26. Click **Next**.

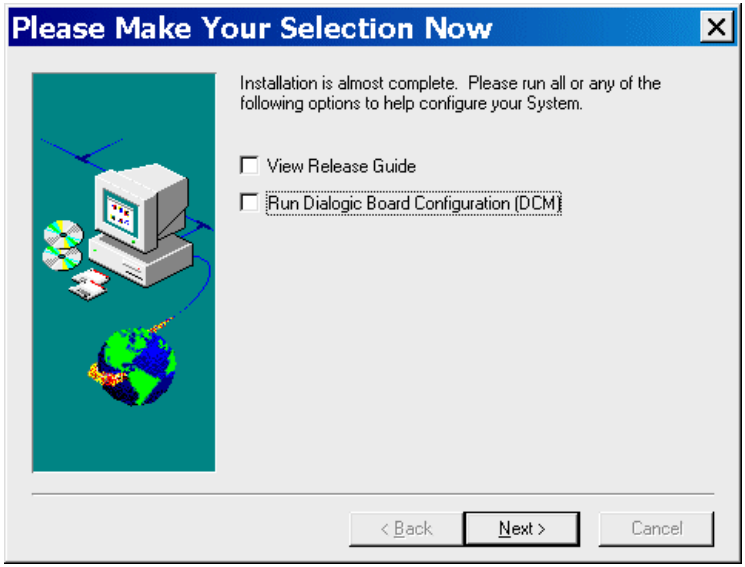


Figure 3-26 DCM window

Step 12: Performance monitor

The Dialogic software installs a performance monitor. This will not work until you reboot the system. A warning message advises you of this (see Figure 3-27). Click **OK**.

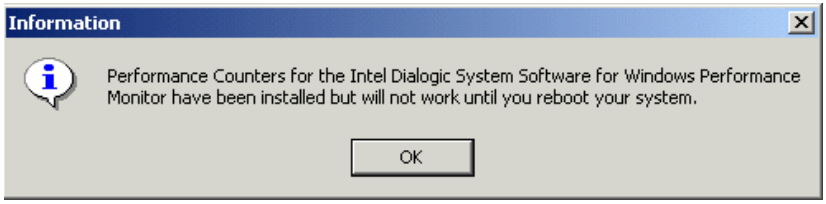


Figure 3-27 Monitor warning

Step 13: Rebooting

To ensure the Dialogic software is fully functional, you need to reboot. Click **OK** as shown in Figure 3-28 on page 48.

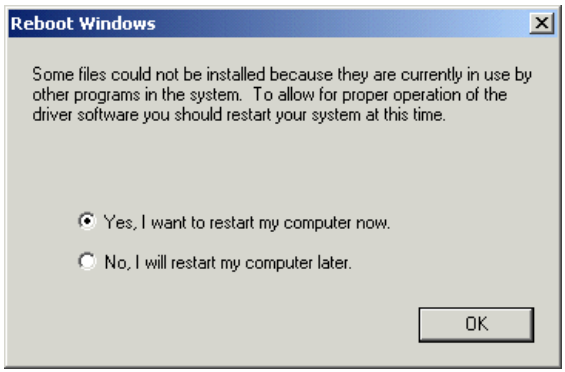


Figure 3-28 Reboot window

After rebooting, you should install the Dialogic System Release 5.1.1 Service Pack 1.

3.6 Installing Dialogic System Release 5.1.1 Service Pack 1

In this section, we installed the Dialogic System Release 5.1.1 Service Pack 1. The Service Pack was downloaded from the Dialogic Web site.

Important: At the time this installation procedure was executed and documented for this redbook, Service Pack 1 included on the WebSphere Voice Response for Windows CD was unavailable. Although the following instructions pertain to the use of the downloaded files from the Dialogic Web site, you are now instructed to use only the software supplied on the WebSphere Voice Response for Windows CD or the WebSphere Voice Response for Windows CSD. Up to three PTRs are found on the WVR Windows CD for installation. Please notice that the installation will vary from the following instructions.

These are the steps to complete the installation of Service Pack 1.

Step 1: Run setup.exe

From the WebSphere Voice Response for Windows CD, run the setup.exe program. Click **Next** as in Figure 3-29 on page 49.

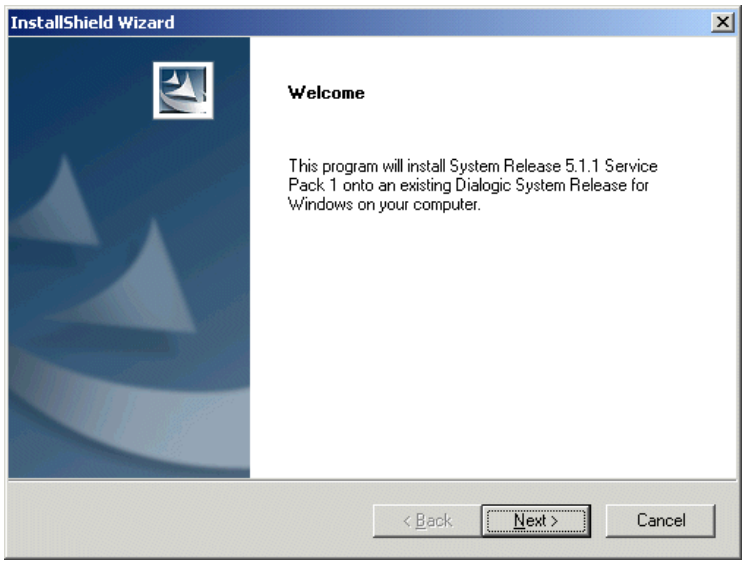


Figure 3-29 Install window

Step 2: License agreement

The license agreement is presented as shown in Figure 3-30. Click **Yes**.

Figure 3-30 License window

Step 3: Registration

Ensure you use the same name and company you entered when installing the base code of 5.1.1 as shown in Figure 3-31. Click **Next**.

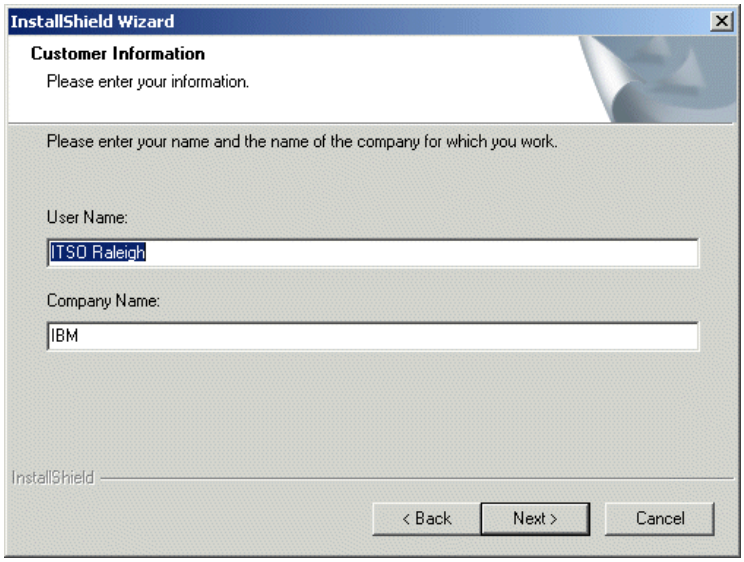


Figure 3-31 Registration window

Step 4: Reboot

Once completed the reboot option appears, as illustrated in Figure 3-32 on page 51. The system must be rebooted to continue. Click **Finish**.

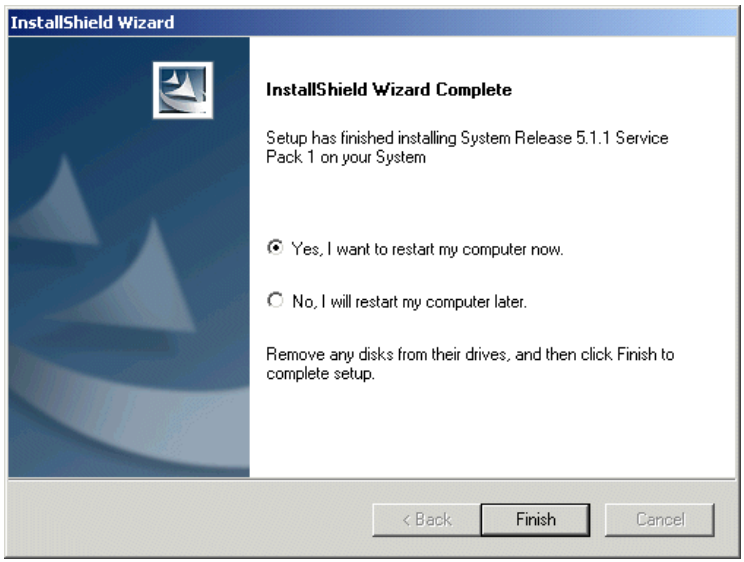


Figure 3-32 Reboot window

After rebooting you will move through the hardware detection windows until you reach the device driver window, as illustrated in Figure 3-33. Windows 2000 will try to determine what drivers match the card. Click **Next**.

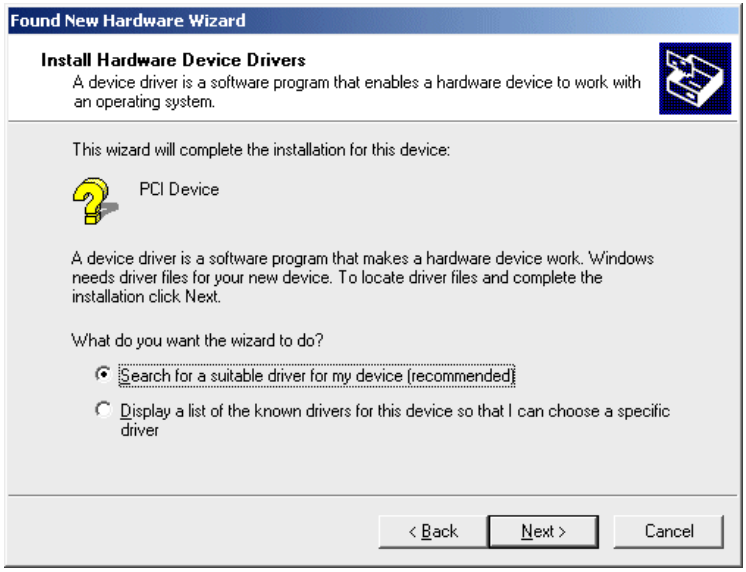


Figure 3-33 Device driver window

The Dialogic software provides a set of drivers. We select the option **Specify a location**, as shown in Figure 3-34. Click **Next**.

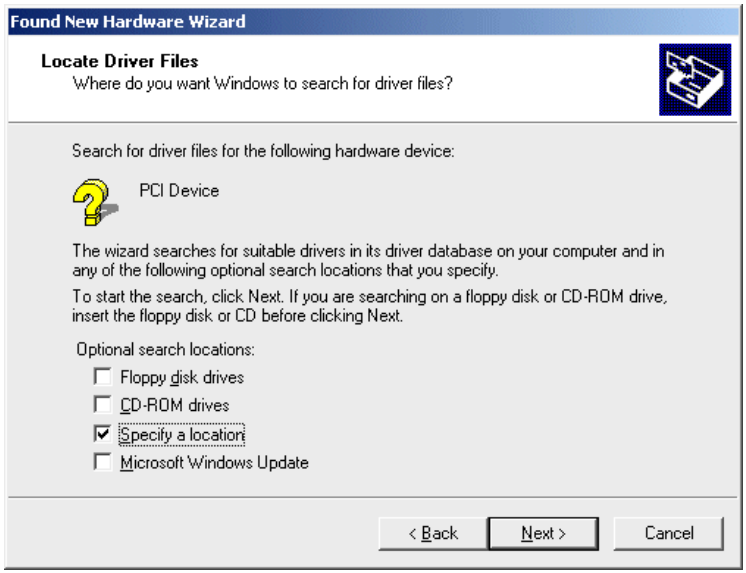


Figure 3-34 Specify location window

We need to manually point to the driver, so click **Browse**, as shown in Figure 3-35.

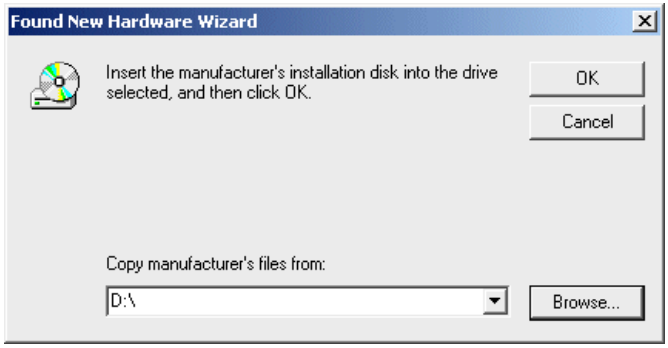


Figure 3-35 Browse window

The drivers are located in the DRV subdirectory of the installed Dialogic software. In our case this is the default folder. Highlight this folder and click **Open**, as in Figure 3-36 on page 53.

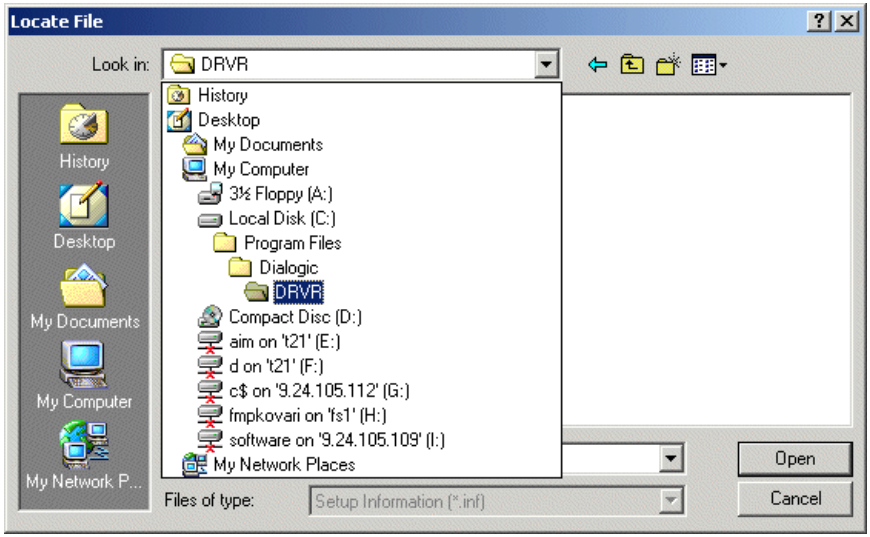


Figure 3-36 Dialogic driver location window

You are presented with a set of drivers. For the D/120JCT card, select the **dlgcsram_nt4.inf** file, then click the **Open** button, as shown in Figure 3-37.

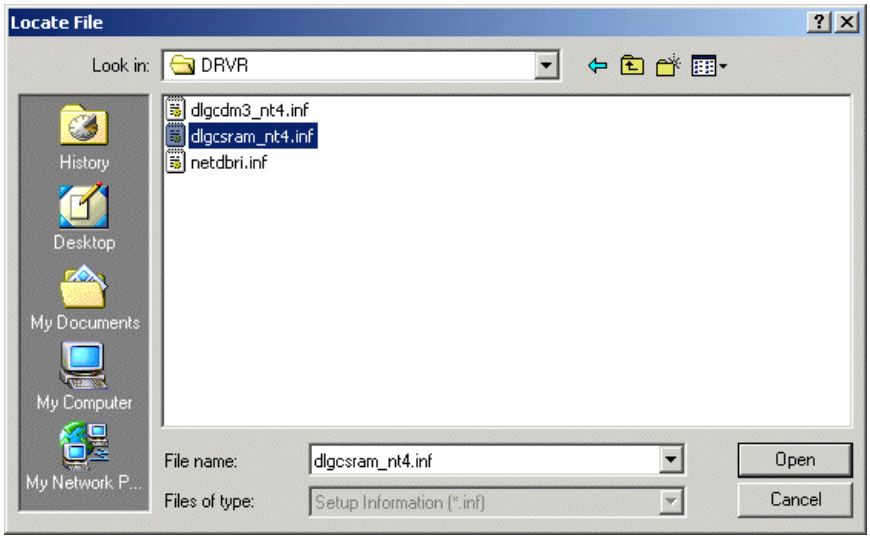


Figure 3-37 Dialogic driver window

A confirmation window appears, as in Figure 3-38 on page 54. Click **OK** if correct.

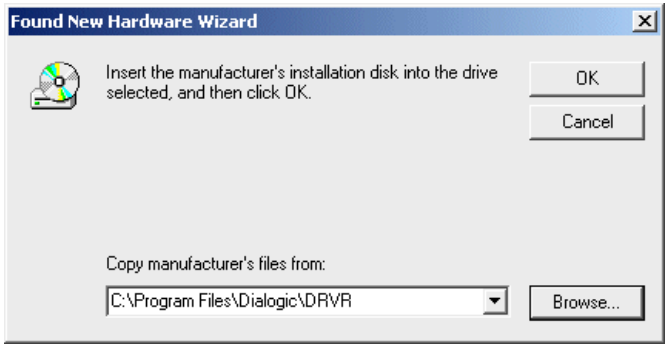


Figure 3-38 Driver confirmation window

The Hardware wizard indicates the driver it found (Figure 3-39). Click **Next**.

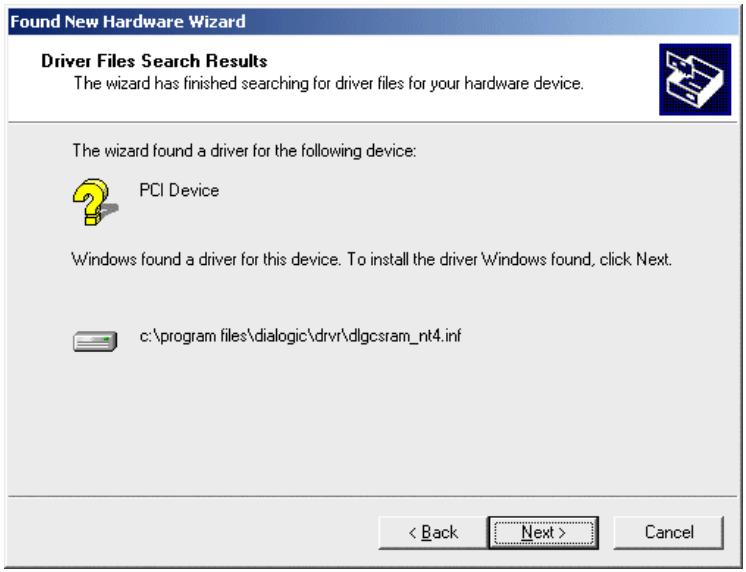


Figure 3-39 Drive found window

The Hardware wizard presents a completed window that looks like Figure 3-40 on page 55. Click **Finish**.

Figure 3-40 Finish window

3.6.1 Dialogic card configuration

Before installing Voice Server, the Dialogic card needs to be configured. These settings can be broken down into two sections: Global settings and specific country settings.

To configure the Dialogic card, use the Dialogic Configuration Manager (DCM) in the Dialogic application folder. Click **Start -> Programs -> Intel Dialogic System Software -> Configuration Manager-DCM**. First the application must connect to the machine, specified by the host name given to the machine. In our lab, 2000Server was the machine name where we installed the Dialogic card. This is shown in Figure 3-42 on page 56.

To set the Service to start automatically upon bootup, click **Service -> Startup Mode -> Automatic**, as shown in Figure 3-41 on page 56.

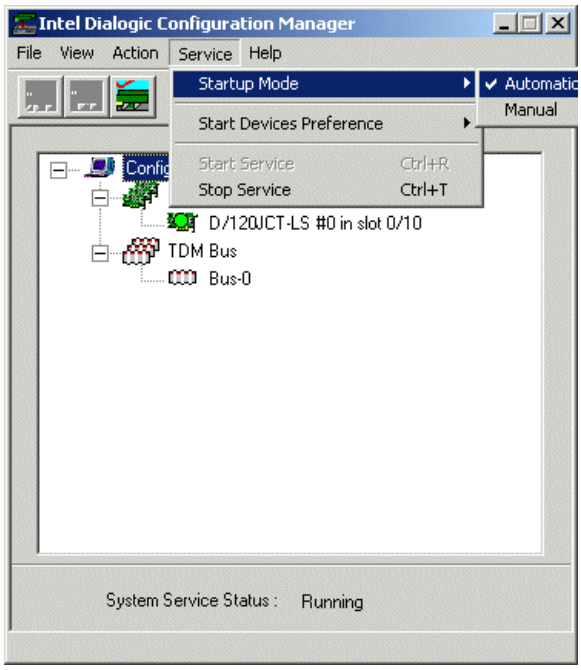


Figure 3-41 Starting up automatically

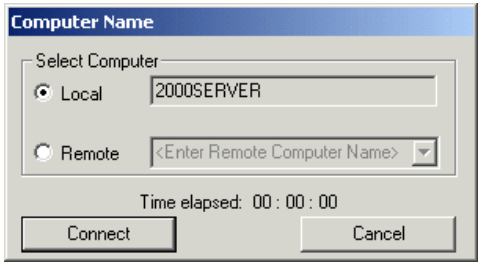


Figure 3-42 DCM connection window

The DCM window will show you the installed Dialogic card and the Time-Division-Multiplex (TDM) Bus. Figure 3-43 on page 57 shows the DCM window.

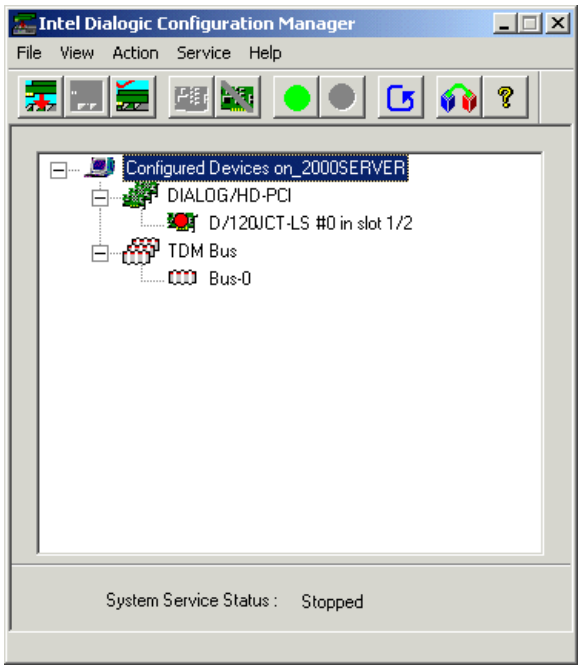


Figure 3-43 DCM window

Right-click the installed card. In this case, it is a D/120JCT-LS. Select **Configure Device**, as illustrated in Figure 3-44 on page 58.

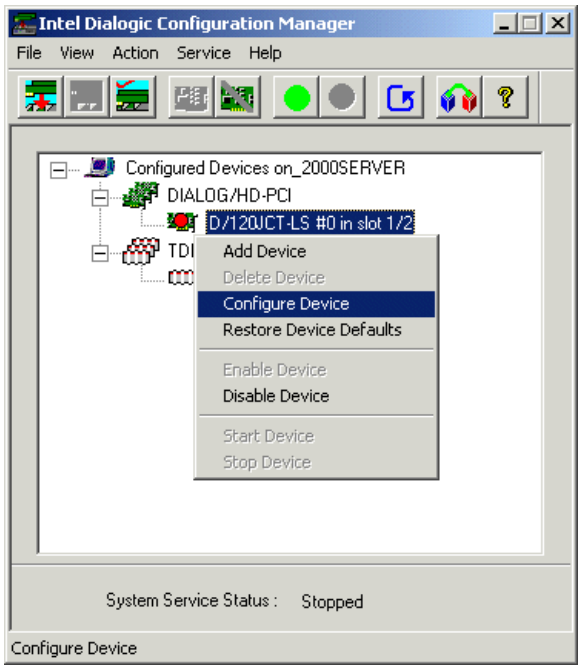


Figure 3-44 Configure Device option window

The properties window for the selected card is displayed. Changes are required on the Country tab. Set the parameter values for the D/120JCT-LS board as shown in Table 3-2.

Table 3-2 DCM settings table

At this tab	Ensure that this parameter	Has this value
Country	Country	*your country*

Select the **Country** tab and ensure you have your country selected. Click **OK**, as in Figure 3-45 on page 59.

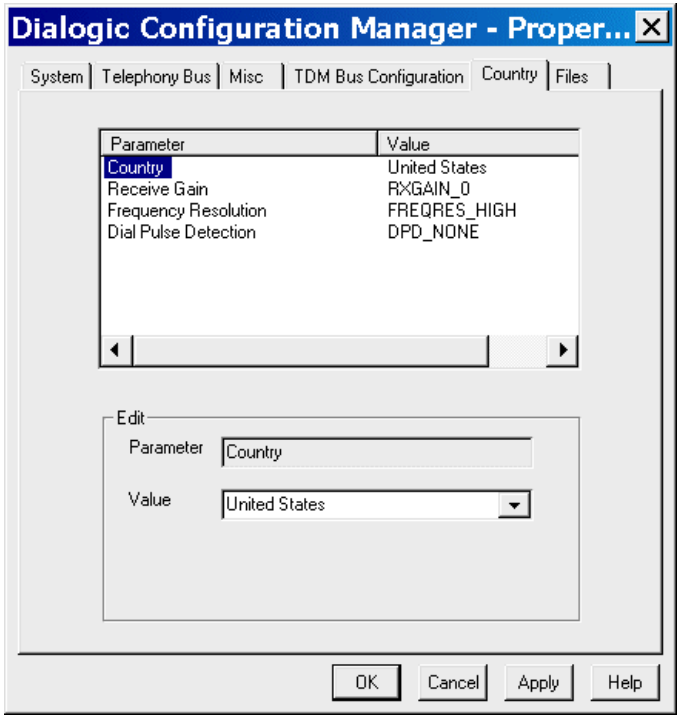


Figure 3-45 Country parameter window

Now right-click the **Bus-0** under the TDM Bus Configuration tab and select **Configure Device**, as illustrated in Figure 3-46 on page 60.

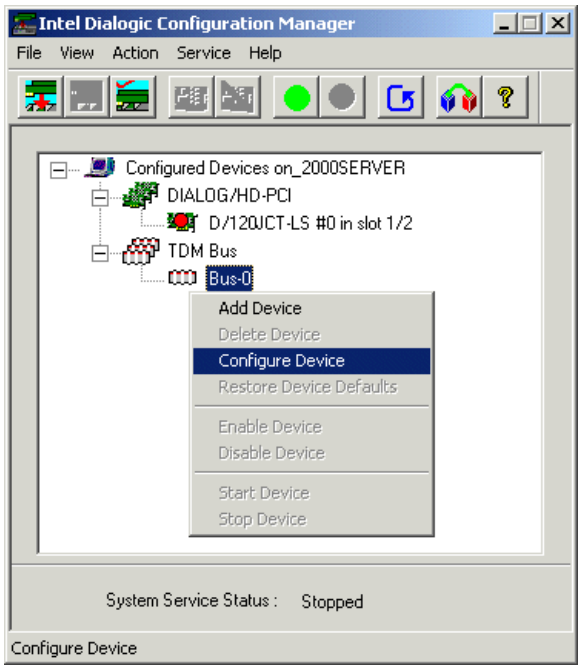


Figure 3-46 TDM Bus selection

The Bus configuration window will appear. One parameter needs to be changed here, as shown in Table 3-3.

Table 3-3 TDM Bus parameter

At this tab	Ensure that this parameter	Has this value
TDM Bus Configuration	TDM Bus Type (User defined)	SCBus

Click **Apply** to save the settings, then click **OK** when done, as shown in Figure 3-47 on page 61.

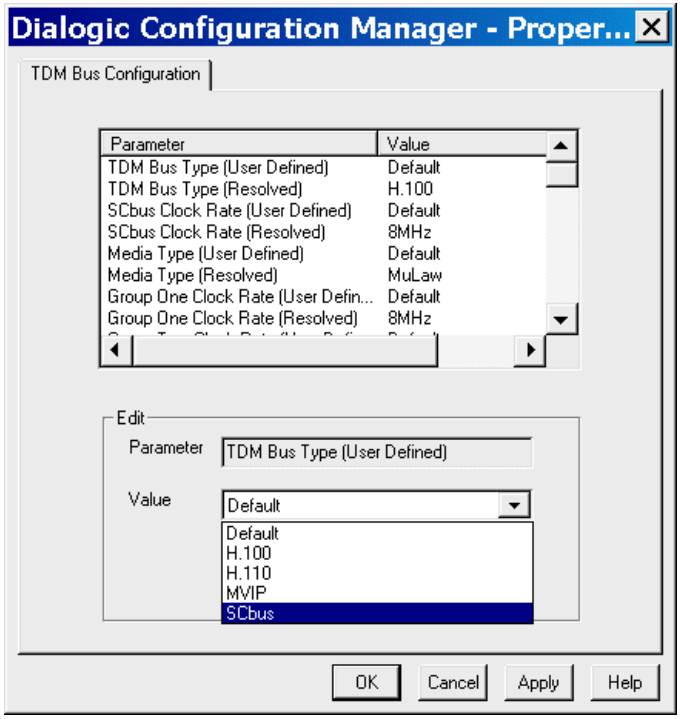


Figure 3-47 TDM Bus configuration window

Start the card to test the configuration by clicking the green circle, as in Figure 3-48 on page 62. Once completed, the card will be active. The card has a green circle on it to indicate it has been started.

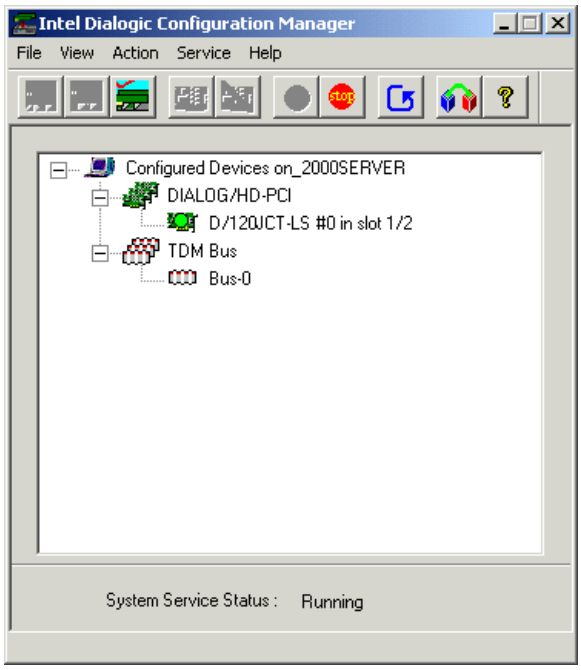


Figure 3-48 Started DCM window

3.6.2 Testing the card

We tested the card by running a sample application. Make sure that the telephone cables are connected to the card. In our scenario, two analog lines were connected to the Dialogic card. Start the Horoscope program found in the sample directory in the Dialogic folder, as shown in Figure 3-49 on page 63.

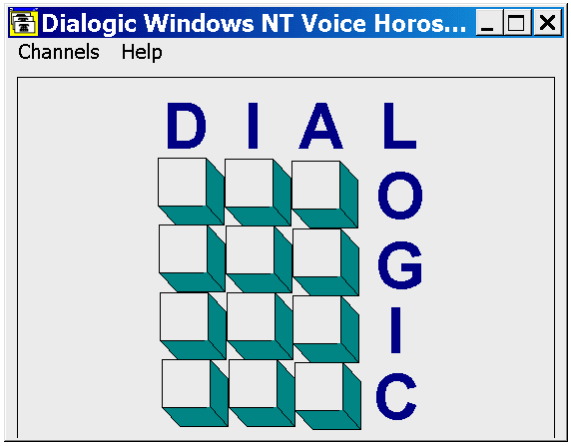


Figure 3-49 Horoscope window

The Horoscope program opens so incoming calls can be received. To do this, click **Channels -> Open**, as shown in Figure 3-50.

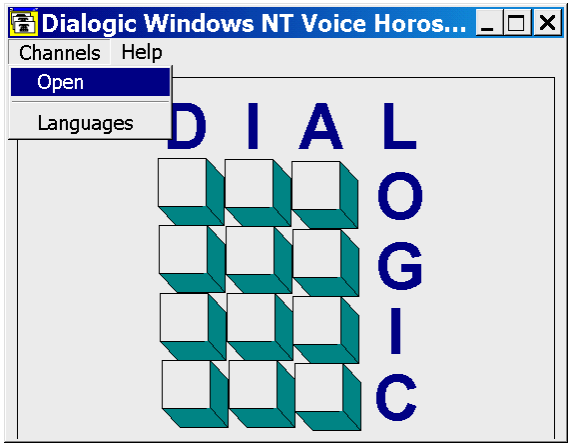


Figure 3-50 Channel opening window

Once the channels are opened, calls can be made. Four lines are ready and waiting. When a call is received, the phone appears off the hook, as shown in Figure 3-51 on page 64.

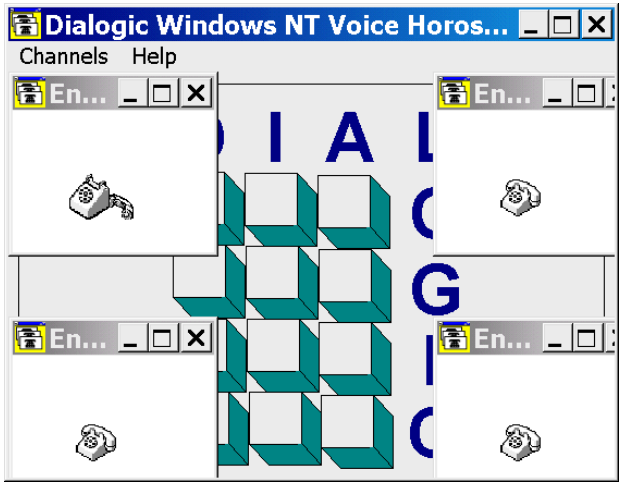


Figure 3-51 Call received window

The user is required to enter in data via dual tone multi-frequency (DTMF). If successful, the display will change to reflect the input, as illustrated in Figure 3-52.

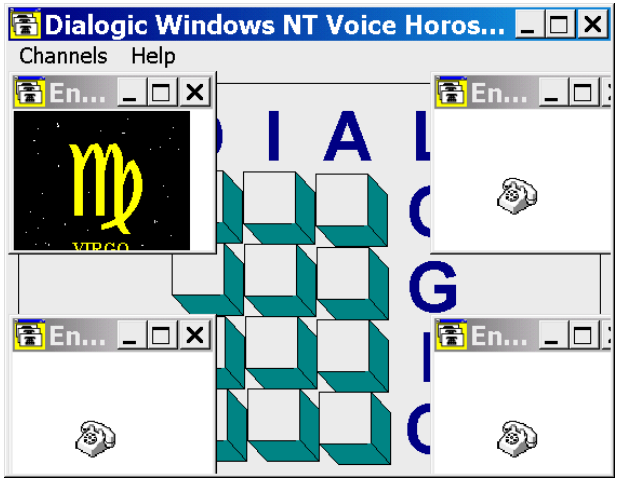


Figure 3-52 Data entered window

3.7 Installation of WebSphere Voice Response

After successfully running the Voice Horoscope application, you can begin the installation of WebSphere Voice Response.

Note: Before starting the installation, it is important that the Dialogic services are started. If you install WebSphere Voice Response while the Dialogic services are down, you may encounter errors after the installation.

Step 1: Run setup

Once downloaded and unzipped, run the setup.exe program. You will see the Welcome window. Click **Next** as shown in Figure 3-53.

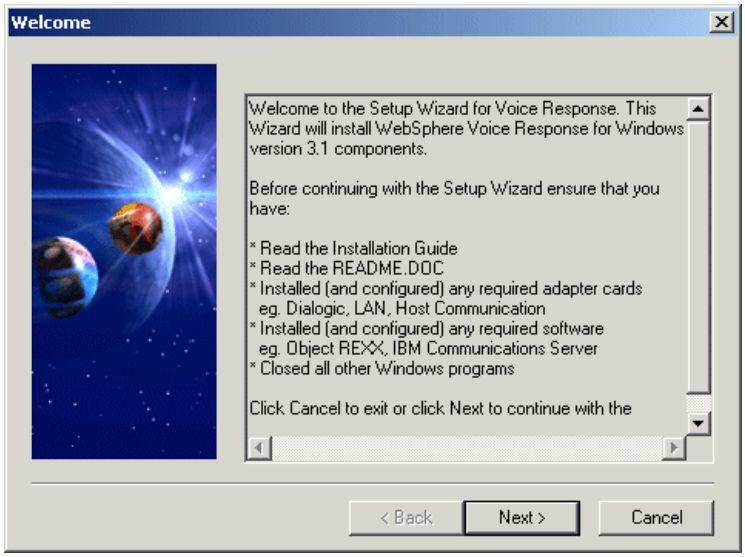


Figure 3-53 Welcome

Step 2: System selection

The System Selection window will be next. In the ITSO, we installed the Runtime System. Click **Next**, as shown in Figure 3-54 on page 66.

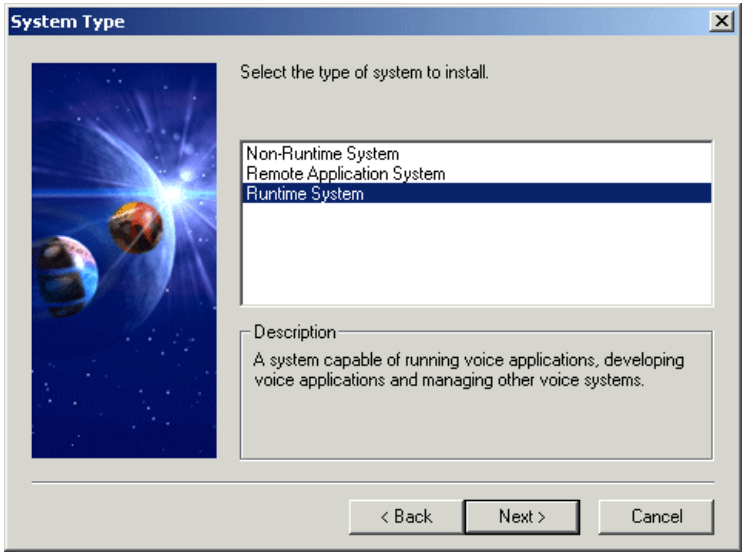


Figure 3-54 System Type

Step 3: Select destination

The next window will ask you where you would like to install the WebSphere Voice Response. The default is to install it in C:\dtalk (Figure 3-55 on page 67). Unless you have a good reason, it is recommended that you leave the default selection. Click **Next** when done.

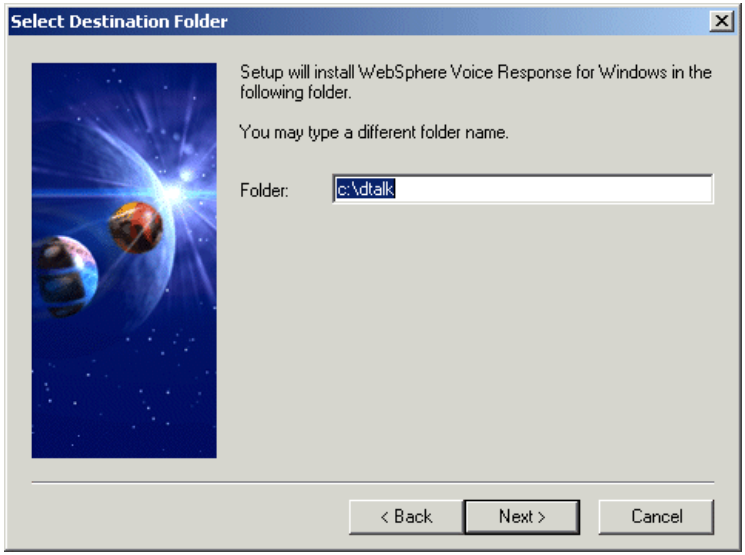


Figure 3-55 Destination Folder window

Step 4: Components

The next window is the Select Components section of the install. In the lab, we installed all components as in Figure 3-56 on page 68. If you want to examine or change the components being installed, click **Change**. Otherwise, install all components by clicking **Next**.

Note: In order to change the default installation location and/or languages installed from the default based upon your system locale settings, highlight the Region Parameter Set component and click **Change**. You will then see a list of regions to choose from. Find the region you wish to use and check the respective box. You may then repeat this process with the Voice Language(s) component. Be sure to also de-select what has already been selected, unless you want more than one language installed. After you have made the appropriate changes, click **Continue**.

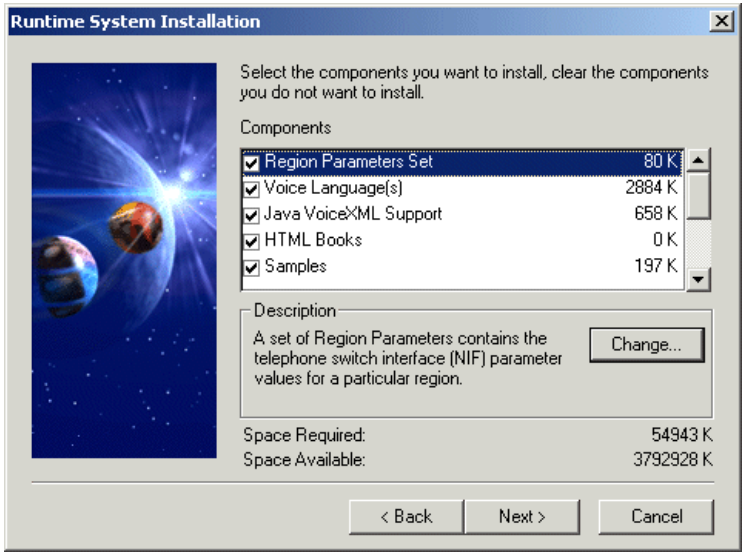


Figure 3-56 Install components

Step 5: Language verification

The information window at this step informs you of the language selection you have made. If you have chosen a language that is not supported by the Voice Response Java/VoiceXMl component, the system will default to the first supported voice language chosen, oe en_US if none are supported. See Figure 3-57. Verify this is the correct language you want to configure the system and click **OK**.

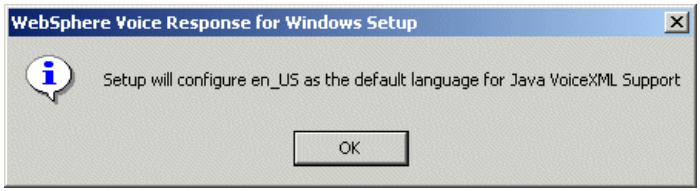


Figure 3-57 Language verification

Step 6: Migration

You will now be asked if you have a previous installation on your system. You should not have a previous installation, but if you do select **Automatically search for an old installation**. In the lab, we made sure that we did not have a previous installation. See Figure 3-58 on page 69. Click **Next**.

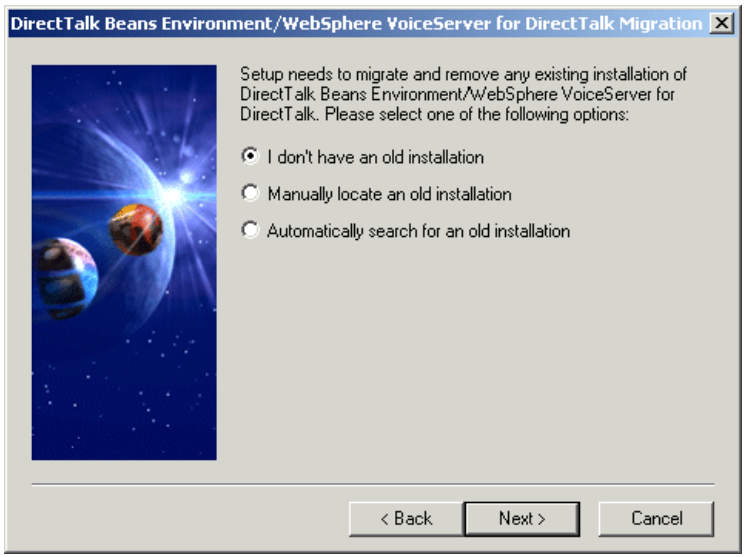


Figure 3-58 Migration window

Step 7: Setup verification

At this point, you have an option to review your selections before they are installed. After you have reviewed your selections, click **Next**, as shown in Figure 3-59.

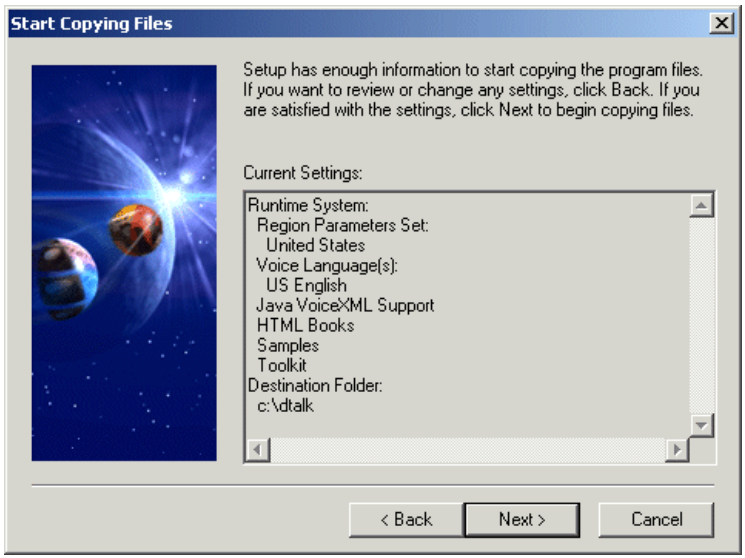


Figure 3-59 System verification

Step 8: Finish

After the installation is complete, you will see the window shown in Figure 3-60. Click **Finish**.

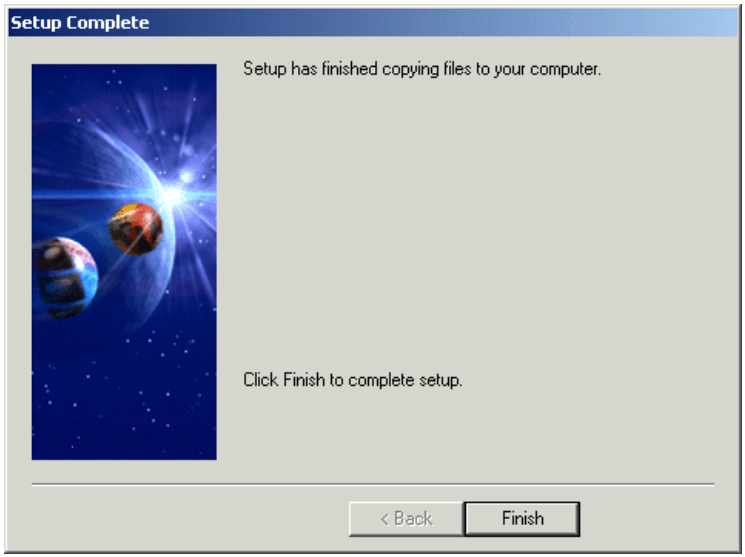


Figure 3-60 Finish

3.7.1 WebSphere Voice Response ServicePak 1

In this section, we will install the WebSphere Voice Server ServicePak 1.

Step 1: Installing

Once downloaded and unzipped, run the setup.exe program. You will see the Welcome window. Click **Next** as in Figure 3-61 on page 71. This ServicePak can be found at http://www-3.ibm.com/software/pervasive/products/voice/universalaccess_2_121.shtml.

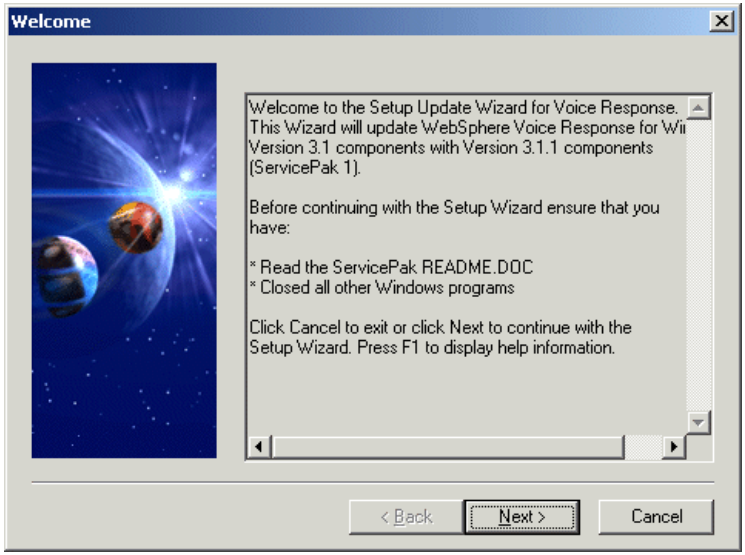


Figure 3-61 Welcome

Step 2: Backup

This information window asks you if you would like to back up the files that are updated by this service. Although in the lab we haven't had to back up the ServicePak, it is recommended that you do back up the files. Click **Yes**, as in Figure 3-62.

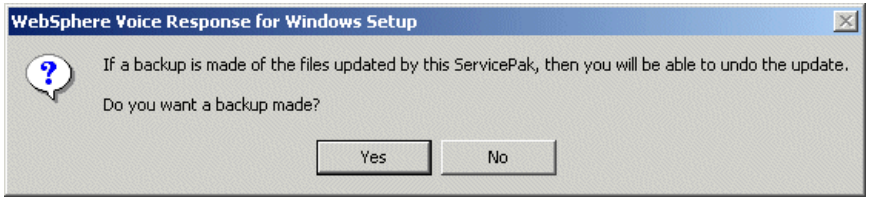


Figure 3-62 Backup

Step 3: Finish

After the files have been backed up and updated, you will see the Setup Complete window. Click **Finish** as shown in Figure 3-63 on page 72.

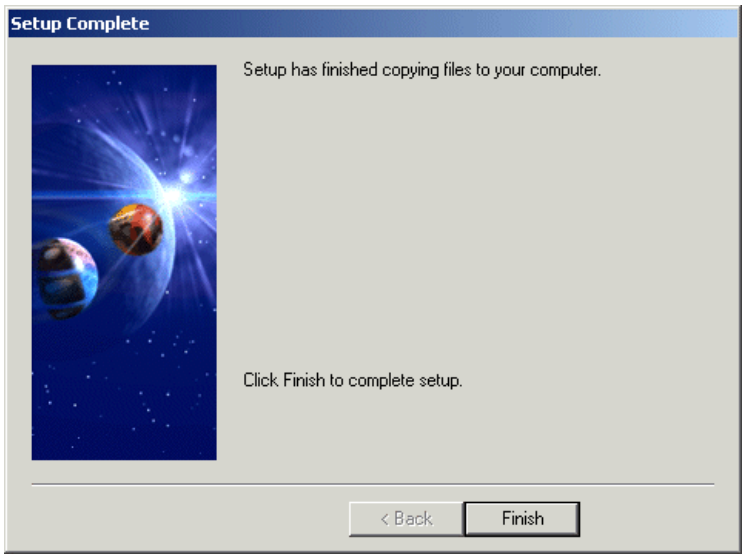


Figure 3-63 Finish

3.8 Configuring licensing software

In this section, we will configure the licensing software.

3.8.1 Configuring LUM for stand-alone setup

Perform the following steps to configure LUM.

Step 1: Configuration Tool

Before testing the installation of the WebSphere Voice Response, it is advised that you first configure the LUM system that was installed earlier. To do this, click **Start -> Programs -> License Use Runtime -> Configuration Tool** as shown in Figure 3-64 on page 73. NodeLocked License Server will be selected by default. Unselect **NodeLocked License Server** and select **Network License Server (NetworkLS) and Central Registry License Server (CrLS)** only. Click the **Direct Binding** tab when finished.

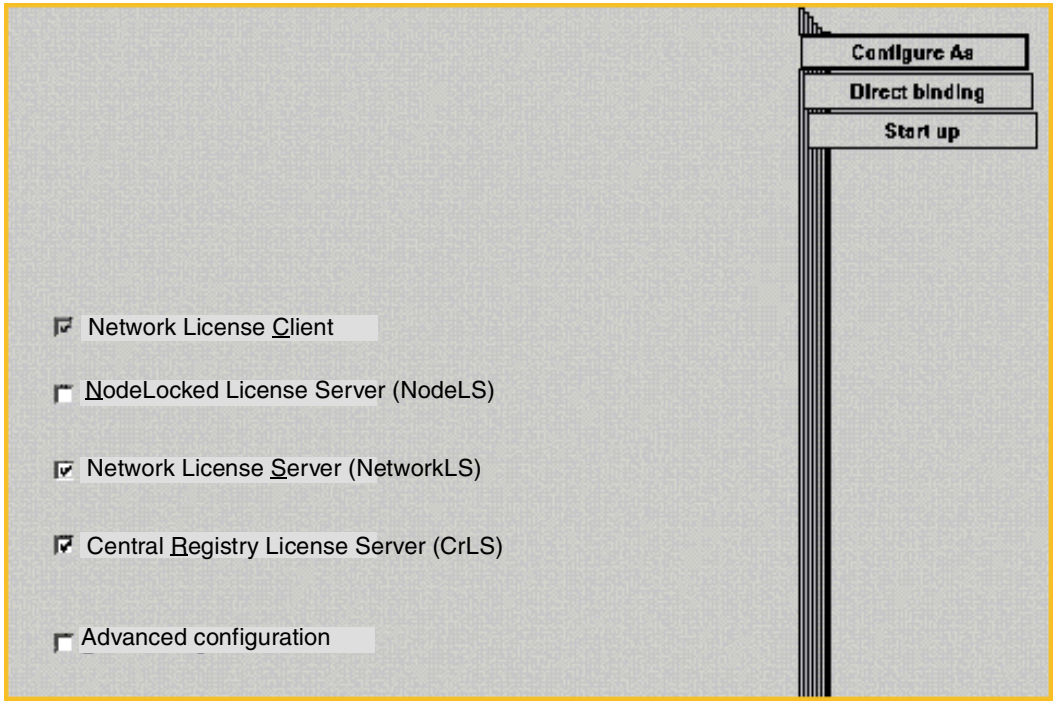


Figure 3-64 Configure As window

Step 2: Direct Binding

Type the short name (IP host name for TCP/IP operation) of a machine that will act as a license server for this client. If the machine you are configuring is also the server, type the machine's own name.

Use the checkboxes under the name field to specify whether the machine will act as a network license server (**Network LS**), a central registry license server (**Central Registry LS**), or both. If the machine is a stand-alone system, select both. Click **Add**.

After you have specified both a network license server and a central registry license server, select the **Start up** tab.

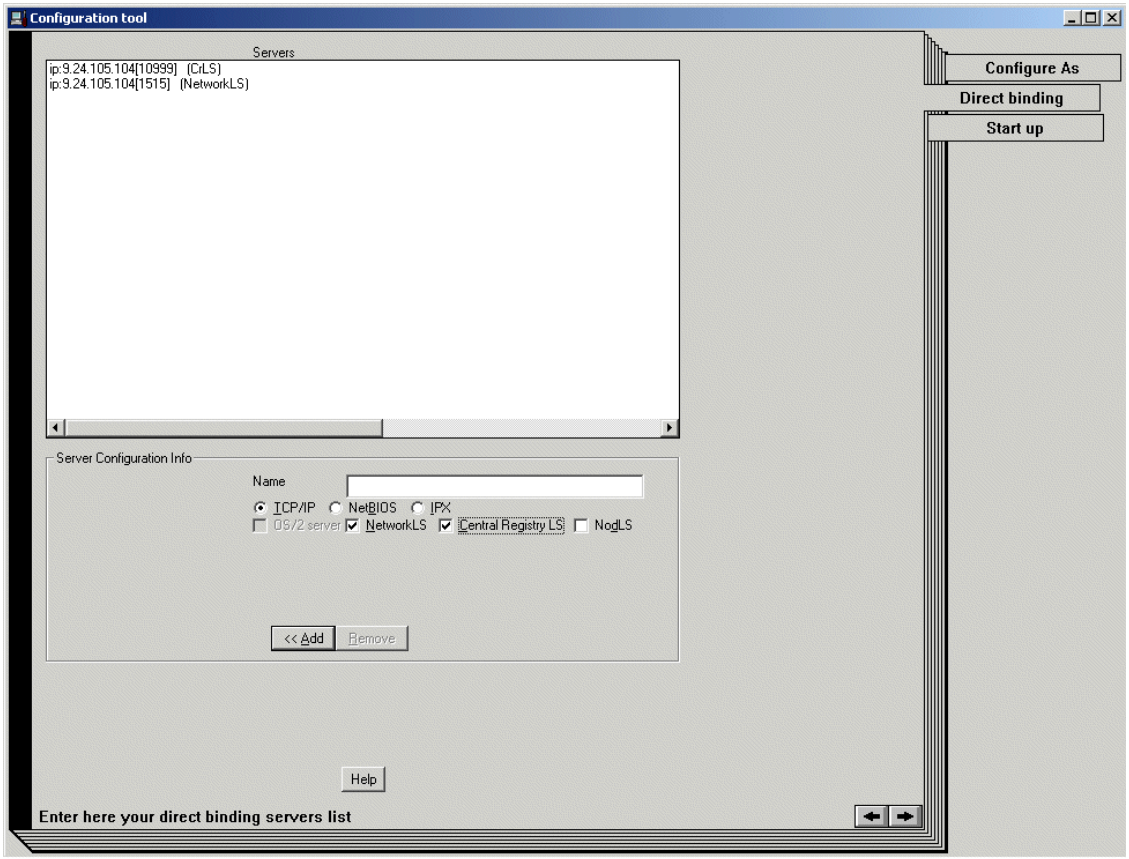


Figure 3-65 Direct Binding

Step 3: Startup

As shown in Figure 3-66, click the **Start up** tab and select **Start Services at system startup** checkbox.

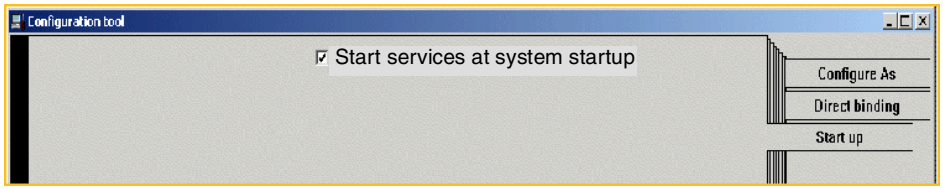


Figure 3-66 Start up

Close the configuration tool notebook.

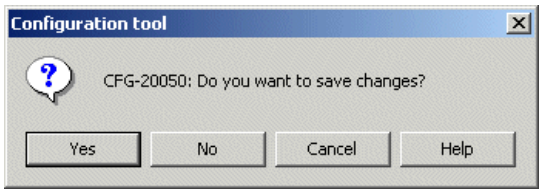


Figure 3-67 Save changes

Click **Yes** to save the changes, shown in Figure 3-67.

Click **OK** to confirm, if prompted.

Step 4: Start license servers

As in Figure 3-68, start your license servers by selecting **Start -> Programs -> License Use Runtime -> Service Manager Tool**. Then click **Service -> Start**.

After the license servers have been started, the icon will turn green and you will be informed they have been started.

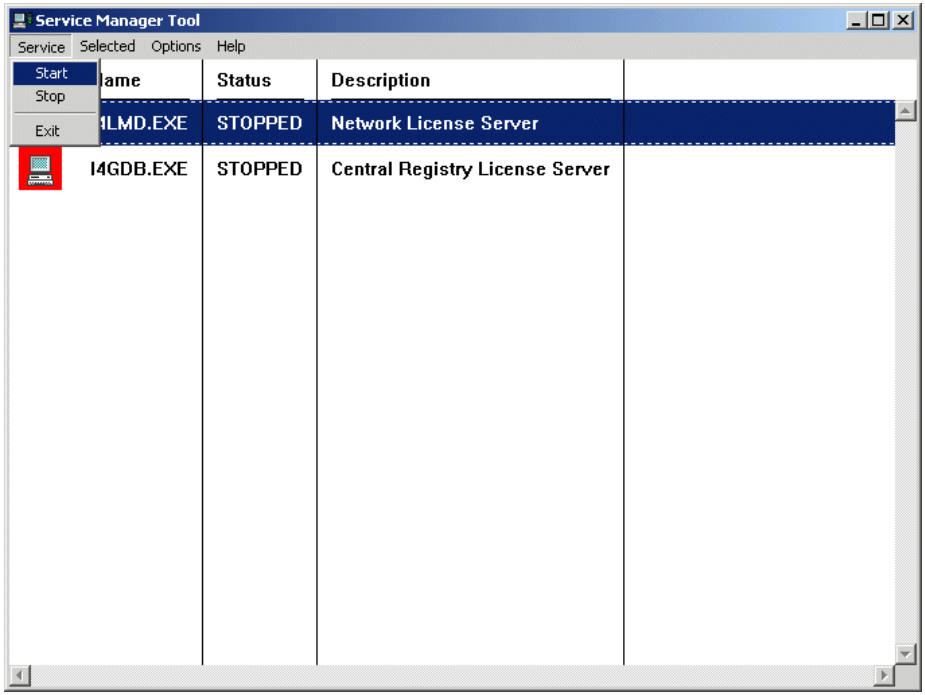


Figure 3-68 Start license servers

Step 5: Enroll licenses

Complete the following steps to enroll licenses:

1. Start the Basic License Tool by selecting **Start -> Programs -> License Use Runtime -> Basic License Tool**.
2. Select **Products -> Enroll -> Single Product.** See Figure 3-69.

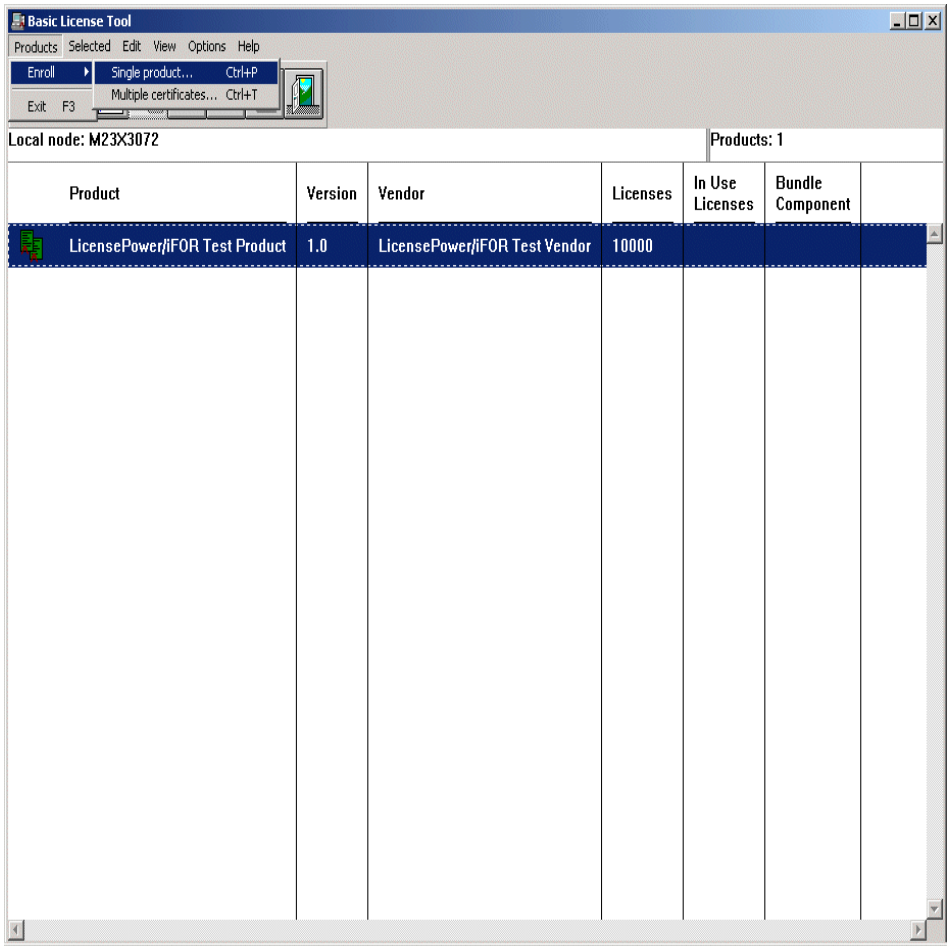


Figure 3-69 Enroll single product

3. See Figure 3-70 on page 77. Click **Import** to display an import window.

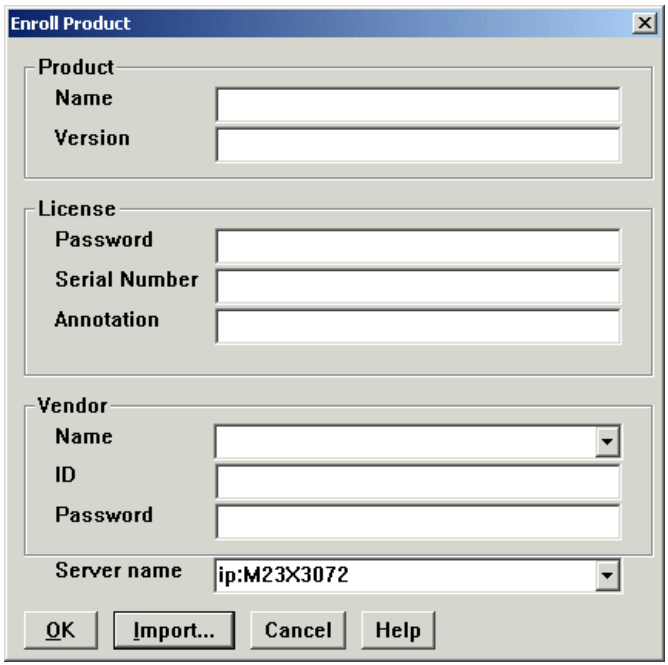


Figure 3-70 Enroll Product window

4. In the Filenames drop-down list, select the **\license** directory, which contains the enrollment certificate rtslines.lic. This file may be found in the folder license.
5. Click **OK**, and the enroll product window is redisplayed. See Figure 3-71 on page 78.

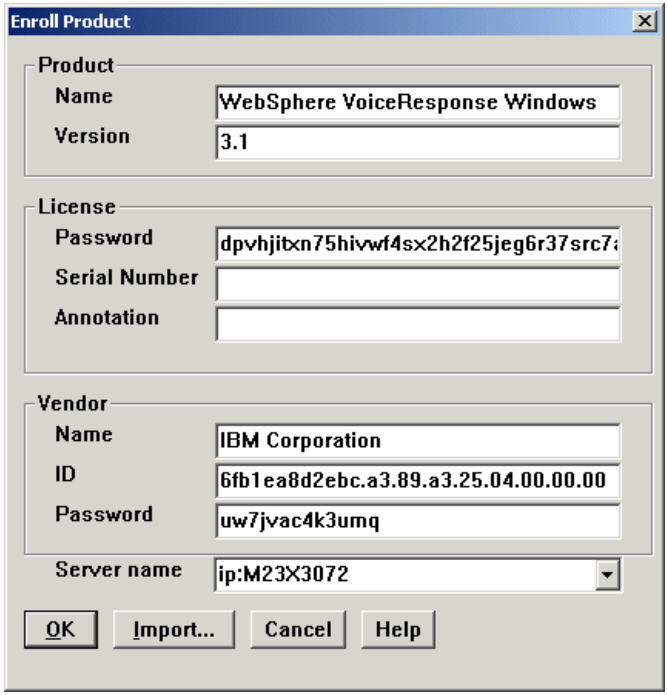


Figure 3-71 Imported information

6. Click **OK**.
7. The enroll license window is now displayed, see Figure 3-72 on page 79. Type your company details in the Administrator information fields.

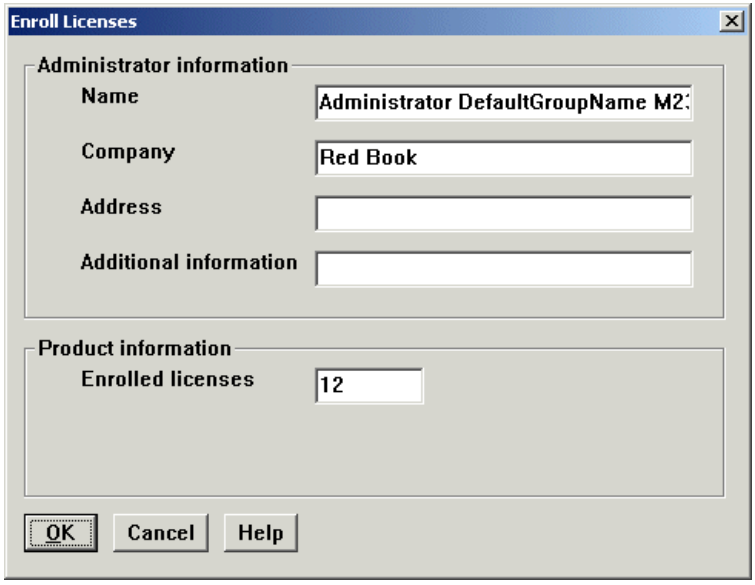


Figure 3-72 Enroll Licenses window

8. In the Enrolled licenses field, type the number of licenses you would like to be available to this WebSphere Voice Response machine. This should be the number of telephony lines that you intend to configure on the machine and must not exceed the number of licenses you have purchased. If installing a network service to support multiple WebSphere Voice Response machines, type the total number of licenses in the pool.
9. Click **OK**.

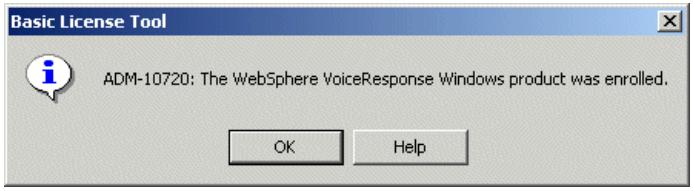


Figure 3-73 Product enrolled

10. In the confirmation window that is now displayed, click **OK**. See Figure 3-73.

Step 6: Distribute your licenses

From the Basic License Tool main window (Figure 3-74 on page 80), perform the following steps:

1. Right-click **WebSphere Voice Response Windows license entry** and select **Distribute licenses**.

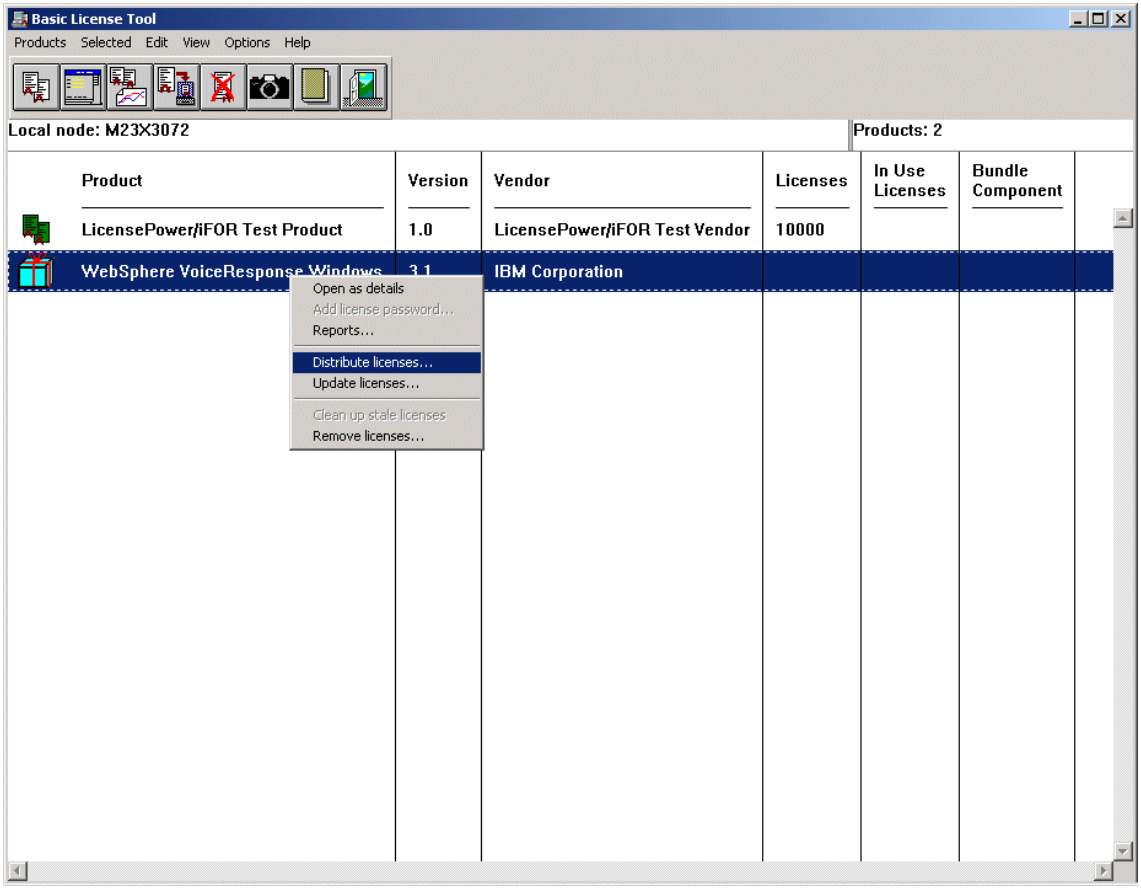


Figure 3-74 Distribute licenses

2. Right-click the server you want to hold the licenses for this client machine and select **Set number of licenses**. See Figure 3-75 on page 81.

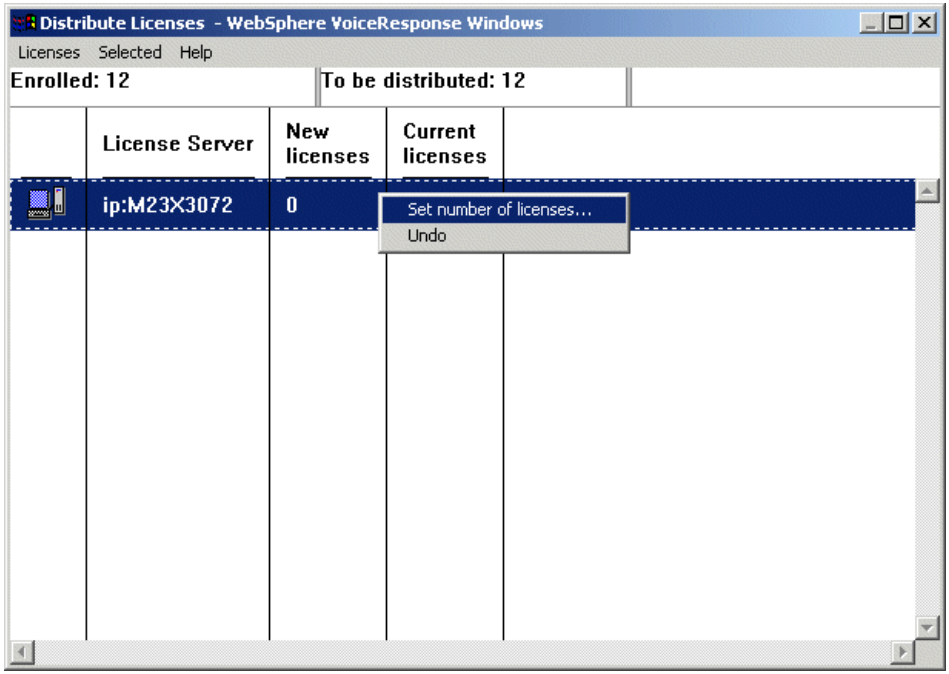


Figure 3-75 Number of licenses

3. Enter the number of licenses that you want to distribute to this server. This will usually be the number of licenses you enrolled on this client machine. Do not exceed the number of licenses you have purchased. See Figure 3-76.

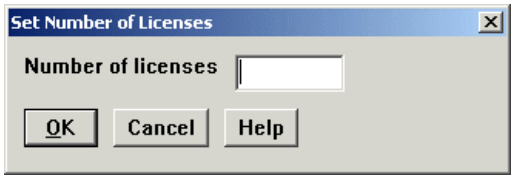


Figure 3-76 Set number of licenses

4. The number of licenses you will be distributing will have to match the number of telephony lines you will be running. See the licensing agreement for more details. Click **OK**.

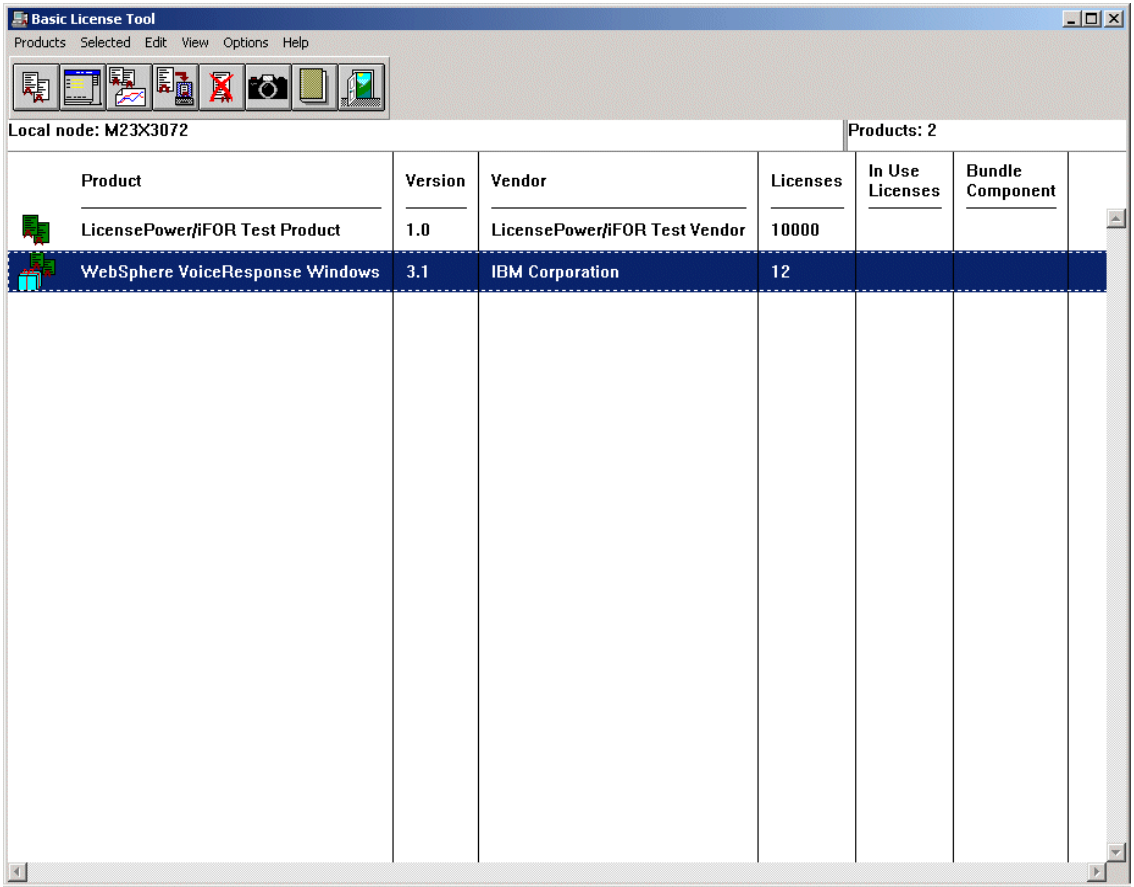


Figure 3-77 Licenses distributed

5. In the Distribute Licenses window, select **Licenses -> Distribute**.
6. After the licenses have been distributed successfully, you should see the number of licenses in the licenses column, as in Figure 3-77.
7. Click **OK** to confirm
8. Close any Basic License Tool windows that are still open.

Step 1: Testing the WebSphere Voice Response installation

Perform the following steps to test the WebSphere Voice Response install:

1. Open a command prompt and go to the directory to C:\dtalk.

2. Type the command **dtjver** and press Enter.

   You will see that the current version of the DirectTalk Beans environment is 3.1.

3. To launch the basic WebSphere Voice Response application that will allow you to test the DTMF functionality, click **Start -> Programs -> IBM WebSphere Voice Response for Windows -> Voice Response Voice System**.

   If Java and VoiceXML support is installed, there is a minimized window that contains the message: `The hostManagerImpl for this machine was successfully created.` See Figure 3-78.

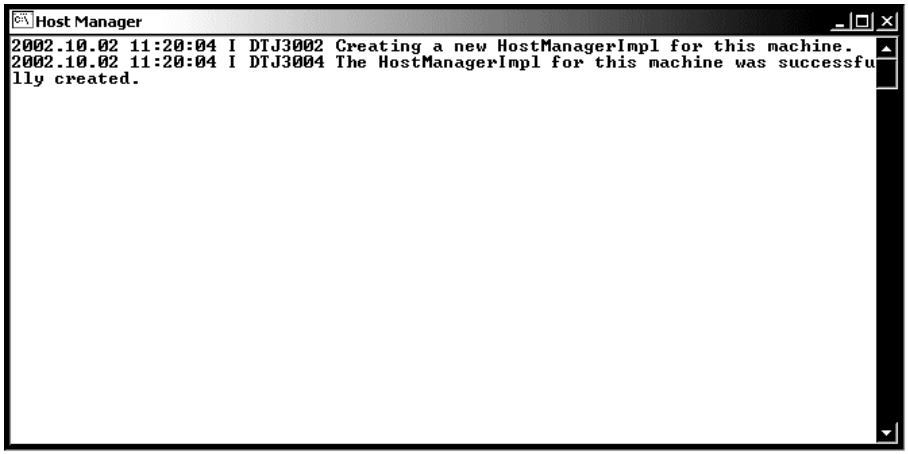


Figure 3-78 HostManagerImpl created

Other commands that will be helpful for a distributed system are **tsmcon -h <servername>** and **ctts -p "this is a test" -T** <TTS engine name on the client running Voice Server>. With these two commands, you can test the connection between telephony servers and voice server clients. The following examples are the actual commands we used. You will want to be mindful that the server name and TTS engine name variables match your configuration.

```
tsmcon -h 9.27.111.244
ctts -p "this is a test" -T ttsen_GB
```

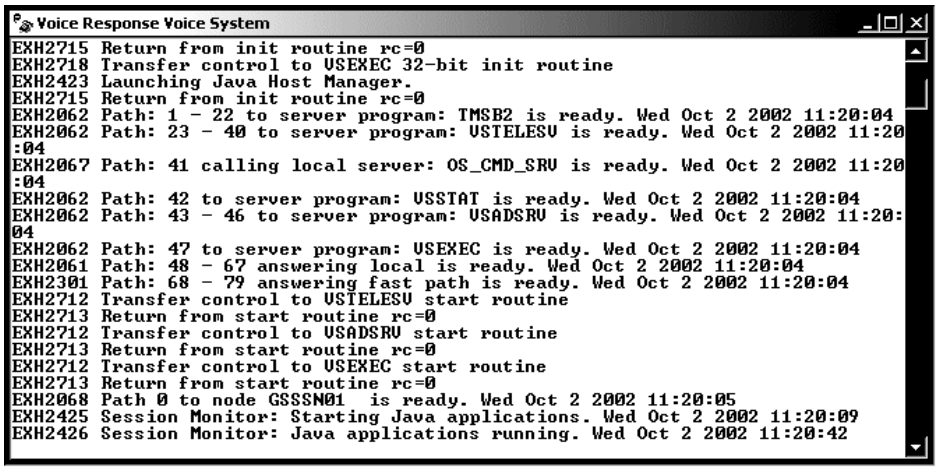


Figure 3-79 Voice Response Voice System

- The Voice Response Voice System window (Figure 3-79), shows in detail the process as they start. Take note of the licensing information detailing the requests made.
- Once the system is up and ready it will broadcast the message: `Path 0 to node GSI Nodename is ready`. You will also see two Java session monitor messages, one detailing the start of the Java applications and the second confirming the start of those applications. The system is now up and ready for you to place a phone call. Place a phone call.

Note: At this point, only the DTMF feature will work in the Voice Response example.

Important: Be sure to bring down the node before installing additional software by pressing CTRL+C in both prompt windows. You will be asked to terminate all batch processes. Type `Y` for the shutdown to finalize.

3.9 Installation of WebSphere language support

Step 1: Setup

You must install language support before you can install the WebSphere Voice Server for Windows. Once downloaded and unzipped, run the setup.exe program. You will see a window similar to Figure 3-80 on page 85. Click **Next**.

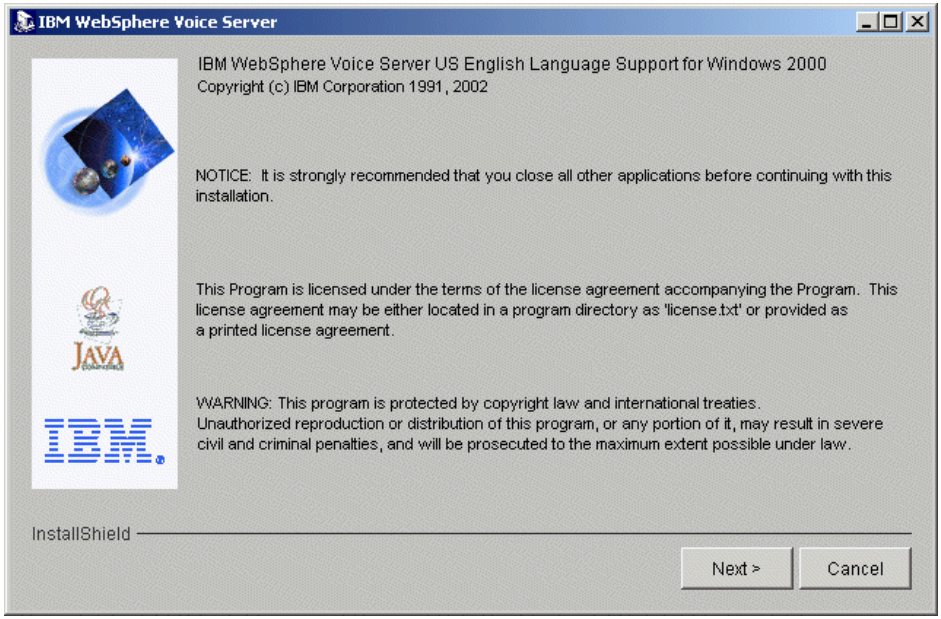


Figure 3-80 Welcome window

Step 2: License agreement

In the Software License Agreement window (Figure 3-81 on page 86), you are required to read the agreement. When done click **I accept the terms in the license agreement**.

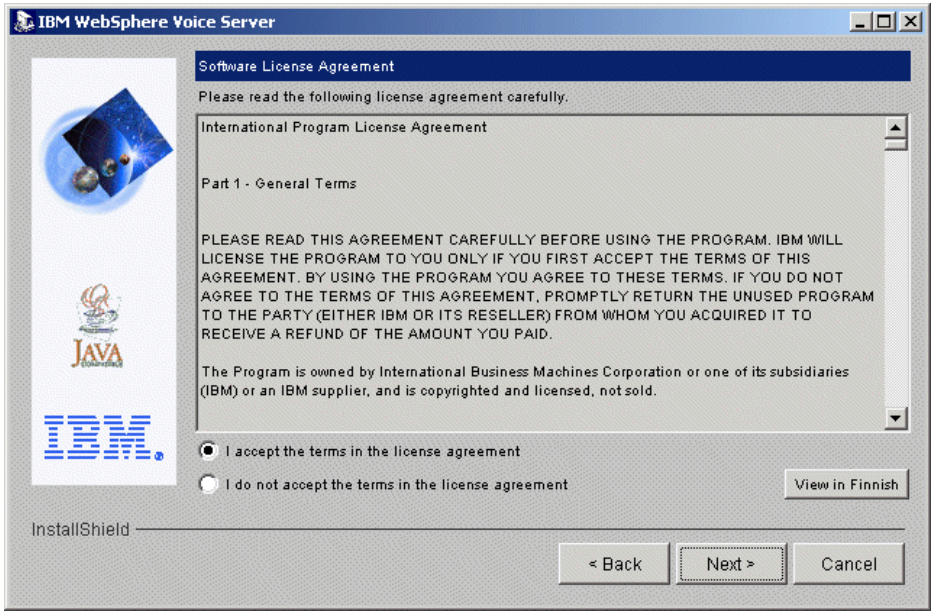


Figure 3-81 License agreement

Step 3: README

At the README window (Figure 3-82 on page 87), click **Next**.

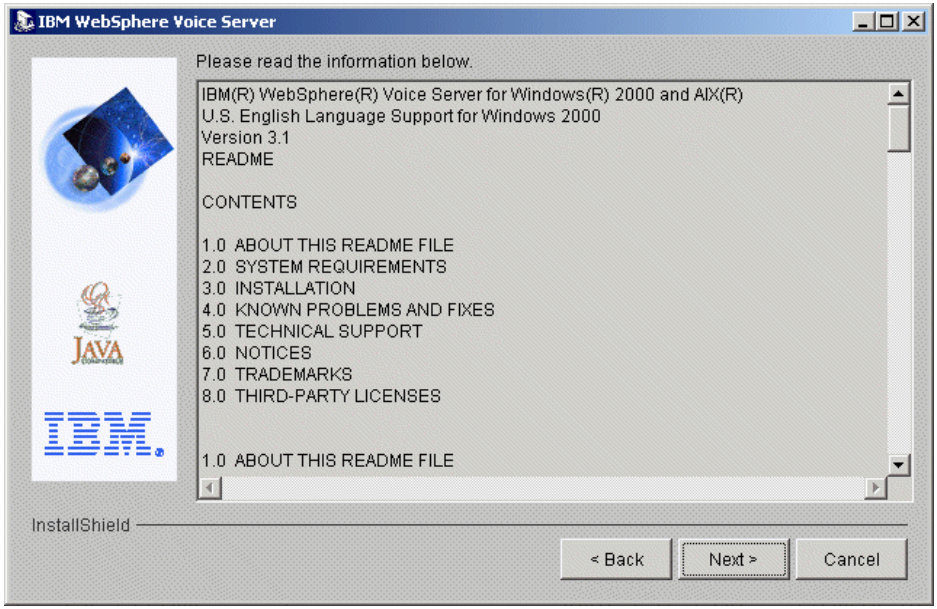


Figure 3-82 README window

Step 4: Confirmation

At the confirmation window (Figure 3-83 on page 88), click **Next**.

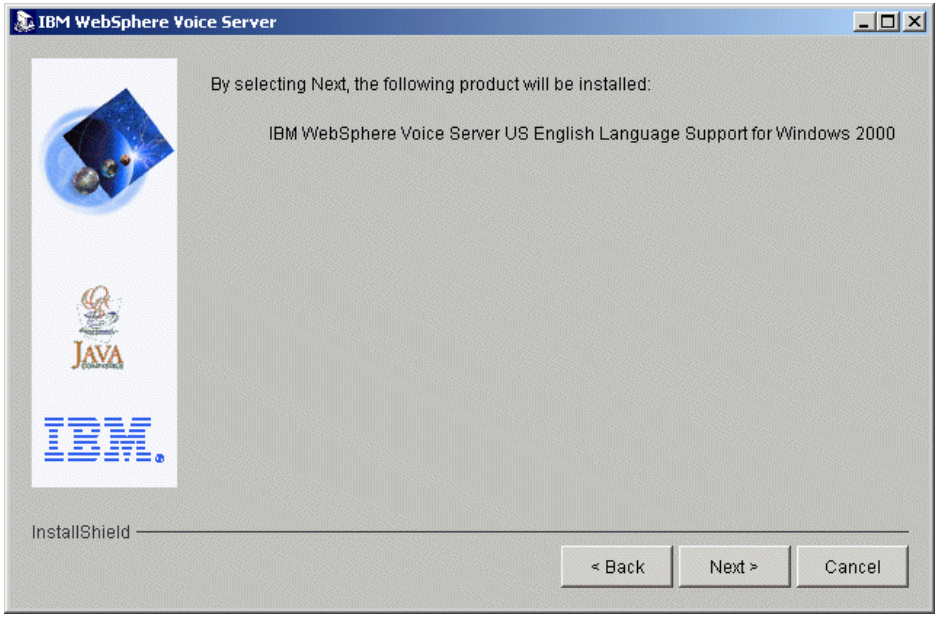


Figure 3-83 Install confirmation window

Step 5: Finish

After a successful install, the Finish window will appear as in Figure 3-84 on page 89. Click **Finish**.

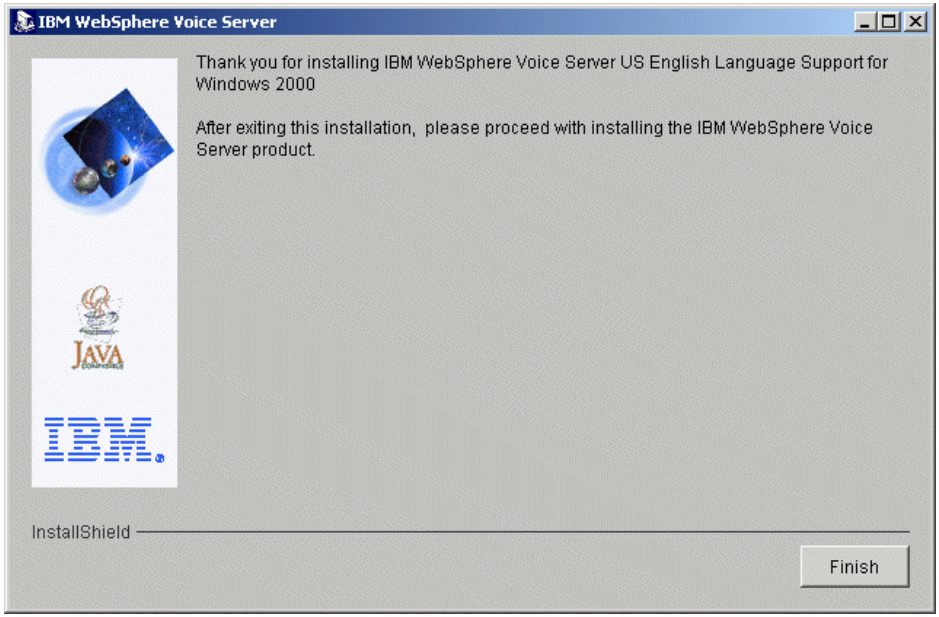


Figure 3-84 Finish window

You are now ready to install WebSphere Voice Server.

3.10 Installation of WebSphere Voice Server

In this section, we will install WebSphere Voice Server.

3.10.1 Stand-alone installation

For the stand-alone installation, perform the following steps.

Step 1: Welcome

Once downloaded and unzipped, run the setup.exe program. You will see a window similar to Figure 3-85 on page 90. Click **Next**.

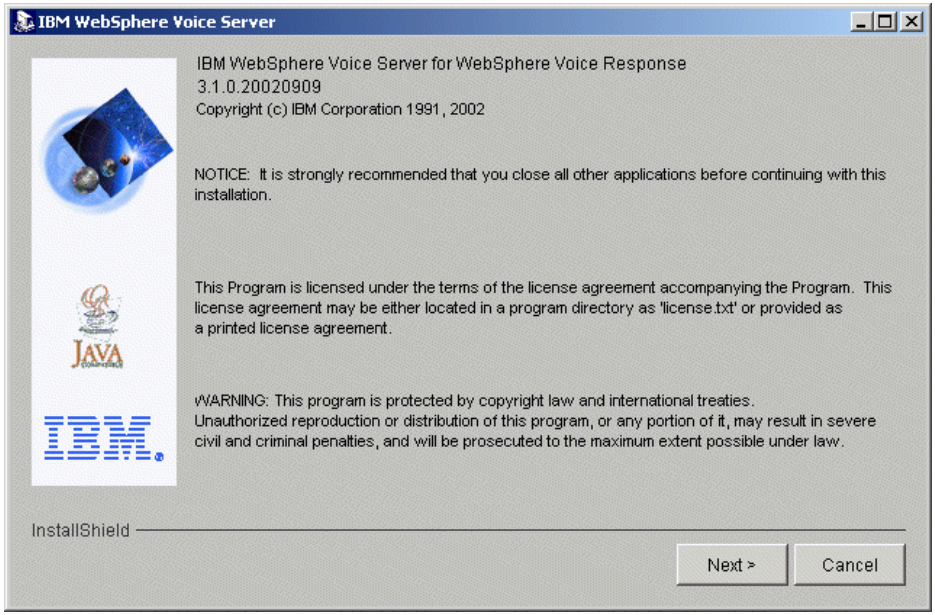


Figure 3-85 Welcome window

Step 2: Client or Server selection

If you are installing a distributed system and this is your client, you would click **Client**. For our installation, we are installing a stand-alone system - server in a distributed system. Click **Server**, as in Figure 3-86 on page 91.

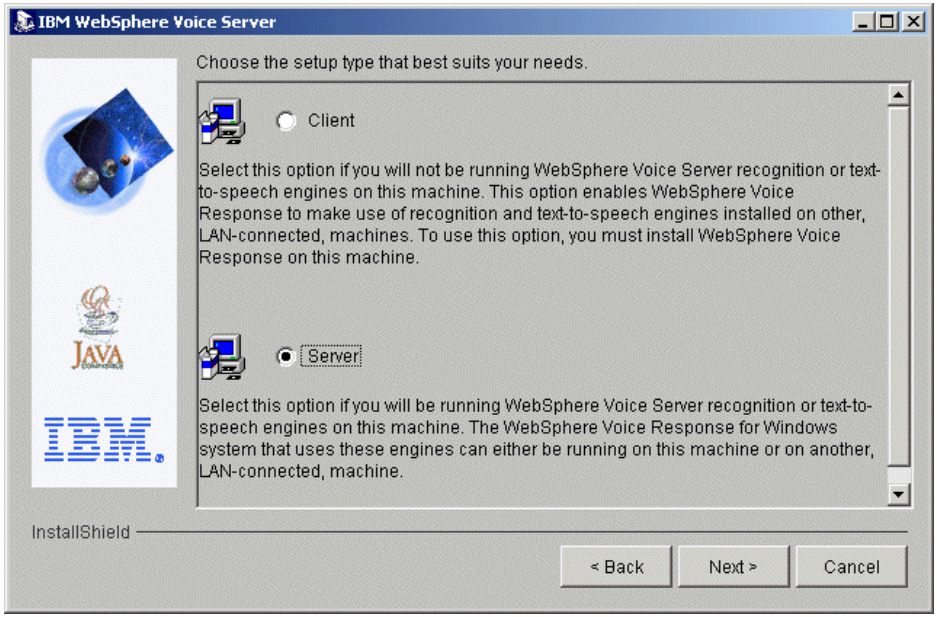


Figure 3-86 Client or Server selection window

Step 3: Pass, Fail, Warning

Once you have chosen Client or Server, you will see the window that details the requirements necessary for a successful install. If all three requirements are met, the window will appear similar to Figure 3-87 on page 92. If not, there will be a fail or warning message next to one of the requirements. If a fail or warning message does appear, WebSphere Voice Server will not install correctly until the error is resolved.

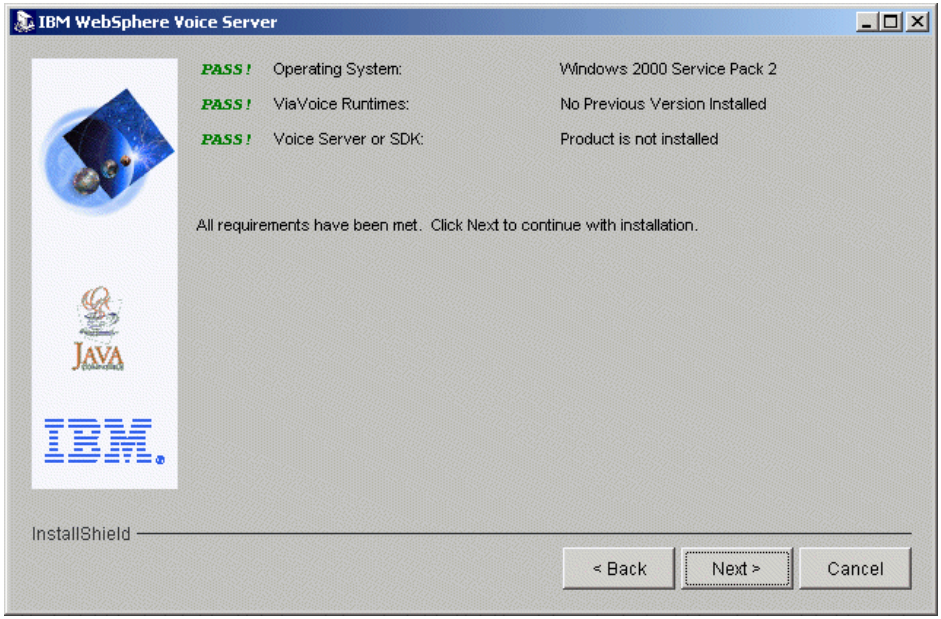


Figure 3-87 Pass

Step 4: License agreement

At the License Agreement window (Figure 3-88 on page 93), you are required to read the software license agreement. When done, select **I accept the terms in the license agreement**. Then click **Next**.

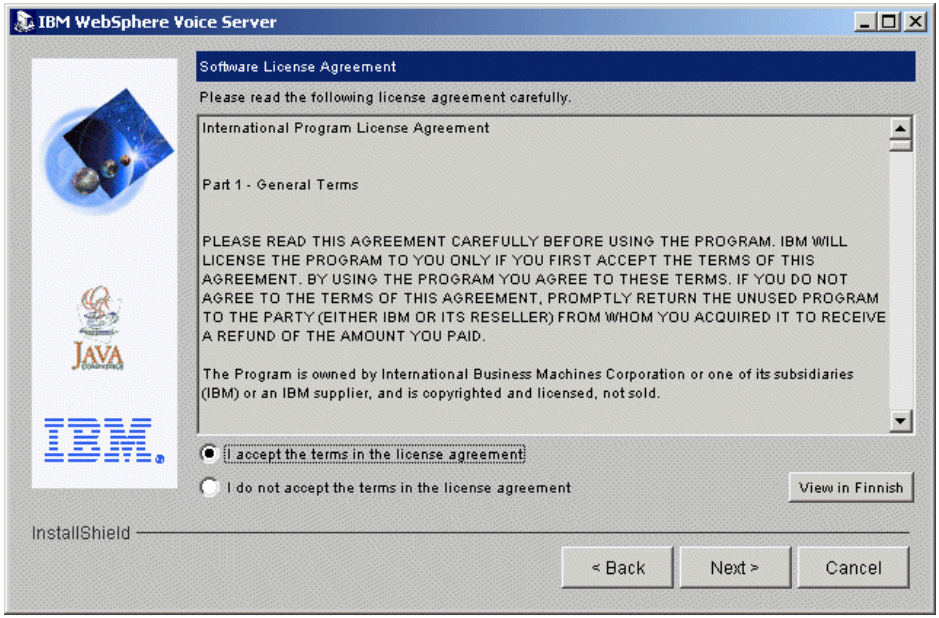


Figure 3-88 License Agreement window

Step 5: Information

In Figure 3-89 on page 94 is the README file for WebSphere Voice Server regarding the install. When you are finished reviewing it, click **Next**.

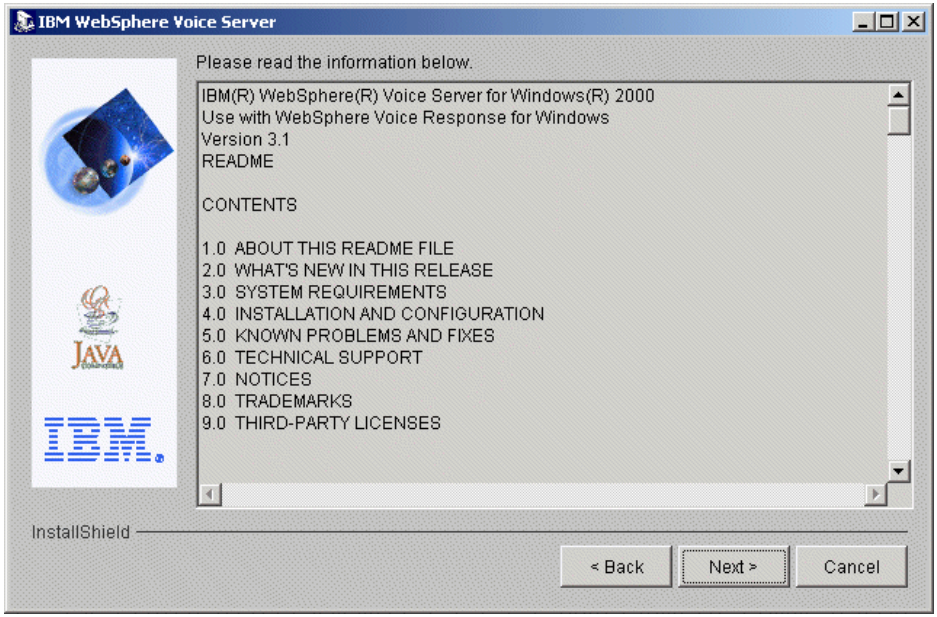


Figure 3-89 Information window

Step 6: Destination

In Figure 3-90 on page 95, you are asked to specify where you would like the WebSphere Voice Server installed. Unless you have a good reason to change the default, it is recommended that you choose the default location.

If this directory does not exist, you will be prompted to create the directory. Click **Yes**.

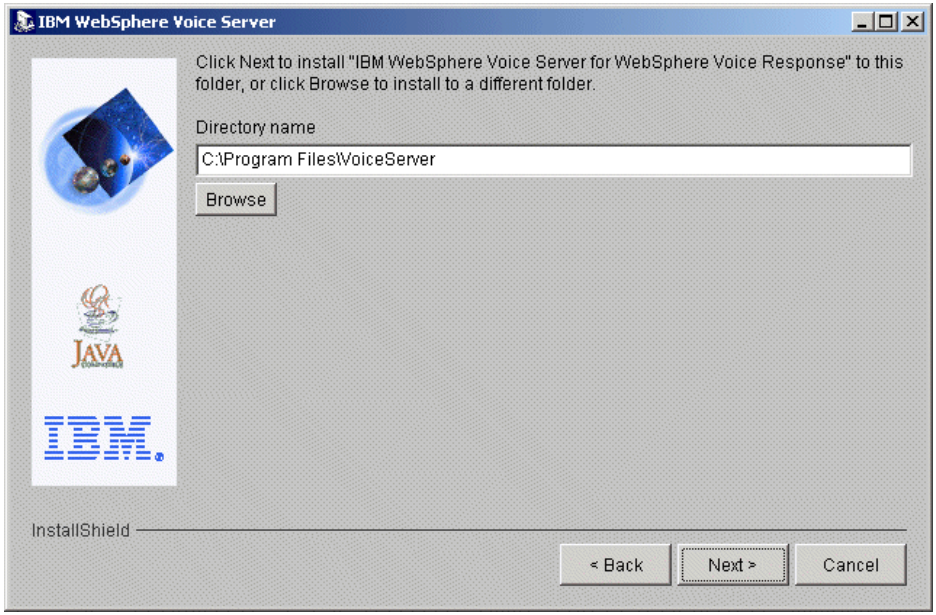


Figure 3-90 Destination window

Step 7: Language selection

Depending on which language support you have installed, the appropriate check boxes will be highlighted. Verify the correct language is going to be installed as in Figure 3-91 on page 96, In our example, we have installed US English as the primary language. Click **Next**.

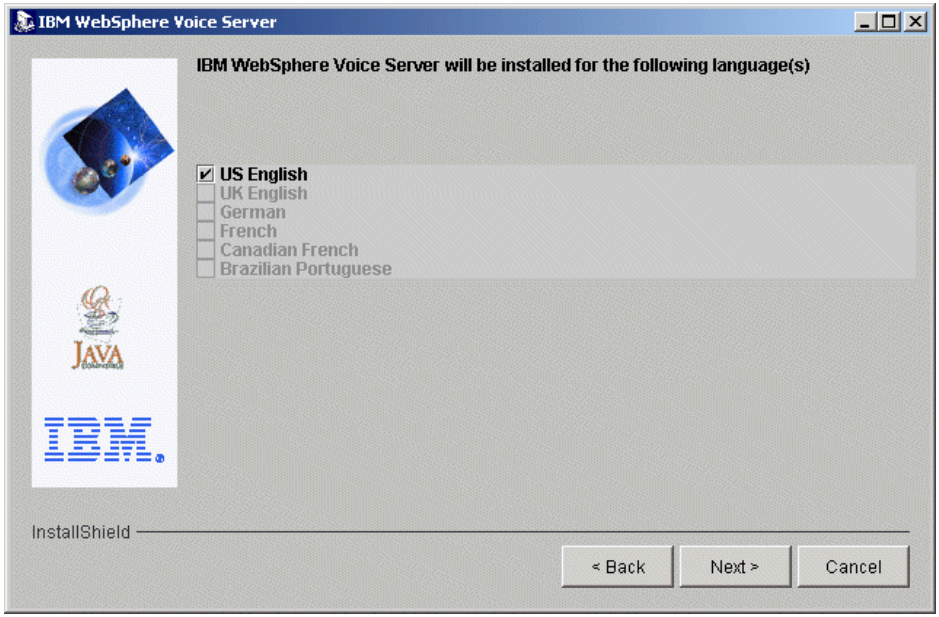


Figure 3-91 Language selection

Step 8: Finish

After the installation, the Finish window (Figure 3-92 on page 97) will appear. Click **Finish**.

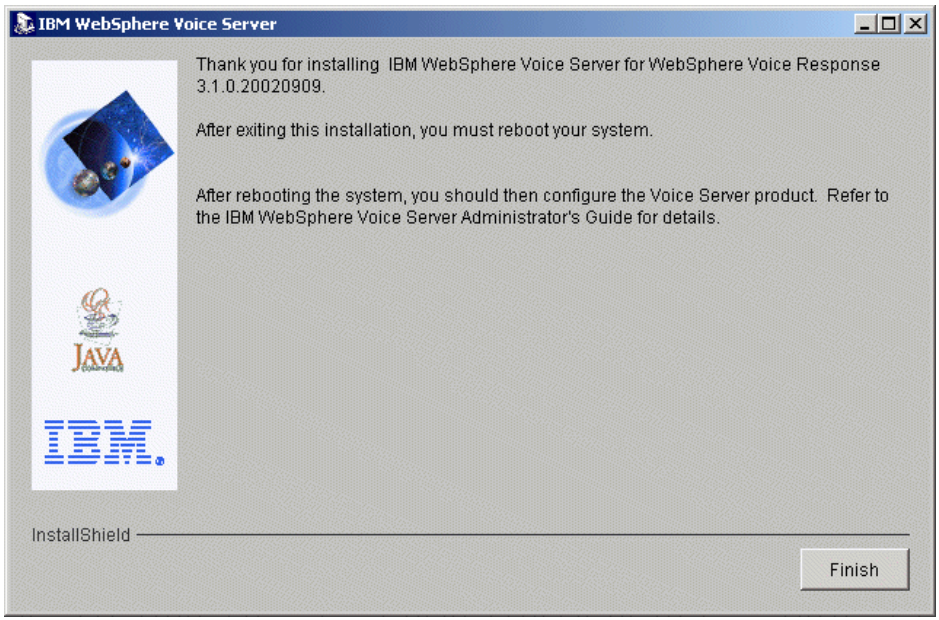


Figure 3-92 Finish window

Important: You must reboot the system before attempting to configure the WebSphere Voice Response or Voice Server.

3.11 Configuration of WebSphere Voice Server

After the installation of WebSphere Voice Server, you must configure the WebSphere Voice Response and Voice Server correctly in order to run the weather.vmxl sample application. Perform the following instructions:

Step 1: VVTConfig.tcl

1. Click **Start -> Programs -> IBM WebSphere Voice Response for Windows -> Voice Response Voice System**.
2. Click **Start -> Programs -> IBM WebSphere Voice Response for Windows -> Voice Response Command Prompt**.
3. At the c:\dtalk prompt, type `vvtconfig.tcl`. You will see a window similar to Figure 3-93 on page 98. Fill in the answers in accordance with your system configuration.

Note: If you are installing a distributed system, and you run the **vvtconfig.tcl** command on the telephony machine, you will be asked on which host(s) are the router processes for this client running. You can use the IP address of the client that contains the WebSphere Voice Server. If there are multiple voice server clients, separate them with a space.

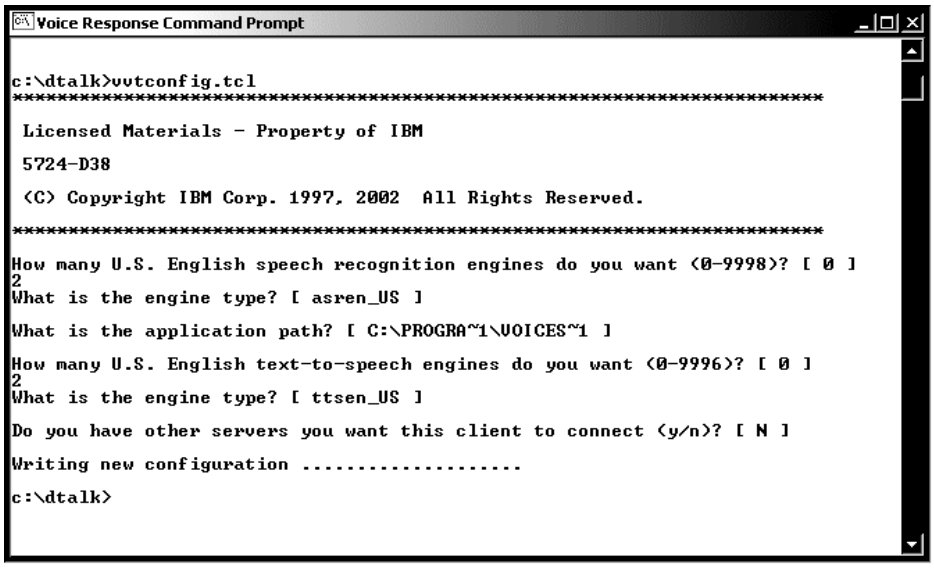


Figure 3-93 vvtconfig.tcl

4. After the new configuration has been written, start the process manager by typing `pm`. (There will be no process manager installed on the client in a distributed system).
5. Then type `pm config`.

Note: Every time you make changes via the vvtconfig.tcl, you will need to run **pm config** in order for the changes to take effect. You can read more about the process manager in *WebSphere Voice Server for Windows Use Administrator's Guide.*

Note: In order to avoid having to start the process manager each time you restart the Windows environment, you can use the command **pm install**. This will install the process manager as a service and will be started when Windows is started.

Step 2: WebSphere Voice Response Configuration notebook

As shown in Figure 3-94, perform the following instructions:

1. Click **Start -> Programs -> WebSphere Voice Response for Windows**.
2. Select **Voice Response Configuration** to display the Voice Response Configuration properties notebook.
3. Click the **WVS** tab to display settings associated with the WebSphere Voice Server including:
   - Enable or disable Voice Server support
   - Log recognition results
   - Display console windows on startup
   - Record incoming voice data
   - Number of restarts
4. Under the WebSphere Voice Server Support**,** select **Enabled**. The remaining check boxes are diagnostic options that should remain unchecked.
5. To save your changes, select **Configuration -> Save and Close**.

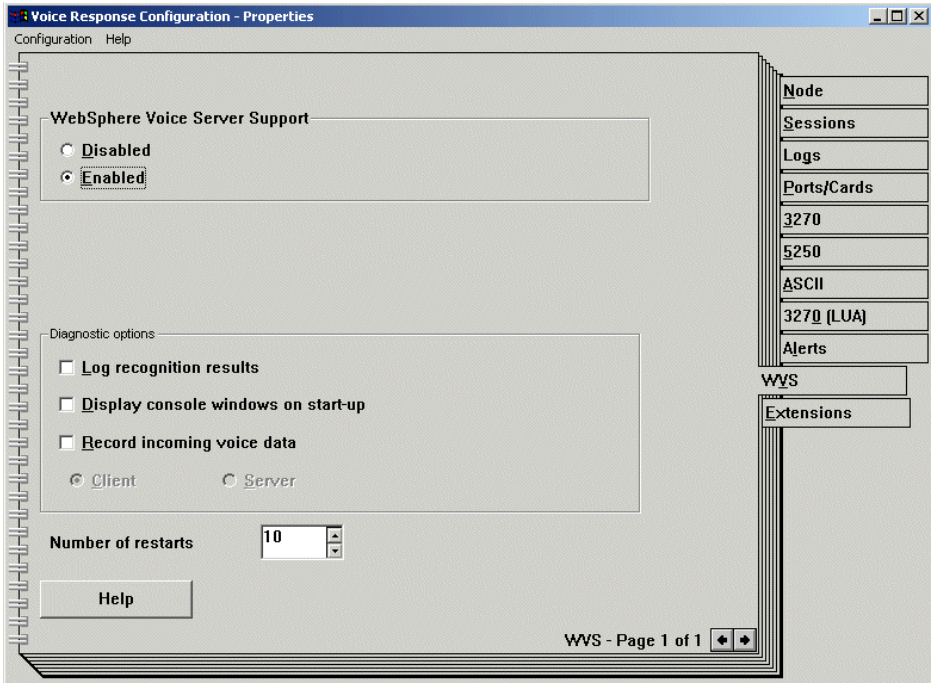


Figure 3-94 Enable WebSphere Voice Server

Step 2: Configure default.cff

To enable the application to be independent of the technology being used, speech recognition and text-to-speech in the WebSphere Voice Response Beans Environment can be used without hard-coding details of the speech technology inside the application. Instead, you must specify the details in the configuration file (default.cff).

If you already have Java and VoiceXML applications running, read the following instructions to work out how to modify your own default.cff file to get the Weather.VXML application running. In particular, copy the RecoService entries and TTSService entries for the languages you have installed to your own default.cff. Also, in the NodeName entry, copy the RecoService, TTSService, RecoDefinition, and TTSDefinition keywords for the languages you have installed to your default.cff.

Note: Before making alterations to the default.cff file, it is highly recommended that you create a backup copy.

The default.cff is located in the c:\dtalk folder. You will also find a default.sample.cff, which will be used for an example. Perform the following steps:

1. Open both the default.cff file and the default.sample.cff files in Wordpad.

Note: These files should be edited in Wordpad, as Notepad does occasionally create errors.

2. In the default.sample.cff file, find the TTSService entries for the text-to-speech languages you have installed and the RecoService entries for speech recognition languages.

```
TTSService=WVS_TTSen_US
 PluginClass=com.ibm.telephony.directtalk.TTSVVNT
 TTSType=TTSen_US
 InitSessionString=EngineName=ttsen_US
;

RecoService=WVS_Recoen_US
 PlugInClass=com.ibm.telephony.directtalk.RecoVVNT
 InitSessionString=EngineName=asren_US, Profiles=vvsample,
BargeInMode=BARGEIN_RECO_COMPLETE
 InitTechnologyString=LocalBaseDir=c:\\grammars, ServerBaseDir=c:\\grammars
 RecoType=Recoen_US
;
```

3. In the default.cff file, paste the TTSService entries for the text-to-speech and the RecoService entries for speech recognition languages you have installed, as shown in Figure 3-95.
4. In the InitSessionString line, change the Profiles=vvsample line as follows. If you must specify more than one context profile, separate their names using commas.

```
InitSessionString=EngineName=asren_US, Profiles=vvsample,
BargeInMode=BARGEIN_RECO_COMPLETE
```

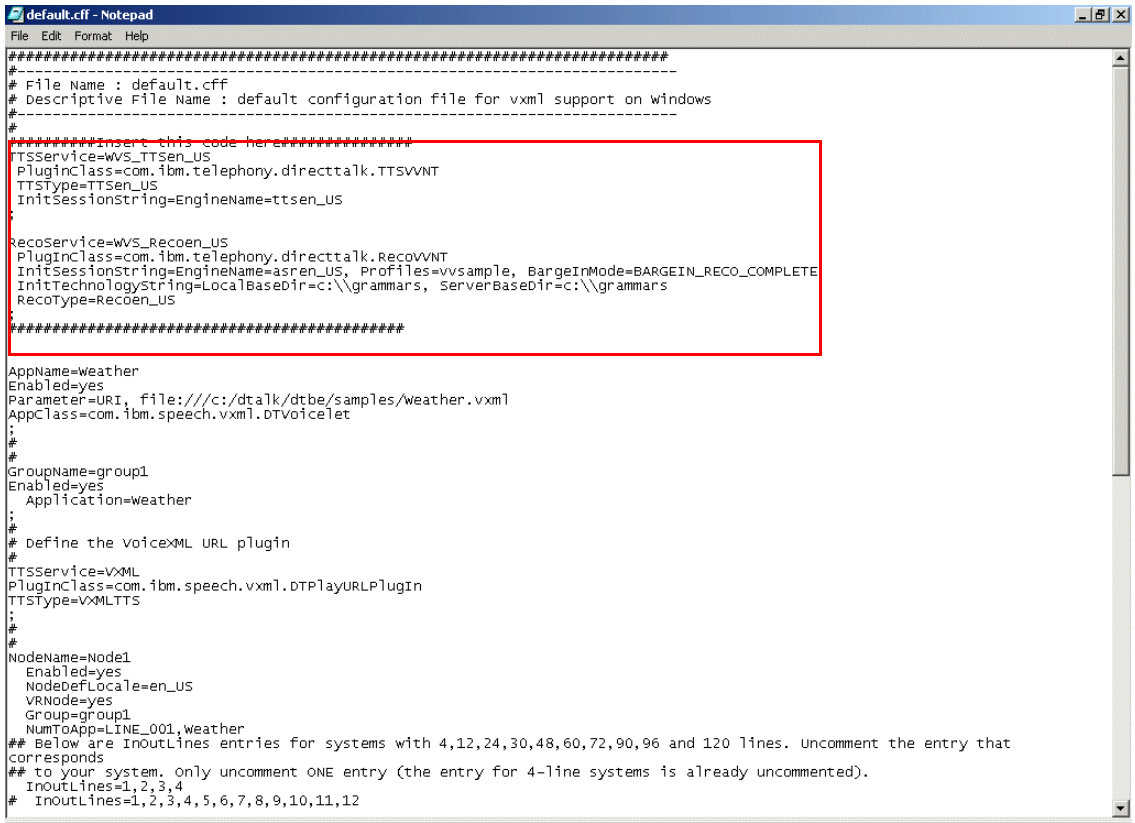


Figure 3-95 default.cff position 1

5. After you have pasted the code TTSService and RecoService entries into the default.cff, in the default.sample.cff file, you will need to find the NodeName=Node1 entry near the end of the file and copy the following. See Figure 3-96 on page 102.

```
TTSService = WVS_TTSen_US
TTSDefinition = en_US, TTSSen_US
RecoService = WVS_Recoen_US
```

```
RecoDefinition = en_US, Recoen_US
```

Note: If you have installed more than one language, you can use * to denote multiple engines. Your line of code would look like this:

```
RecoDefinition=*, Recoen_US
```

6. Copy those lines of code into your default.cff file, in the position that is denoted in Figure 3-96. Be sure to paste those definitions before the semicolon that ends the NodeName=Node1 definition.

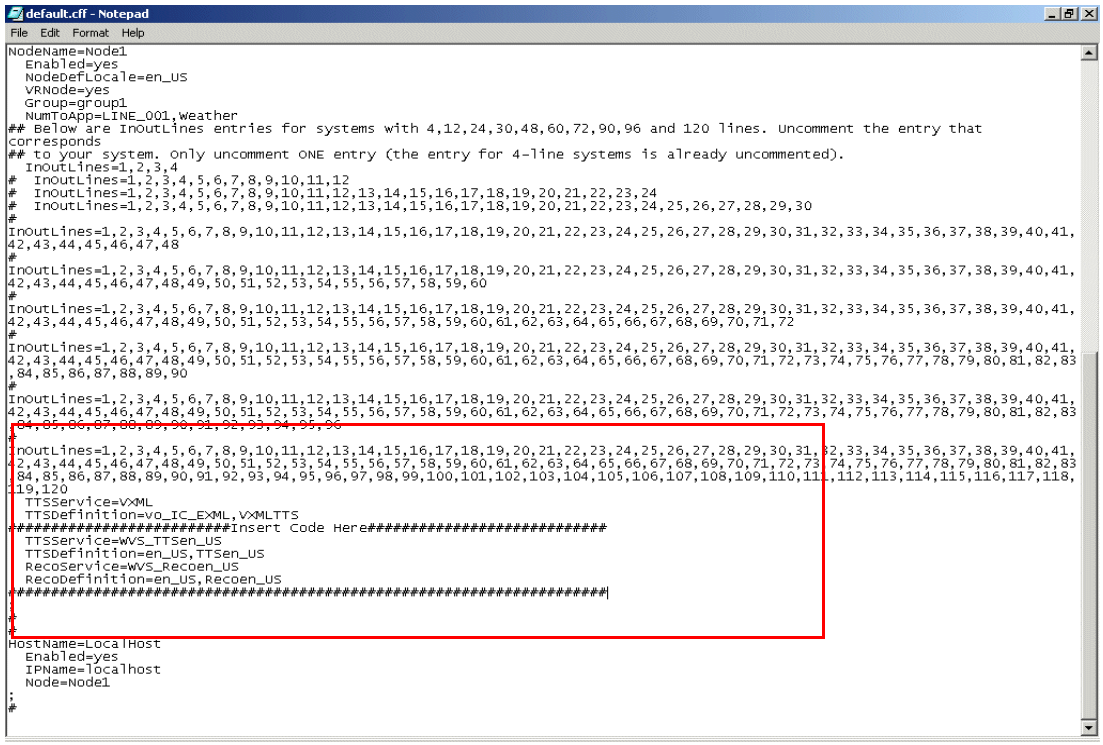


Figure 3-96 default.cff position 2

7. If the server is on a different system from the client, add a HostName entry for the server system, for example:

```
HostName=Sabertooth
Enabled=Yes
IPName =Sabertooth.hursley.ibm.com
Node=Node1
;
```

8. If you are not using c:\grammars\ as your shared file system directory, in your default.cff file edit the entry c:\grammars\ in the InitTechnologyString lines to change the paths that the client and the server will use to access the shared directory you have set up. This shared folder will need to be accessed by all clients that have WebSphere Voice Server installed. Usually this folder is located on the telephony server. For example:

   ```
   InitTechnologyString=LocalBaseDir=c:\\grammars, ServerBaseDir=f:\\grammars
   ```

Note: Each single backslash character (\) in the path must be specified as a double backslash (\\) in the default.cff file.

Important: If you are installing a stand-alone system, you will need to set `ServerBase=c:\\grammars`. In a distributed system the ServerBase path should the mapped drive where the clientDrive:\\grammars can be found.

9. Save the default.cff file (if you get a warning about saving the text file, click **Yes**).
10. If the node is not running, start WebSphere Voice Response for Windows by clicking **Start -> Programs -> WebSphere Voice Response for Windows -> WebSphere Voice System**. Otherwise, if the node is already running, from the \dtalk directory in a Windows command prompt window, do the following:
    a. Run **`dtjstop`** to stop the node
    b. Run **`dtjconf`** to make the changes take effect
    c. Run **`dtjstart`** to restart the node

If you need to make changes to the default.cff file at any time, repeat the above steps after you save the default.cff file.

3.12 Testing system

Before running the VMLSample application, Weather.VXML, it is recommended that you run the vvsample application from a command prompt, which will allow you to determine if your installation is correct or if there is a problem with the configuration.

You can run the vvsample application with any installed language. Before running the application:

1. Ensure that the engines are running. Make a note of the names of each engine that you want to test.

2. Start WebSphere Voice Response. Refer to the *WebSphere Voice Response for Windows: Administrator's Guide* for further information.
3. Ensure that no other applications are running.

To be certain no applications are running, stop them by using the **dtjstop** command from a Voice Response command prompt. If any T-REXX applications are running, stop them using the WebSphere Voice Response Node Manager.

To run vvsample:

1. From the WebSphere Voice Response for Windows folder, open a Voice Response command prompt window.
2. At the command prompt, type `vvsample` to start the application. A menu is displayed within the command prompt window.

VR — Verify recognition

Type `VR` to test speech recognition using a simple, three-word grammar. The application guides you through the test procedure:

a. When prompted, phone into the system on line 001.
b. At the Voice Response command prompt, type the name of the speech recognition engine you want to test.
c. Type three words or phrases to include in your test grammar. The words you specify should be distinct and consist of two or more syllables in length.
d. After the tone, speak one of the words you typed in the previous step. The result of the test is displayed on the screen.

VT — Verify text-to-speech

Type `VT` to test text-to-speech using a simple, text string you specify. The application guides you through the test procedure:

a. When prompted, phone into the system on line 001.
b. At the Voice Response command prompt, type the name of the speech recognition engine you want to test.
c. Type the phrase you want to be spoken back to you using text-to-speech.
d. If the test is successful, the specified phrase is played back over the telephone.

Weather.VXML

To check that the environment has been configured correctly for running Java and VoiceXML voice applications, do the following:

1. Ensure that the engines are running.
2. Start WebSphere Voice Response. Refer to the *WebSphere Voice Response for Windows: Administrator's Guide* for further information. (If WebSphere Voice Response is already running, from the \dtalk directory in a Windows Command Prompt window, run the **`dtjstart`** command to start the node.)
3. Dial into your WebSphere Voice Response system. The Weather.VXML application should answer as the default application. When the application is first run, there may be a delay before it answers the call.

The Weather.VXML application first asks you which language you wish to use. To select a language, press one of the following numbers:

1. US English
2. UK English
3. French
4. German
5. Canadian French
6. Brazilian Portuguese

After selecting a language, you are asked to select the WebSphere Voice Server functionality that you wish to test. To select the functionality, press one of the following numbers:

1. Basic functionality (pre-recorded prompts), without text-to-speech or speech recognition
2. Text-to-speech
3. Speech recognition
4. Both text-to-speech and speech recognition

3.13 Echo cancellation

Applications for WebSphere Voice Server can be configured to allow the barge-in (or cut-through) function for speech recognition. Barge-in permits the end user's speech to interrupt system prompts as the machine plays them. To perform this function, additional voice processing is required at the hardware level, using telephony cards that provide echo cancellation. Supported echo cancellation cards for WebSphere Voice Server are:

- D/80SC
- D/160SC
- D/240SC

- D/320SC
- D/80SC-PCI
- D/320SC-PCI
- D/320JCT

3.13.1 Dialogic card installation

The D/320JCTU echo cancellation card was tested in several Dialogic configurations. The first was in an analog environment, where multiple analog cards were used in conjunction with the D/320JCTU. The second environment used T1 telephony connectivity, using several T1 cards. In both cases, all the cards were connected using the SCBus cable.

In the analog environment, we used the following card combination:

- D/41JCT-LS (a 4-port analog board)
- D/120JCT-LS (a 12-port analog board)
- D/320JCTU

In the digital environment we used the following card combination:

- D/240JCT-T1 (a 24-port digital board)
- D/480JCT-2T1 (a dual-span 48-port digital board)
- D/320JCTU

The D/320JCTU provides up to 32 channels with the echo cancellation function. If there are more than 32 channels requiring this function, additional D/320JCTU cards must be installed.

The following is a step-by-step procedure to install the D/320JCTU card into an analog environment using WebSphere Voice Server for WebSphere Voice Response on Windows 3.1.

Step 1: Installing hardware

Our server had a WebSphere Voice Server for WebSphere Voice Response on Windows 3.1 system running, using a D/41JCT-LS card. Power the machine off and unplug the power supply. We installed both the D/120JCT-LS and D/320JCTU cards into the server, preferably using the lowest numbered slots. Once they have been secured, attach the SCBus ribbon cable so they are all linked, as shown in Figure 3-97 on page 107.

Note: The SCBus cable will fit only one way, since one side has longer plastic lugs. Make sure the cable is installed correctly.

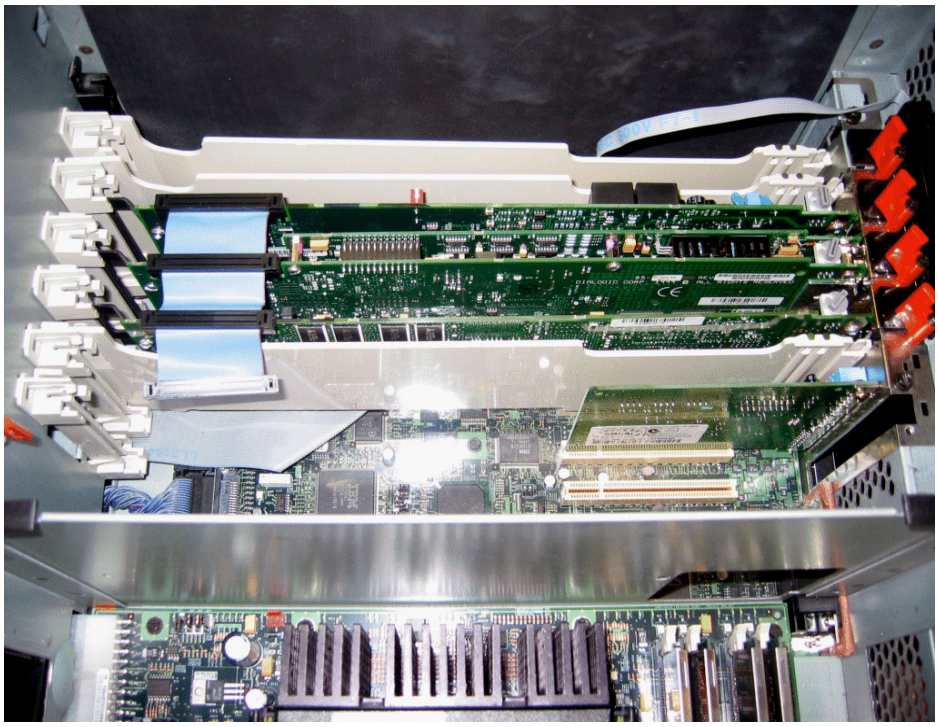


Figure 3-97 Dialogic cards installed with SCBus ribbon

Each Dialogic card is required to have a unique ID number. This is manually set by a rotation switch located on the top of the each card, closest to rear of the computer when inserted. An example of the assigned ID numbers are shown in Example 3-4.

Table 3-4 Unique ID number assignment

Card	PCI slot number	Card ID
D/41JCT-LS	2	0
D/120JCT-LS	3	1
D/320JCTU	4	2

Note: The echo cancellation card must be set with the highest board ID of all the cards in the system.

Step 2: Hardware detection

Perform the following instructions to initiate the hardware detection wizard:

1. Power the machine up and log in as the administrator. The Hardware wizard should appear indicating new hardware has been found, as shown in Figure 3-98. Click **Next**.

Figure 3-98 Hardware window

2. The Hardware wizard has identified the new hardware as a PCI card, as shown in Figure 3-99 on page 109. Click **Next**.

Figure 3-99 PCI device detected

3. The Dialogic software provides a set of drivers. We selected the option **Specify a location**, as shown in Figure 3-100. Click **Next**.

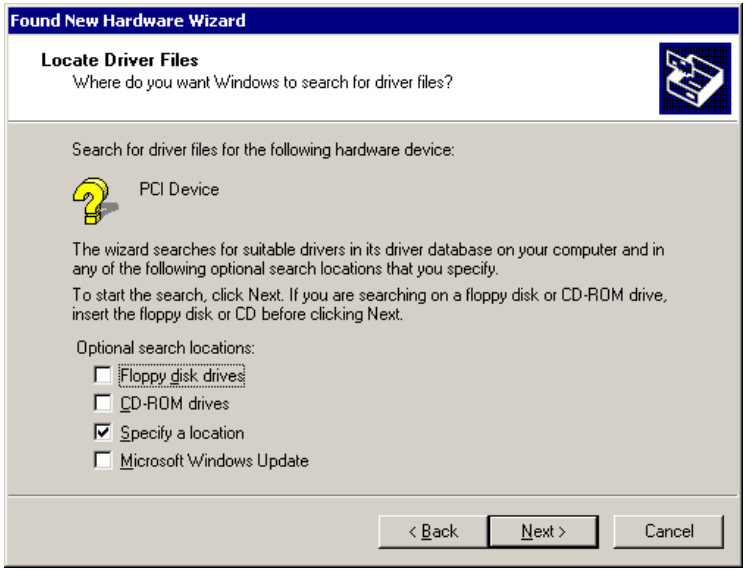


Figure 3-100 Specify location window

4. We need to manually point to the driver, so click **Browse**, as shown in Figure 3-101.

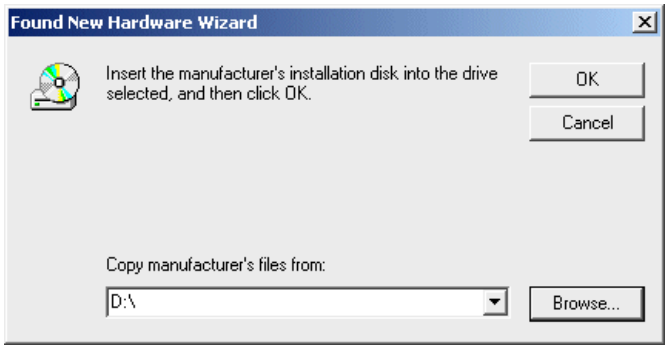


Figure 3-101 Browse window

5. The drivers are located in the DRVR subdirectory of the installed Dialogic software. In our case, this is the default folder. Highlight the folder and click to open, as in Figure 3-102.

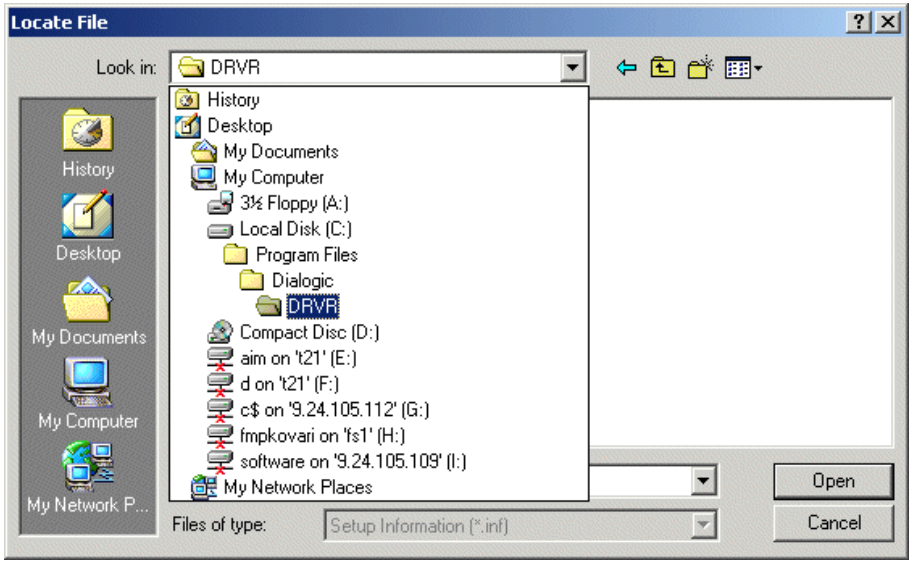


Figure 3-102 Dialogic driver location window

6. The Hardware wizard will indicate the driver it found, as shown in Figure 3-103 on page 111. Click **Next**.

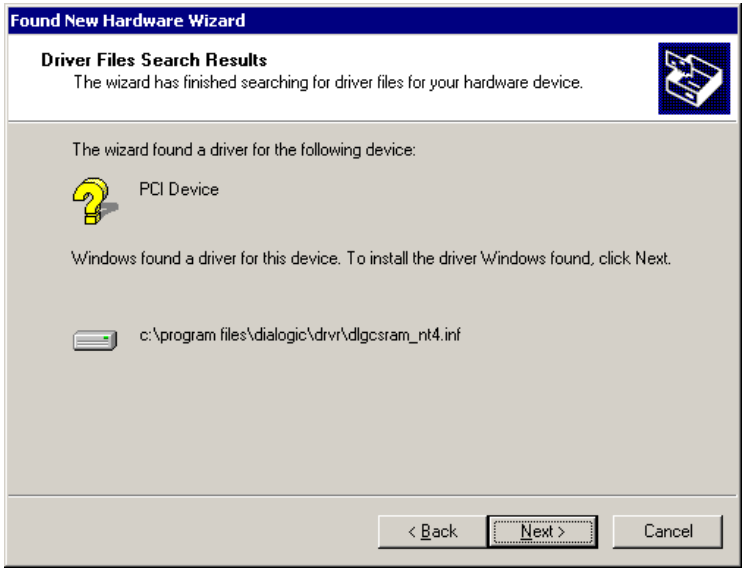


Figure 3-103 Drive found window

7. The Hardware wizard will present a completed window that looks like Figure 3-104. Click **Finish**.

Figure 3-104 Finish window

Step 3: Hardware configuration

The D/320JCTU card must now be configured. Perform the following instructions:

1. Start the DCM interface. This is where the D/320JCTU is configured at the hardware level. Figure 3-105 shows our system. Note the card ID number and how the D/320JCTU (D/320JCT #2) has the highest setting.

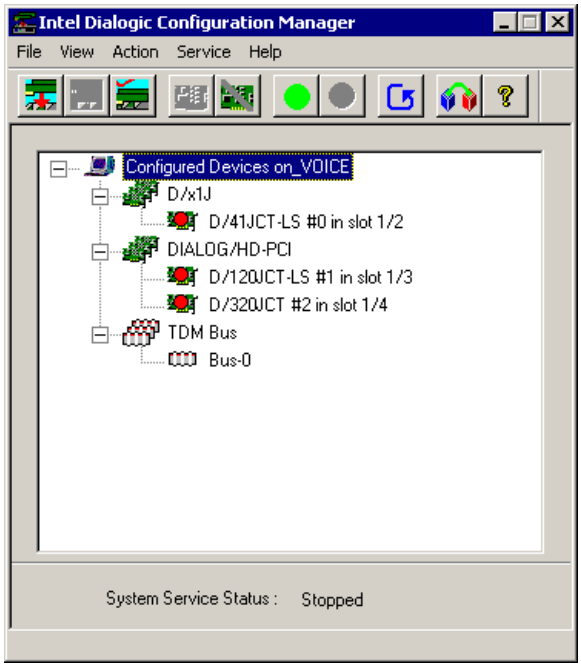


Figure 3-105 DCM of D/320JCTU card

2. Right-click the D/320JCTU card and select **Configure card**. The default parameters for the card will appear. See Figure 3-106 on page 113.

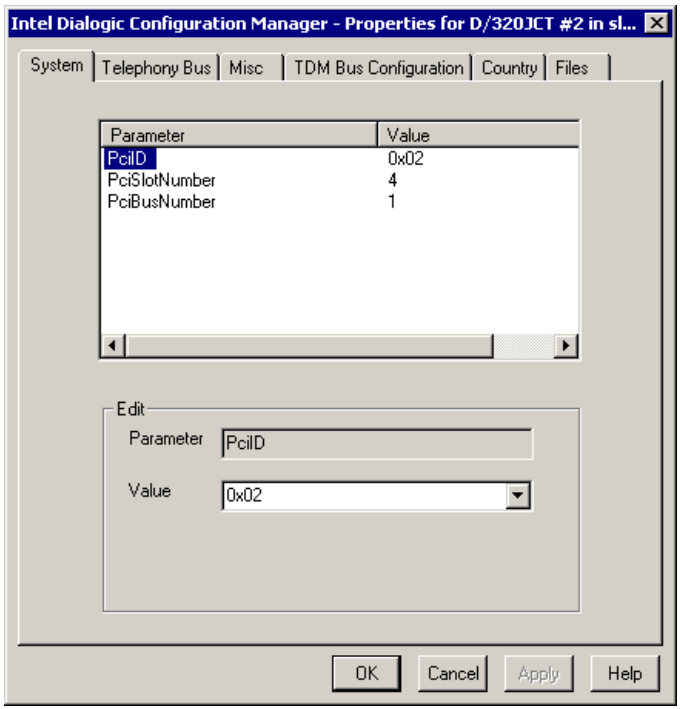


Figure 3-106 Default D/320JCTU configuration

3. Select the **Misc** tab.

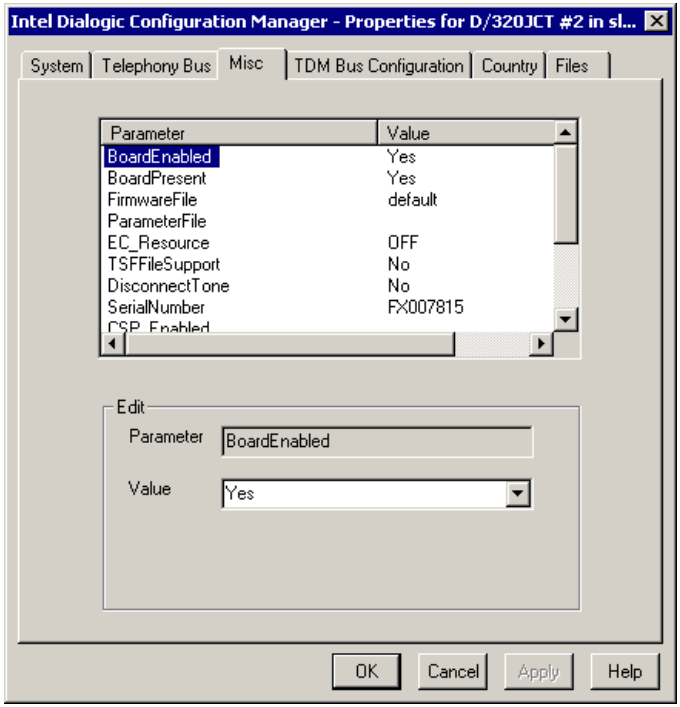


Figure 3-107 D/320JCTU Misc tab

4. To enable echo cancellation, the EC_Resource must be configured to ON. The default is OFF. A configured D/320JCTU is shown in Figure 3-108 on page 115. Click **OK** to save the setting.

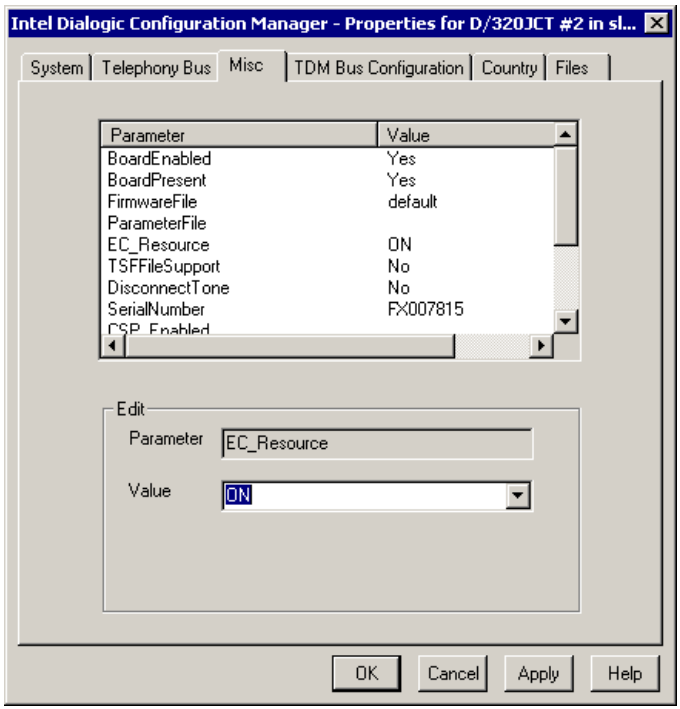


Figure 3-108 Configured D/320JCTU card

5. To activate the new settings, DCM must be started. Click the green button. A running system is shown in Figure 3-109 on page 116.

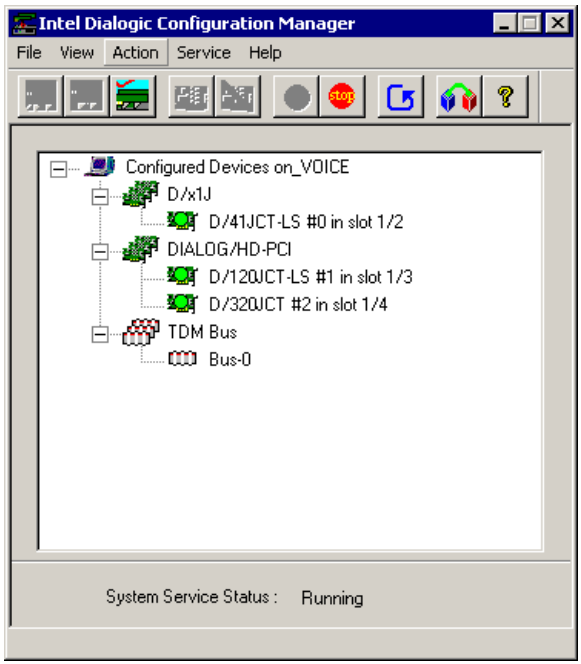


Figure 3-109 Started DCM

Note: We found that the firmware for all the cards in this environment needed to be set to default. Although our system worked when the correct firmware for each card was selected, if VXML support is disabled in WebSphere Voice Response, Voice Response would no longer start. Using the card's default firmware overcame this problem.

Step 4: Software configuration

WebSphere Voice Response needs to be configured to utilize the echo cancellation function of the newly installed card. Verify the D/320JCTU is recognized by WebSphere Voice Response. To do this, click **Start -> Programs -> WebSphere Voice Response for Windows -> WebSphere Voice Response Telephony Server Configuration**.

The card is installed and configured correctly for WebSphere Voice Response to use if the Echo Cancel Parameters window is visible. Figure 3-110 on page 117 displays a successfully installed card.

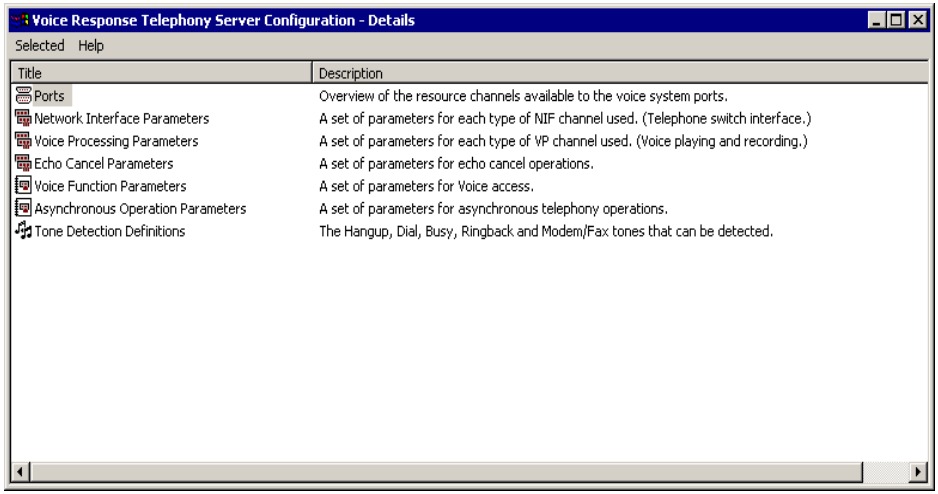


Figure 3-110 Echo cancel parameters in WebSphere Voice Response

6. Right-click **Echo Cancel Parameters** and then **Open**. You will see a window displaying the Echo Cancel Parameter - Parameter Sets, as shown in Figure 3-111. Displayed will be the total number of echo cancellation channels and the number that are being used.

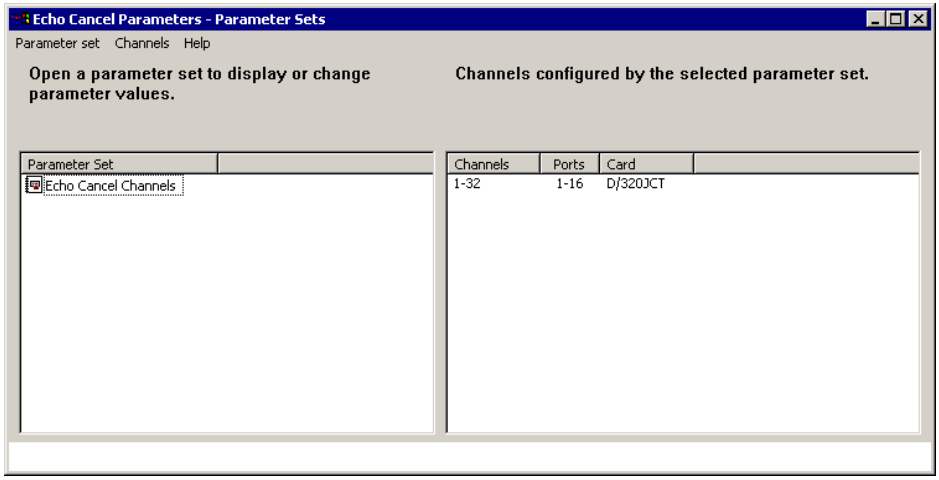


Figure 3-111 Echo cancel parameters - parameter sets

7. Right-click **Echo Cancel Channels**, then click **Open as Parameter Values**. The default parameter values will appear, as shown in Figure 3-112 on page 118.

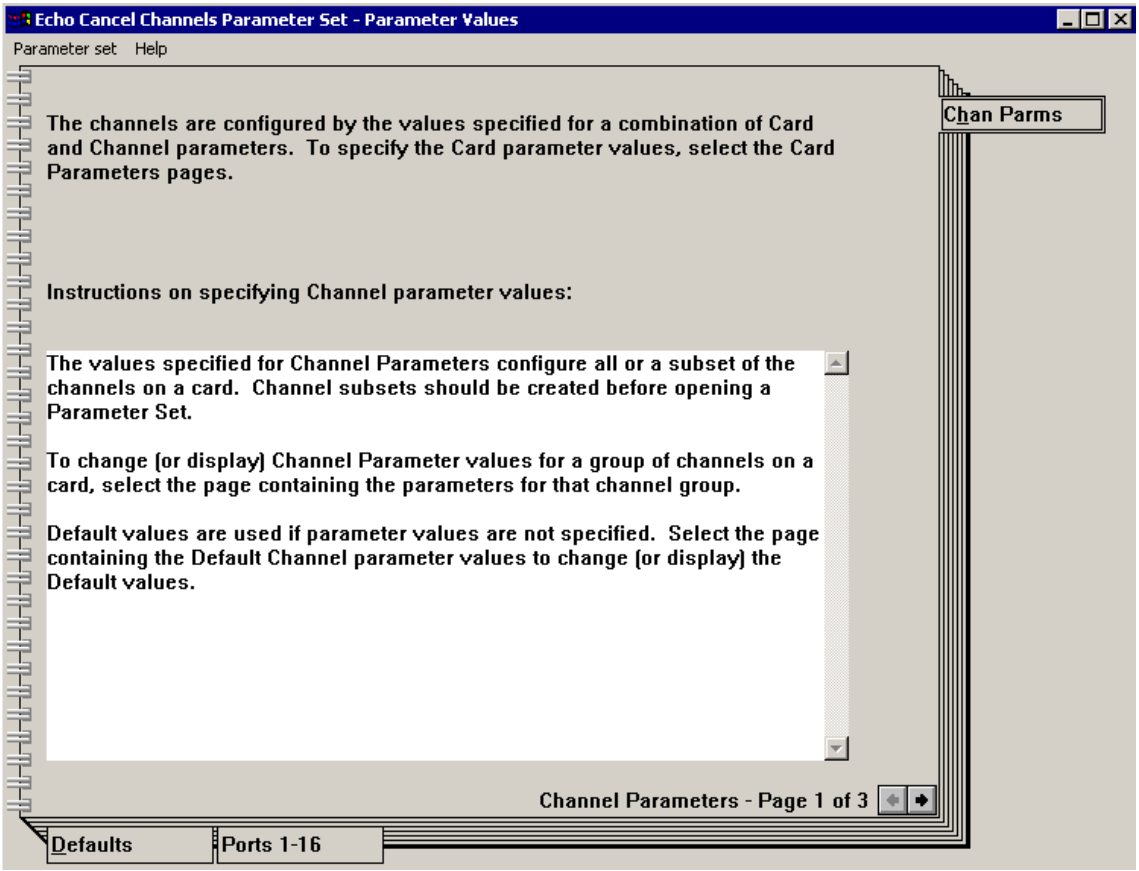


Figure 3-112 Parameter values

8. This system has a total of 16 ports available, as shown in Figure 3-113 on page 119.

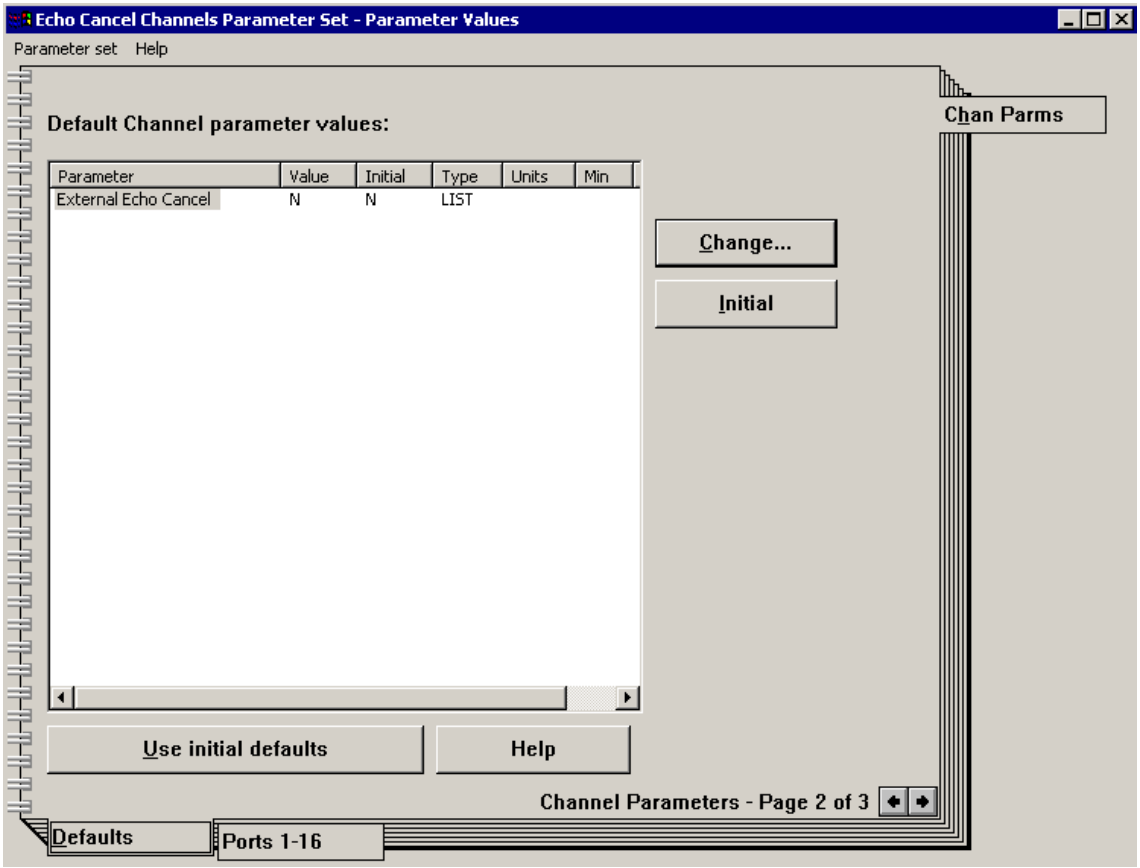


Figure 3-113 Echo cancellation ports tab

9. Echo cancellation must be turned on. Click the **Ports 1-16** tab. The External Echo Cancel default is N. This needs to be changed to **Yes**. Click the **Change** button. You will see a window similar to Figure 3-114 on page 120.

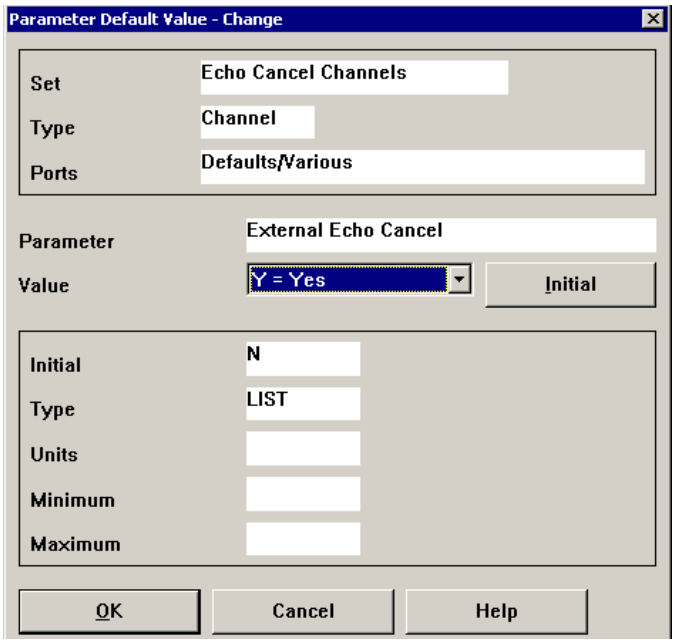


Figure 3-114 Parameter Default Value - Change

10. The Parameter Default value - Change window will appear. On the drop-down menu for the Value setting, select **Yes**.
11. Click **OK** to save. You must exit for the settings to be updated. Close the window by clicking the **X** button. The confirmation window will appear, as shown in Figure 3-115. Click **Save**.

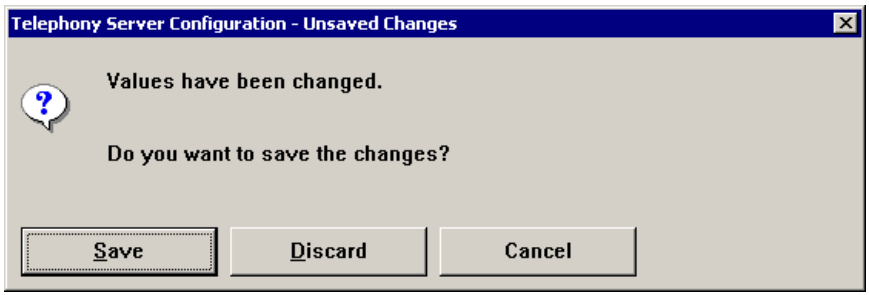


Figure 3-115 Telephony Server Configuration window

12. To verify the card has been configured and recognized by WebSphere Voice Response, start the Voice Response Configuration - Properties by clicking **Start -> Programs -> WebSphere Voice Response for Windows -> WebSphere Voice Response Configuration**.

13. Figure 3-116 shows a configured system where WebSphere Voice Response has all three cards active. If for some reason the card is disabled, highlight it and click **Enable**. The setting is saved when you close the window.

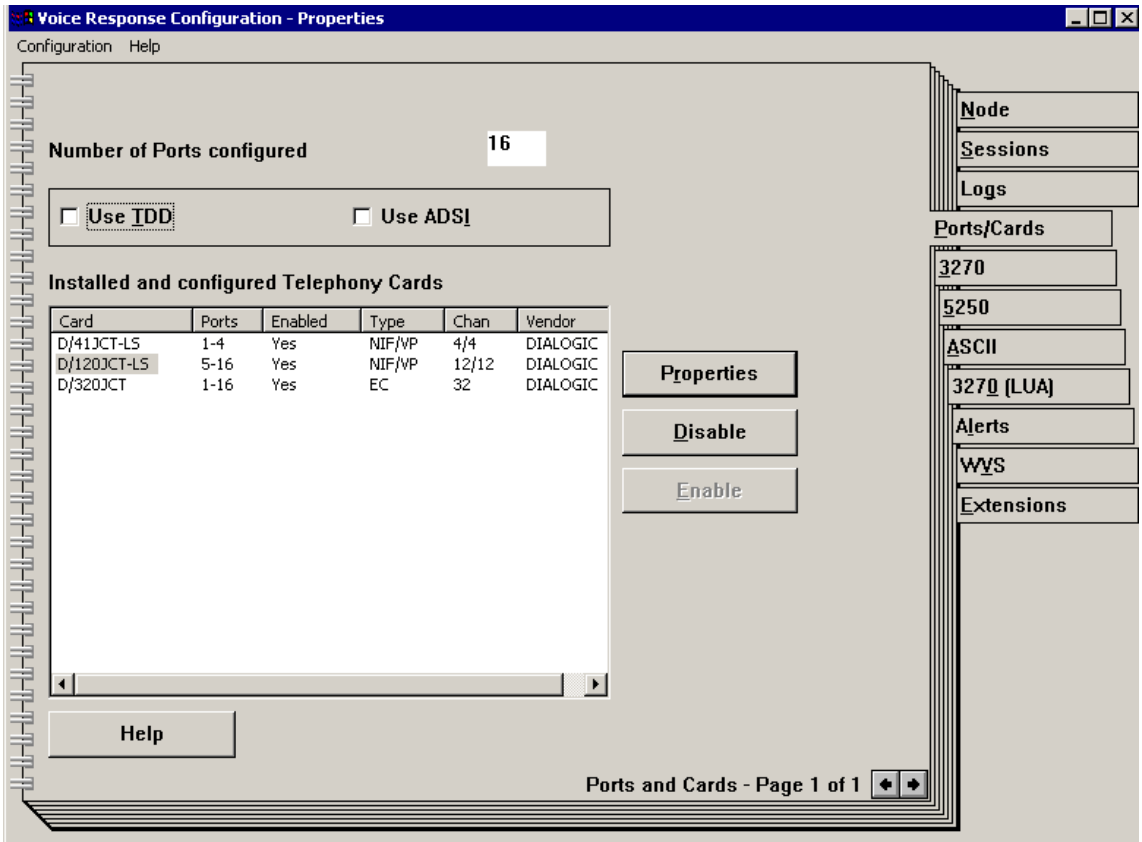


Figure 3-116 Voice Response Configuration Properties

Note: If the new card(s) installed do not appear in the Voice Response Configuration Properties window, you may need to open and save the settings in the Voice Response Telephone Server Configuration window.

3.13.2 Digital environment

We successfully tested the D/320JCTU card and a D/240JCT. This card is a T1, so it required 24 echo cancellation channels. The D/320JCTU can support up to 32 echo cancellation channels.

The installation is similar. However, the following points must be addressed:

- Use the default firmware for all of the T1 cards. Do not use the firmware specific for that card.
- Disable echo cancellation on the T1 card.
- Ensure you have the correct telephone protocol configured in WebSphere Voice Response for each card. This is set on the Network Interface Parameters option in the Voice Response Telephone Server Configuration.

Note: For other card configuration and more information, refer to the WebSphere Voice Response 3.1 CSD readme.txt file.

4

WebSphere Voice Server with WebSphere Voice Response for AIX V3.1

This chapter provides a detailed overview of how we installed WebSphere Voice Server Version 3.1 for AIX in the ITSO labs. It describes how the various components ofWebSphere Voice Server were implemented, including the prerequisite environment of WebSphere Voice Response for AIX Version 3.1. The chapter is then completed with the configurations that we used to implement speech recognition and text-to-speech for multiple languages.

4.1 WebSphere Voice Server with WebSphere Voice Response for AIX

In this section, we discuss the specific additional product information, the programming model, the prerequisites and installation instructions for WebSphere Voice Server.

4.1.1 WebSphere Voice Server

WebSphere Voice Response can use the speech recognition and text-to-speech features of the WebSphere Voice Server. However, it must be installed along with all the necessary prerequisites before Voice Server can be installed.

The baseform component of the speech recognition feature contains a description of how the words should sound. The grammar component of the speech recognition feature defined by the application development allows certain words to be detected in various forms. This provides a highly accurate detection of speech in a telephony environment.

The text-to-speech feature of WebSphere Voice Server converts a text string into audible speech that can be heard by the caller. Two types of text-to-speech are available. They are:

- Formant

  Formant text-to-speech produces speech entirely through software algorithm based on linguistics rules and human speech models.

- Concatenative

  Concatenative text-to-speech produces speech from recordings of units of actual human speech. These units of speech are concatenated according to linguistic rules, resulting in a more natural human-sounding speech. Concatenative text-to-speech requires significantly more memory, CPU processing power, and disk space.

4.1.2 Programming model

WebSphere Voice Response applications can integrate with WebSphere Voice Server in three ways. They are:

- VoiceXML
- JavaBeans
- WebSphere Voice Response state tables

VoiceXML

WebSphere Voice Response supports applications written in VoiceXML. It supports speech recognition and text-to-speech technologies.

JavaBeans

WebSphere Voice Response supports applications written in object-oriented JavaBeans. It supports speech recognition and text-to-speech technologies. IBM WebSphere Studio Application Development can be used to develop applications. JavaBeans provides the platform to enable you to extend the capabilities of VoiceXML to meet your business needs.

WebSphere Voice Response state tables

WebSphere Voice Response state tables provides a graphical means for creating and editing voice applications. It supports speech recognition and text-to-speech technologies.

Integrating different programming models

You can provide voice access to business applications easily using VoiceXML, but more advanced system requirements can be met by integrating the VoiceXML presentation layer with other programs. Figure 4-1 shows how Java is central to this kind of system. The reason for this is that the Java and VoiceXML environment include beans that allow both VoiceXML dialogs and state tables to be called from a Java program.

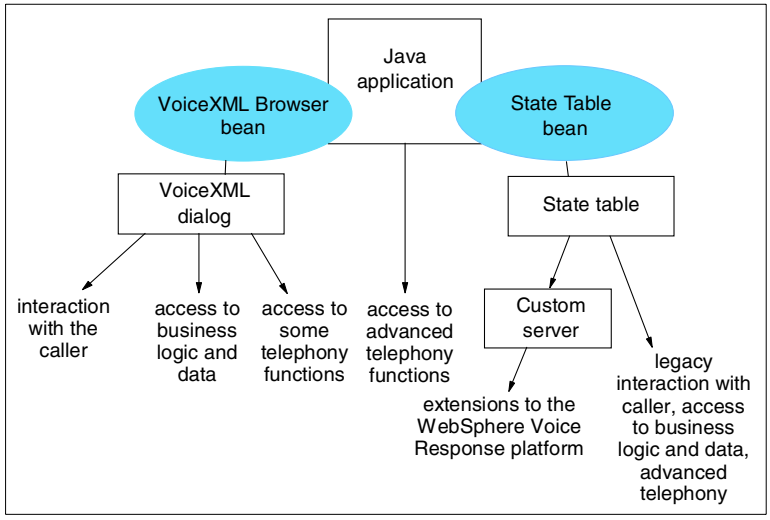


Figure 4-1 Integrating programming models

Supported features

The WebSphere Voice Server features supported in all WebSphere Voice Response application programming environments are:

- Speech recognition with Voice Server
- Barge-in using Voice Server software echo cancellation
- Barge-in using WebSphere Voice Response with echo cancellation hardware
- Allocate/free engine at start of call and keep till end of call
- Allocate/free engine when required during call
- Initial speech detection by Voice Server software
- N-Best recognition confidence scores
- Precompiled grammars

The support for other Voice Server features in each WebSphere Voice Response application programming environment is summarized in the following tables.

Table 4-1 Basic features:

Basic Features	VoiceXML	Java	State Table (AIX)
Static grammar support	Yes	Yes	Yes
Preloaded large static	Yes	Yes	Yes
Small to medium dynamic grammars	Yes	No	Yes
Initial Speech detection by WebSphere Voice Response	Yes (change default to use)	Yes (change default to use)	Yes
N-Best recognition confidence scores	Yes	Yes	Yes
Precompiled grammars	Yes	Yes	Yes

Table 4-2 Advanced speech recognition

Advanced Speech Recognition Feature	VoiceXML	Java	State Table (AIX)
Voice Enrollment	No	No	Yes
Additional detailed result, score, and timing information by word	No	No	Yes
Change Pool path setup of system for grammars dynamically	No	No	Yes
Change default code page used byWebSphere Voice Server for inline grammars	No	No	Yes
DBCS recognition results strings	Yes	No	Yes

Table 4-3 Text-to-speech features

TTS Feature	VoiceXML	Java	State Table (AIX)
Play TTS strings	Yes	Yes	Yes
Play TTS from files	No	No	Yes
Use IBM TTS control attributes to modify speed, control of speech	No	Yes [1]	Yes [1]
Use VoiceXML TTS control tags to control TTS	Some but not all are supported [2]	No	No
DBCS play text strings	Yes	No	Yes

[1] Refer to *WebSphere Voice Server for AIX Application Development with State Tables*
[2] tags are supported. Changing volume by % or pitch by % is not supported. <emp> tag is not supported for concatenative TTS. More detailed information can be found in the *VoiceXML Programmers Guide.*

Where to find further information

See the following for further information on developing voice applications in a particular application programming environment supported by WebSphere Voice Response:

- VoiceXML

  Refer to the VoiceXML Programmer's Guide. Information on the VoiceXML Java bean is included in *WebSphere Voice Response Application Development using Java and VoiceXML* manual (referred to hereafter as *Application Development using Java and VoiceXML*). You can also refer to the Voice eXtensible Markup Language (VoiceXML) specification, Version 2.0 (available at `http://www.voicexml.org/`).

- Java

  Refer to the *Application Development using Java and VoiceXML* manual. For information on developing grammars, refer also to the *WebSphere Voice Server Application Development with State Tables* and (if you are running Windows 2000) the IBM WebSphere Voice Toolkit online help.

- State table

  Refer to *WebSphere Voice Server for AIX Application Development with State Tables* (referred to hereafter as *Application Development with State Tables*).

4.1.3 Architecture

The architecture is slightly different for each application programming environment (see Figure 4-2 on page 129). A distributed architecture enables the WebSphere Voice Server speech recognition and text-to-speech engines to be shared as resources between WebSphere Voice Response, JavaBeans, or VoiceXML applications.

Speech recognition and text-to-speech both require a considerable amount of CPU resource. The speech recognition and TTS servers can be installed on machines connected by a local area network to the machines on which WebSphere Voice Response is running. This then shares the processor requirements over the machines. It ensures that the performance of other voice applications running on the WebSphere Voice Response machines do not deteriorate during speech recognition or text-to-speech operations.

The distributed architecture also improves the redundancy of the system, in the event of a machine failing or being taken out of service for maintenance. Most of the components of this client/server architecture are standard WebSphere Voice Response solution components. The only additions to the architecture are the hardware required to run the speech recognition software, and/or the text-to-speech software. Figure 4-2 shows the architecture of a voice system using WebSphere Voice Server speech technologies.

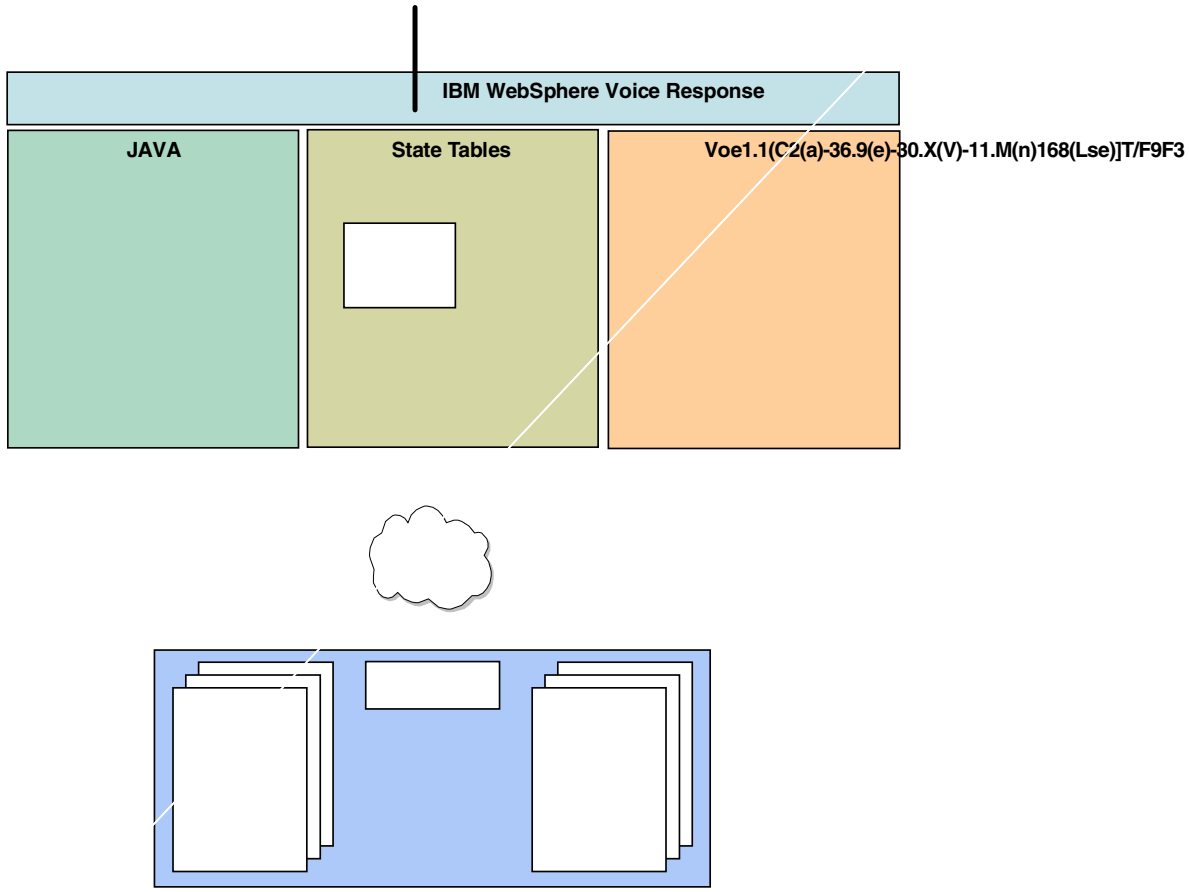


Figure 4-2 Architecture of WebSphere Voice Server speech technologies

4.1.4 WebSphere Voice Server for AIX V3.1 prerequisites

WebSphere Voice Server 3.1 for AIX has a prerequisite of WebSphere Voice Response for AIX 3.1. The following are the minimum hardware requirements of the WebSphere Voice Response for AIX V3.1 telephony platform:

- RS/6000 or pSeries 223 MHz or higher
- CD-ROM drive
- 512 MB RAM minimum required
- 500 MB of disk space minimum required per language install
- Local network: 10/100 Mb Fast Ethernet

Note: The system will require a separately orderable WebSphere Voice Response for AIX, V3.1 (with PTF U483599 installed) product. The general WebSphere Voice Response for Windows hardware requirements are identified in the *WebSphere Voice Response for AIX General Information and Planning Guide*.

The following are the minimum software requirements when used with the WebSphere Voice Response for AIX V.31 telephony platform.

- AIX 4.3.3 Maintenance Level 9 or AIX 5.1
- WebSphere Voice Response for AIX, V3.1 (with PTF U483599 installed)

4.2 WebSphere Voice Server installation

This section describes the test environment we had in the ITSO labs. Covered is the installation and configuration of the various components of the WebSphere Voice Server on the RS/6000.

4.2.1 Telephone structure

The RS/6000 system used was a 44P Model 170. The system was interconnected to a Lucent G3r PBX via a digital T1 trunk. The PBX is sourced by Avaya. Figure 4-3 on page 131 shows this configuration.

In our setup the telephone extension 24141 was used to call the digital T1 channel number (22001). It was then answered by WebSphere Voice Response, running the sample Weather VoiceXML application that is delivered with the product. The sample application utilizes multiple languages, which the caller can select and then communicate and interact with.

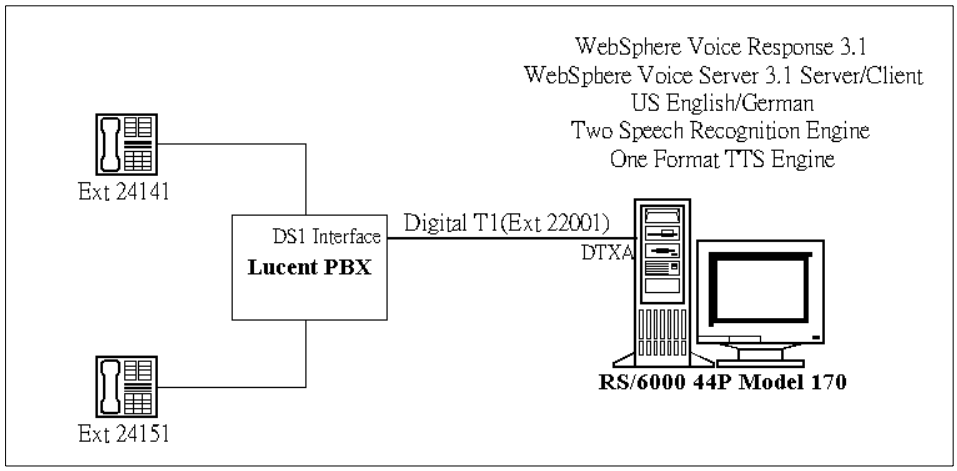


Figure 4-3 Telephony structure for WebSphere Voice Server in ITSO lab

The setup was configured to support both US English and German simultaneously. Two speech recognition engines and one formant text-to-speech engine for the language installed (in our case two, one for English and one for German) were installed on the RS/6000 Model 170. In this configuration, the RS/6000 functioned as both a Voice Server server and as a Voice Server client.

Note: With this environment we were able to write several simple VoiceXML applications for specific testing purposes. One of these tests was the VoiceXML Transfer function.

4.2.2 Computer hardware

As mentioned before, the machine used was an IBM RS/6000. The specifications of the system were:

- RS/6000 44p Model 170 single processor, 2 GB RAM, 12 GB hard disk
- ARTIC960RxD Quad Digital Trunk Adapter (DTXA)
- 100 Mbps Ethernet adapter

4.2.3 Software

Figure 4-4 on page 132 shows the software components installed on the RS/6000 Model 170.

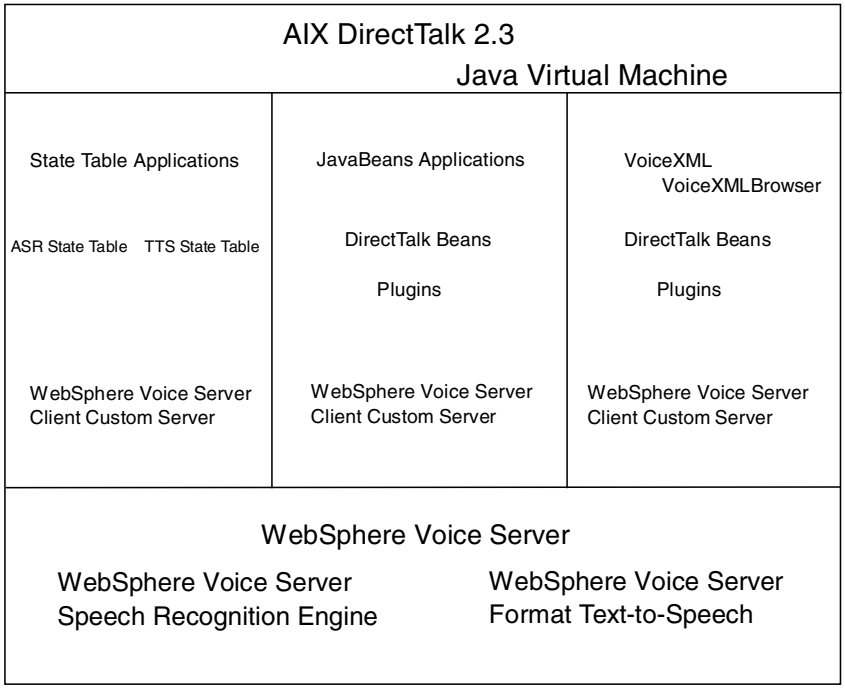


Figure 4-4 Software components of ITSO RS/6000 used

The software level of each component was as follows:

- AIX Version 4.3.3.50
- AIX Java Runtime Engine 1.3.0.14
- C Set++ for AIX Application Runtime 5.0.2.1
- WebSphere Voice Response for AIX Version 3.1.1 with fix level 2.3.0.5100
- WebSphere Voice Server for AIX with Speech Recognition Version 3.1
- WebSphere Voice Server for AIX with Formant Text-to-Speech Version 3.1

4.3 AIX configuration

Before any software is installed, the AIX operating system level must be verified. The current operating system level is found by running the command **oslevel**. In our case it was 4.3.3.0. Running the following command will show the maintenance levels of the operating system:

```
oslevel -g | grep bos.64bit
```

If the actual level is 4.3.3.50, the AIX operating system is at level 8, which is supported by WebSphere Voice Server. If the level is lower than 4.3.3.50, you must update the AIX system.

No AIX system tuning parameters were changed for our implementation. However, if your system is running in full production, it may be desirable to alter some of the operating characteristics of the operating system. The default settings of the most commonly modified system parameters can be examined by executing the command **`lsattr -El sys0`**. System performance tuning is a complex process and is not discussed in this book.

Changes to AIX system parameters can be made via the System Management Interface Tool (SMIT). This is the recommended tool to use, since it is less prone to typographical and syntactical errors than using line commands. It also has the advantage of generating two log files, smit.script and smit.log, in the root directory. These provide an audit trail of the configuration work that has actually been carried out. These two files tend to grow very large over time and should be removed or archived periodically.

Note: We used **`smit`** for the majority of the systems administration and configuration work.

We used smit to do two actions:

1. Enlarge the space for folders:
   - Enter the command **`smit`**.
   - Click **System Storage Management -> File Systems Add / Change / Show / Delete File Systems -> Journaled File Systems -> Change / Show Characteristics of a Journaled File Systems.**
   - From the list, select the directory.
   - In the SIZE of file system box, input a number sufficient for the folder.
   - Click **OK** to implement.
   - When finished, press F12 to exit.
2. Add the user to a group:
   - Type the command `smit`.
   - Click **Security & Users -> Groups -> Change / Show Characteristics of a Group**.
   - Click **List** and select the proper group. Click **OK** to return.
   - Click **List** in the USER list and select users to add. Click **OK** to return.
   - Click **OK** again to implement.

– When finished, press F12 to exit.

Note: We always used **smitty**, which is the text-mode for **smit**, for the installation and upgrading of all software in this project.

There are three actions using smitty as below:

1. The **smitty update_all** command is used to update the level of software.
   – Change the current directory to the target directory that contains the .bff files to update.
   – Type the command **smitty update_all**.
   – Input the directory name as ./, then press Enter.
   – Highlight **PREVIEW only** and press F4. Select **No**. Then press Enter to come back.
   – In the main menu, press Enter to go to the Confirmation window. Press Enter again to start the update.
   – When finished, check the upper left corner to see if the update is OK or not. If it has failed, study the detailed information in the output to correct the error.
   – Press F10 to exit.
2. The **smitty install_latest** command is used to install the new software.
   – Change the current directory to the target directory that contains the .bff files to be installed.
   – Type the command **smitty install_latest**.
   – Input the directory name as ./, then press Enter.
   – Highlight **Software to install** and press F4. Select the multiple relevant files using F7. Then press Enter to come back.
   – Highlight **PREVIEW only** and press F4. Select **No**. Then press Enter to come back.
   – In the main menu, press Enter to go to the Confirmation window. Press Enter again to start the install.
   – When finished, check the upper left corner to see if the update is OK or not. If it has failed, study the detailed information in the output to correct the error.
   – Press F10 to exit.

3. The **smitty install_remove** command is used to remove the installed software.
   - Type the command **smitty install_remove**.
   - Highlight **Software name** and press F4. Select the multiple relevant files using F7. Then press Enter to come back.
   - Highlight **PREVIEW only** and press F4. Select **No**. Then press Enter to come back.
   - In the main menu, press Enter to go to the Confirmation window. Press Enter again to start the update.
   - When finished, check the upper left corner to see if the update is OK or not. If it has failed, study the detailed information in the output to correct the error.
   - Press F10 to exit.

4.4 Installation of WebSphere Voice Response for AIX

The RS/6000 is used as both WebSphere Voice Server (server and client) and WebSphere Voice Response. WebSphere Voice Response requires the ARTIC960RxD Quad Digital Trunk Adapter (DTXA) to provide the telephony connection to the PBX or CO.

4.4.1 Hardware installation of WebSphere Voice Response for AIX

Our first step was to ensure the RS/6000 was capable of supporting the DTXA telephony card. This check is essential, since certain RS/6000s do not support the DTXA card. The card needs to be installed in a free full-length PCI slot. It is recommended that either slot 1 through 3 is used.

Note: Do not change the card between slots. Otherwise, it will appear there are two cards in the system configuration. One status is defined and another is available. You must use **diag -a** and **cfgmgr** to get rid of the defined one manually.

4.4.2 Software installation of WebSphere Voice Response

It is important to note that WebSphere Voice Response for AIX Version 3.1 manages its internal database with DB2 Version 7. This is supplied as part of the WebSphere Voice Response filesets. If your machine is installed with DB2, you may need to ensure that your instance of DB2 does not conflict with the one supplied with WebSphere Voice Response.

The installation of WebSphere Voice Response for AIX Version 3.1 is carried out using **smitty**, with the root user logon. The procedure is as follows:

1. Ensure that there is sufficient disk space for /usr and /var using **`df -k`**. You need 10 GB for /usr and 200 MB for /var.
2. Install x11.samples.lib.Core 4.3.3.0 from the AIX installation CD 3.
   a. Type the command **`smit install_latest`**.
   b. Select **List** in the INPUT device/directory.
   c. Select **/dev/cd0** and click **OK** to come back.
   d. Select **List** in the SOFTWARE install.
   e. Click **Find**. Input **x11.samples** and Click **OK**. Select **x11.samples.lib.Core 4.3.3.0** in the subitem of x11.samples. Click **OK** to come back.
   f. Click **OK** twice. When finished, check the status in the right center. If it has failed, study the detailed information in the output to correct the error.
   g. Click **Done** and **Cancel** to exit.
3. Update x11.samples.lib.Core level to 4.3.3.10 with **`smitty update_all`**. (For detailed usage, please refer to step 1 on page 134.)
4. Using **`smitty install_remove`**, select **ifor_ls.msg.en_US.base.cli** and **ifor_ls.msg.en_US.base.gui**.
5. Change the current directory to /aix43. Issue the command **`smitty install_latest`**, and select the all filesets.

   Wait for the installation to complete. There is a warning about a file owned by fileset bos.rte.ifor_ls when installing ifor_ls.library. Skip it.
6. Change the current directory to /base. Use **`smitty install_latest`** to select from the following list of components (for detailed usage, refer to step 1 on page 134):

- Java130.adt
- Java130.rte
- X11.vfb
- db2_07_01.client
- db2_07_01.cnvucs
- db2_07_01.conn
- db2_07_01.cs
- db2_07_01.das
- db2_07_01.db2
- db2_07_01.elic
- db2_07_01.tspf
- devices.artic960

- devices.artic960dev
- devices.dirTalk.artic960
- dirTalk.DT
- dirTalk.DTBE
- dirTalk.ISDN.ATT
- dirTalk.DMS100
- dirTalk.ISDN.Euro-ISDN
- dirTalk.ISDN.INS1500
- dirTalk.ISDN.com
- dirTalk.SP
- ipfx
- vacpp.cmp
- vacpp.ioc

Note: Do not select the class libraries for AIX 4.1,4.2 and 5.0 items in vacpp.ioc.

7. Another user is also created during the installation, called dtdb23in. You must specify the password for the user during the installation or it will be skipped. This user is implicitly used by WebSphere Voice Response to manage its database. In normal operating conditions, there is no need to log on as the dtdb23in user.
8. When installation is complete, exit smitty.

4.4.3 Creation of an administrator

Perform the following steps to create an administrator:

1. During the installation, a user named dtuser is created.
2. Use `smit` to set the password for dtuser:
   a. Select **Security & Users** -> **Users** -> **Change a User's Password.**
   b. Click **List** in the User NAME. Select **dtuser**. Click **OK**.
   c. Input the dtuser's new password twice.
   d. Click **OK** to confirm.
   e. Click **Cancel** four times to exit.

   If other optional WebSphere Voice Response features were installed, more users will automatically be created. For instance, if the optional feature SS7 is selected, two users, named ss7user and hauser, will be created.

   You may have to consider if the automatically created users may by chance have the same user name as your existing user.

3. Use the following command to set the file permissions and ownerships:

   ```
   /usr/lpp/dirTalk/tools/vae.setuser
   ```

   If this is the first time the **vae.setuser** command is issued and since the database has not been created yet, you will notice DB2 error messages similar to those shown in Figure 4-5. Ignore the DB2 error messages.

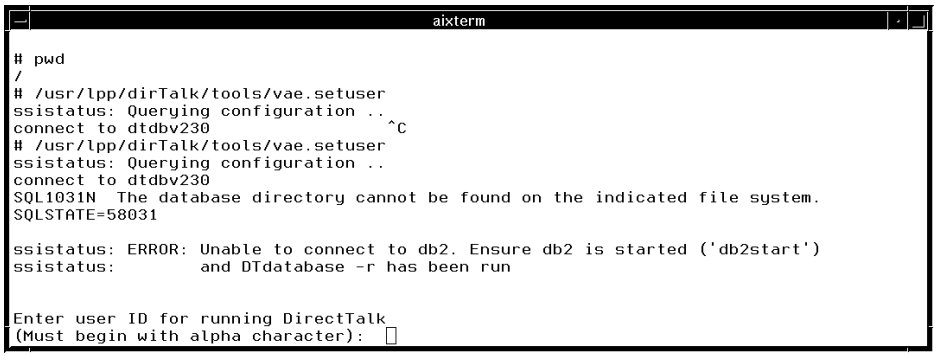


Figure 4-5 DB2 error messages for first time dtuser log on

When prompted to enter the user, type `dtuser` as the user name.

When the **vae.setuser** is running, a list of messages will be displayed. Some of the messages are shown in Figure 4-6. The last message will show that the command performs a checksum test on the files system. If the test is correct, no further error messages will be displayed.

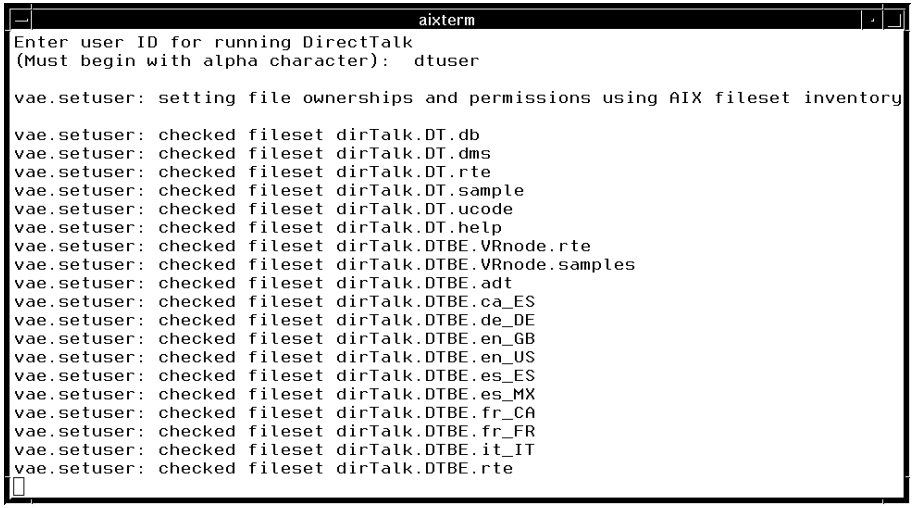


Figure 4-6 Some of the messages displayed with the vae.setuser command

4.4.4 Installation of PTFs for WebSphere Voice Response

The next step is to install PTFs for WebSphere Voice Response. Perform the following steps:

1. Change to the PTF directory and **smitty update_all** is used. (For detail usage, please refer to Page 134.)

   When installing software updates, there is an option to commit or not to commit the software updates. It is impossible to remove the software updates once they are committed. It is advisable not to commit the software. This means they can be backed off later if required. After the installation is finished, use **lslpp -l dirTalk.DT.rte** to check the fileset version is 2.3.0.5100.

   After the update, run the **. /usr/lpp/dirTalk/tools/vae.setuser** command. This will ensure that the file permissions and ownership belong to dtuser.

2. Shut down and restart AIX with the **shutdown -rF** command. This will make the newly installed software effective. The DTXA is loaded with the latest device driver during the restart.

4.4.5 Create database for WebSphere Voice Response

Perform the following instructions to create a database for WebSphere Voice Response:

1. Log on as root and check that there is sufficient disk space in the /home directory to hold the WebSphere Voice Response database that is about to be created.

2. To create the database, set the environment for dtuser using this command:

   ```
   . /usr/lpp/dirTalk/tools/vae.setenv
   ```

   The environment is correctly set if the output of the **echo $VAE** command is:

   ```
   /usr/lpp/dirTalk
   ```

   Ensure the size of /tmp is 169 MB or more using **df -k /tmp**. (For details on usage, please refer to step1 on page 133.)

   The command to create the database is:

   ```
   /usr/lpp/dirTalk/tools/DTdatabase -r
   ```

   A list of messages will be displayed.

4.4.6 Grant owner for DTXA

The next step is to assign WebSphere Voice Response as the owner of the DTXA card. The card was identified with the **lsdev -C** command, which gave the last digit of the DTXA identifier, riciop0, as 0. To assign WebSphere Voice Response as the owner of the DTXA card, issue this command:

```
dt_setowner -s0
```

To confirm that the command is successfully executed, run the command:

```
lsdev -C
```

This command is used to display the names, bus locations and all the associated device drivers of the RS/6000. The partial output of the command is shown in Figure 4-7.

```
riciop0    Available 10-71        IBM ARTIC960RxD Quad Digital Trunk PCI Adapter
ddriciop0  Available 10-71-00     IBM ARTIC960RxD PCI Device Driver
dtline0    Available 10-71-01     Artic card currently owned by WebSphere Voice Response
dtdd       Available              WebSphere Voice Response system-wide Telephony Kernel
Extension
dtxa0      Available -00          WebSphere Voice Response Digital Trunk X Adapter
dtpack0_0  Available -00-00       WebSphere Voice Response Digital Trunk Processor.
dtpack0_1  Available -00-01       WebSphere Voice Response Digital Trunk Processor.
dtpack0_2  Available -00-02       WebSphere Voice Response Digital Trunk Processor.
dtpack0_3  Available -00-03       WebSphere Voice Response Digital Trunk Processor.
```

Figure 4-7 Partial output of lsdev -C command showing DTXA

The device DTXA is known as riciop0 and the device driver is named ddriciop0. Each DTXA has four digital trunk packs. Only one digital trunk pack, dtpack0_0, is used in our configuration. WebSphere Voice Response is the registered owner of this pack, as indicated by dtline0.

Change the current directory to /usr/lpp/devices.artic960/bin and run **ricdiag**.

1. Select 1: Test all ARTIC960 adapters.
2. Select 2: Run Interface Board with internal wrap tests. No external wraps are required.
3. Select 1: Run tests one time.

The result is shown in Example 4-1.

Example 4-1 Result window of diagnosed card

```
      Enter choice: 1
```

```
      The following ARTIC960 adapter(s)
      have been selected for test and
      successfully opened.
                                        Interface
      Logical card #      Slot #        Board ID
      --------------      ------        --------
            0              255             d31014

   -- Press Enter to continue --
Type a 'q' to quit.

Card # = 0   Slot = 255   Pass = 1
Executing Diagnotic Load
Test Completed Successfully

Card # = 0   Slot = 255   Pass = 1
Executing Processor Test
Test Completed Successfully

Card # = 0   Slot = 255   Pass = 1
Executing Timer Test
Test Completed Successfully

Card # = 0   Slot = 255   Pass = 1
Executing Bus Interface Test
Test Completed Successfully

Card # = 0   Slot = 255   Pass = 1
Executing Memory Test
Test Completed Successfully

Card # = 0   Slot = 255   Pass = 1
Executing Interface Board Test

 ------ INSTRUCTION MESSAGE -----
ARTIC960 Co-Processor Platform
Quad T1/E1 Card
Testing without the wrap plug has been selected. No external wrapping will be
performed.

    -- Press Enter to continue --
```

Note: If an error occurs, there must be some error with the hardware environment. It is recommended that you change the DTXA card or seat it again.

4.4.7 License Use Management

WebSphere Voice Response for AIX Version 3.1 is license-enabled software that must be enrolled and managed using the IBM License Use Management (LUM) product. LUM was installed with the AIX system.

Software licensing comprises three steps:

1. Configure and identify the machines in your network to be license servers
2. Enroll the WebSphere Voice Response licenses
3. Distribute the WebSphere Voice Response licenses

Create a license server

Complete the following steps to create a license server:

1. Start the configuration tool with the **i4cfg** command. Figure 4-8 shows the selections we used in our setup. To satisfy our environment, the following boxes are checked:

- Network License Client
- Network License Server (NetworkLS)
- Central Registry License Server (CrLS).

The NodeLocked Licensed Server (NodeLS) must remain unchecked.

If this environment is different from your network environment, you may need to check different boxes.

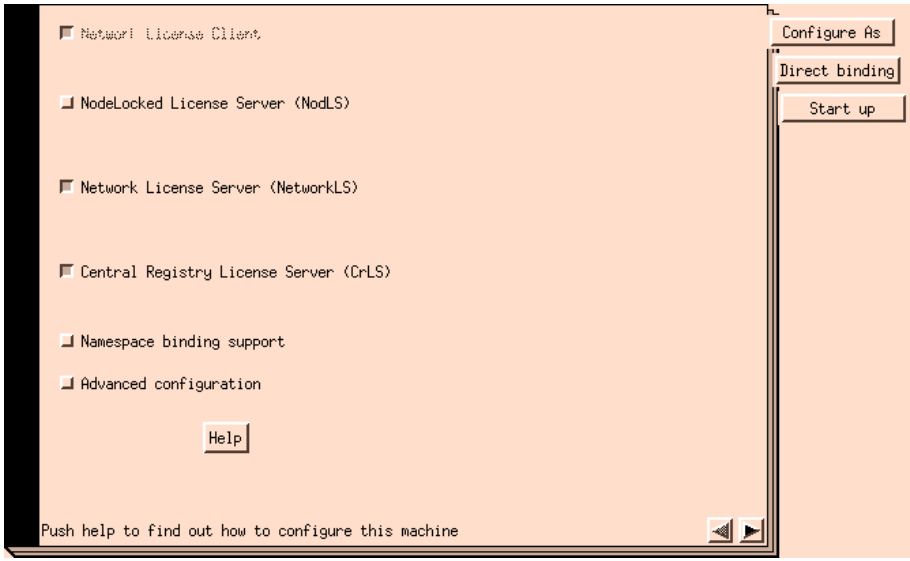


Figure 4-8 Configuration tool

2. From the Direct Binding window, input the host name aixwvs3 as server name and select **NetWorkLS** and **Central Registry**. Then click **Add.** The RS/6000 with the host name of aixwvs3 is selected as the network license server (NetworkLS). This is shown in Figure 4-9.

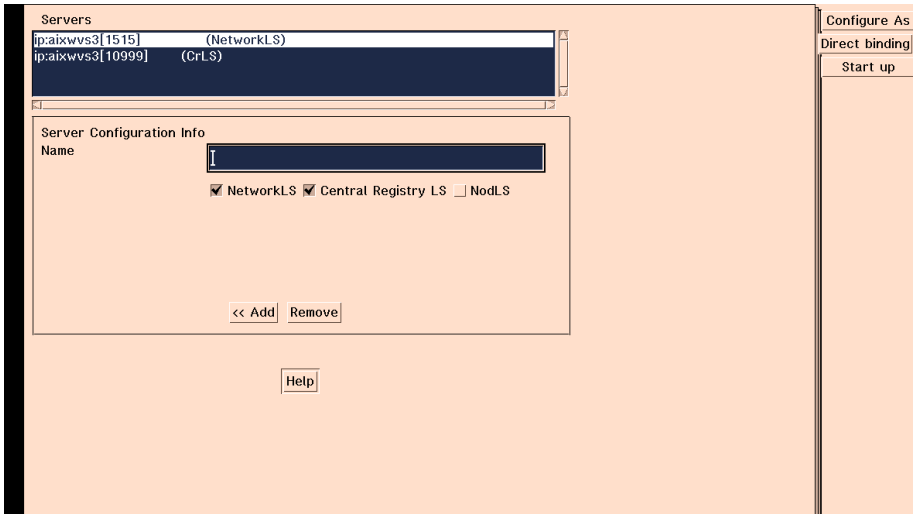


Figure 4-9 Host aixwvs3 selected as network license server

3. Select the **Startup** tab and select **Start Services at system startup**.

   When the configuration is successfully completed, press Alt+F4 and you will see a window similar to Figure 4-10.

Figure 4-10 Configuration tool

4. As per the instruction on the final Configuration tool window, start the LUM with the **`i4cfg -start`** command. The successful completion of the command is shown in Figure 4-11 on page 144. This completes the configuration step of LUM.

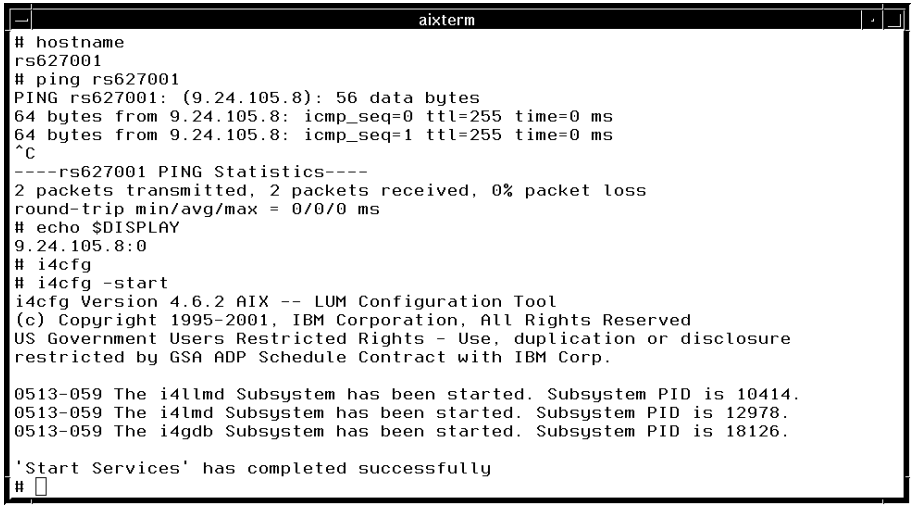


Figure 4-11 LUM runtime started successfully

Enroll license

The next step is to enroll the WebSphere Voice Response base product license and the optional licensed features. Perform the following steps:

1. To start the LUM Basic License Tool, use the **`i4blt`** command.

   From the Basic License Tool window, select **Products -> Enroll -> Multiple certificates** to bring up the Import window. In the Filter field at the top of the Import window, type `/usr/lpp/dirTalk/db/license/*.lic` and press Enter. This will bring up all the licenses that you purchased and will display them in the Files area, as shown in Figure 4-12 on page 145.

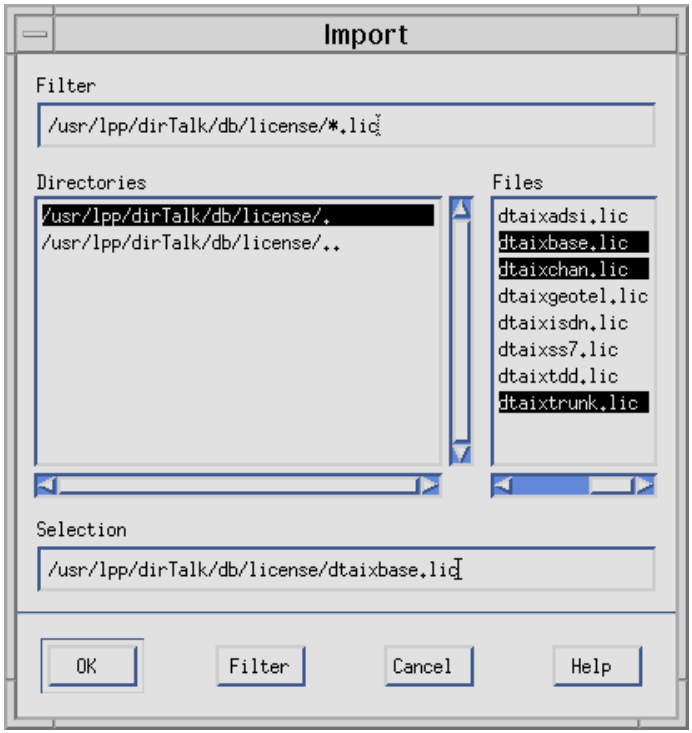


Figure 4-12 Import window showing all licenses purchased

2. We selected three licenses to be enrolled. They are:
   - dtaixbase.lic
   - dtaixchan.lic
   - dtaixtrunk.lic
3. Click **OK**. You will see a window similar to Figure 4-13 on page 146.

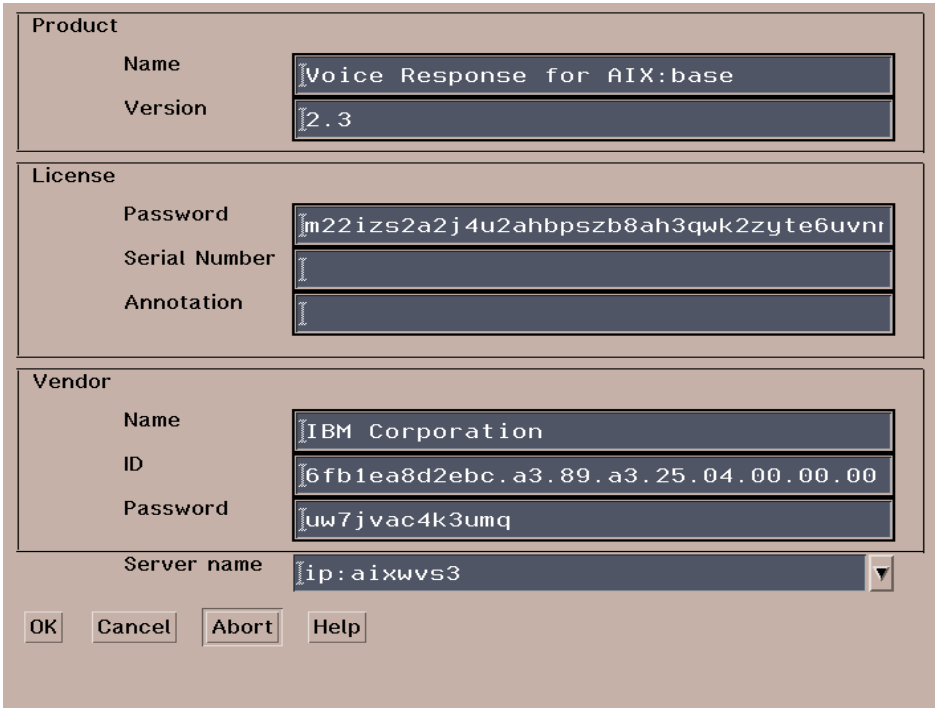


Figure 4-13 Product enrollment

4. Click **OK**. You will see a window similar to Figure 4-14 on page 147. This window does not explicitly show you which product license you are about to enroll. Therefore, you need to take note of the Selection field (Figure 4-12 on page 145), which is the Import window.

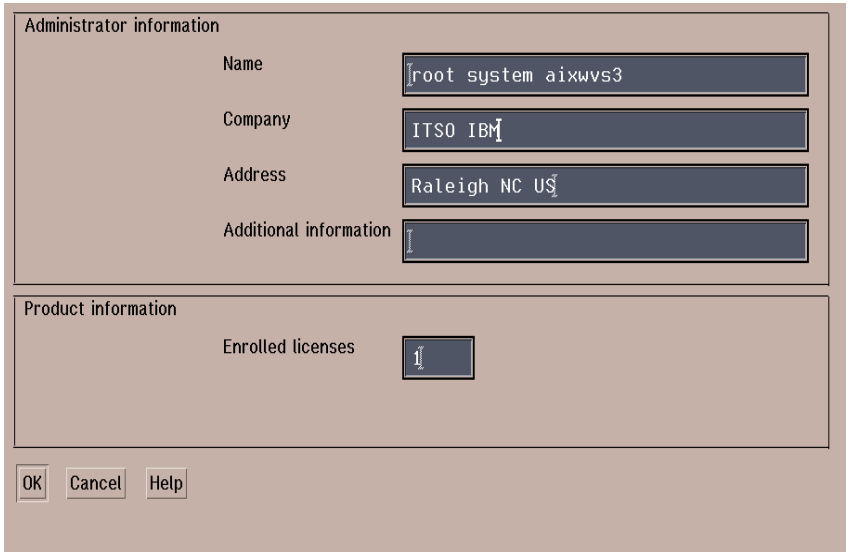


Figure 4-14 Enrolled licenses window

5. Click **OK** after all the fields are filled. The same steps are applied to enroll the WebSphere Voice Response channel license and WebSphere Voice Response DTXA trunk license.

   The following licenses are enrolled:

   – 1 WebSphere Voice Response base license, as only one RS/6000 is used.

   – 24 channel licenses, as we intend to use all 24 channels of the T1 trunk.

   – 1 trunk license of the DTXA card. If we were to use two trunks of the DTXA card (which can support up to four trunks), we need to enroll two trunk licenses. Accordingly, the number of channel licenses may also need to be increased beyond 24.

   If the product has already been enrolled, the error message shown in Figure 4-15 appears.

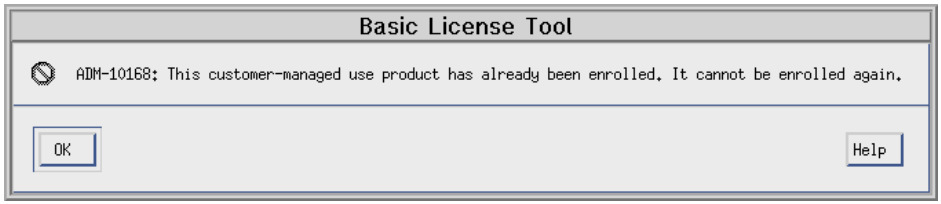


Figure 4-15 Attempt to enroll a product already enrolled triggers this message

Distribute licenses

To distribute the license, use the Basic License Tool. Perform the following steps:

1. From the Basic License Tool, right-click the WebSphere Voice Response AIX license entry and select **Distribute licenses** from the drop-down menu. The display selected from the WebSphere Voice Response for AIX:channel product is shown in Figure 4-16.

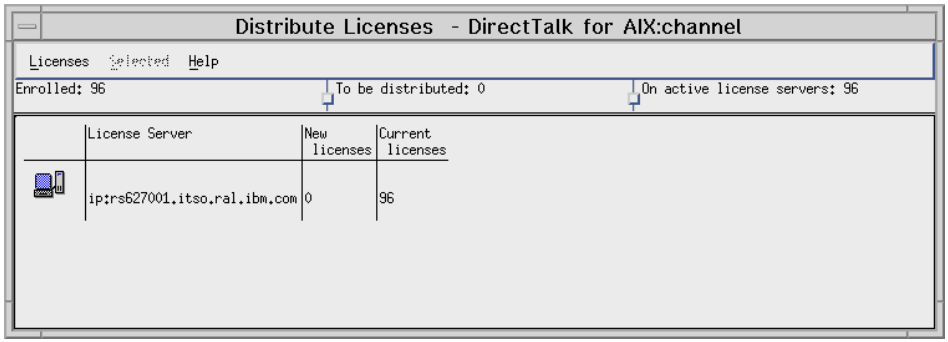


Figure 4-16 Distribute Licenses window

2. Then right-click the product and select **Set number of licenses**. Input `1` and close it.
3. Click **License -> Distribute**.

   Finally, the Enroll window is displayed similar to Figure 4-17 on page 149:

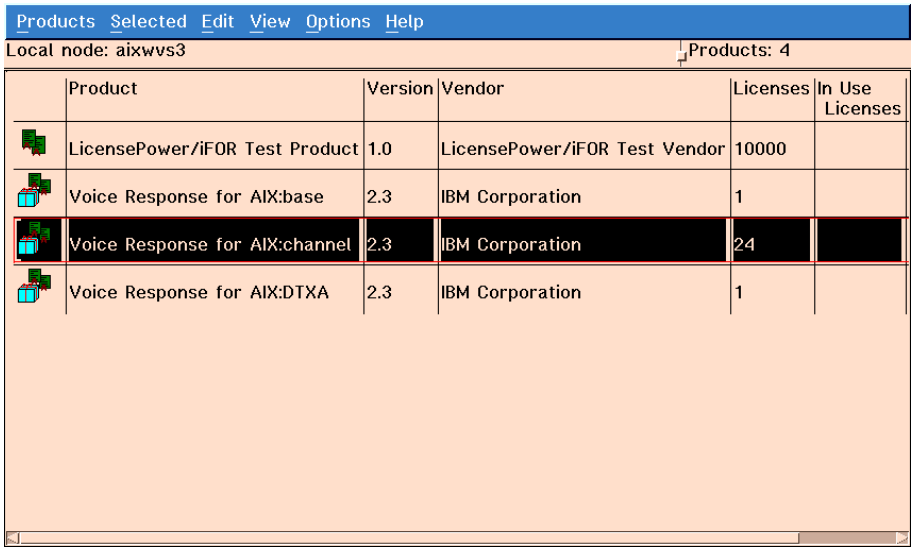


Figure 4-17 Basic License Tool window

4.4.8 Starting WebSphere Voice Response

Once the WebSphere Voice Response product licenses are distributed, you can start WebSphere Voice Response, as follows:

1. Open a new window, log on as dtuser using the command **`su - dtuser`** and the following messages appear:

Example 4-2 command response

```
WebSphere Voice Response User Login
1) Start WebSphere Voice Response Processes
2) Do Not Start WebSphere Voice Response
Enter choice (or <ENTER> for option list)
```

2. Select **1** to start WebSphere Voice Response immediately. It will ask for the display to which WebSphere Voice Response will send the startup messages. That display will also be used to control and monitor WebSphere Voice Response. In our setup, the display is local to WebSphere Voice Response. The default value of :0 is used.

Note: If you want to administer the WebSphere Voice Response in another AIX server, you can use ipaddress:0 to export the console to the monitoring server. Before this, you must run **`xhost +`** to make available the console authority to other hosts.

3. Press Enter to confirm. A string of messages will be displayed in the window where WebSphere Voice Response is started. See Example 4-3.

Example 4-3 Initial WebSphere Voice Response startup messages

```
     ************************************

     WebSphere Voice Response Initialization

     ************************************

Cleaning up
Removing 3270 pooling memory segments.
lu_clean:  remove shared memory
lu_clean:  remove semaphore

Recovering Database files...
Performing Database Recovery...

Number of recoverDMS errors = 0

/usr/lpp/dirTalk/sw/bin/NOMEM &
```

4. WebSphere Voice Response will open another window to show its status. Logically, it is titled the WebSphere Voice Response Status window. The node name of the WebSphere Voice Response installation is also displayed on the title bar. In our installation the node name was localhost. See Example 4-4. The WebSphere Voice Response Status window, in a majority of the installations, is left opened while WebSphere Voice Response is running. This window is really the tail end of the commands from the WebSphere Voice Response status log.

Example 4-4 Tail end of commands from WebSphere Voice Response status log

```
NM Welcome to WebSphere Voice Response from Node Manager (NM)
NM Starting initialization procedure
Checking fileset consistency
*** WARNING ************************************************************
dirTalk.ISDN.DMS100 is at Level 2.3.0.5000 - the matching level is 2.3.0.5026
dirTalk.ISDN.com is at Level 2.3.0.5000 - the matching level is 2.3.0.5028
************************************************************************
Checking import consistency
ssistatus: Querying configuration ..
ssistatus: aixwvs3 is a Standalone system
fscheck 188: Filesystem Check ok
Running some tests on DTXA ) before we use it. (Takes approximately 1 minute)..
```

5. If you launch WebSphere Voice Response for the first time, you will see the Software License Agreement. Click **Accept**.

   After a few more minutes WebSphere Voice Response will open another window. This is the logon window, as shown in Figure 4-18.

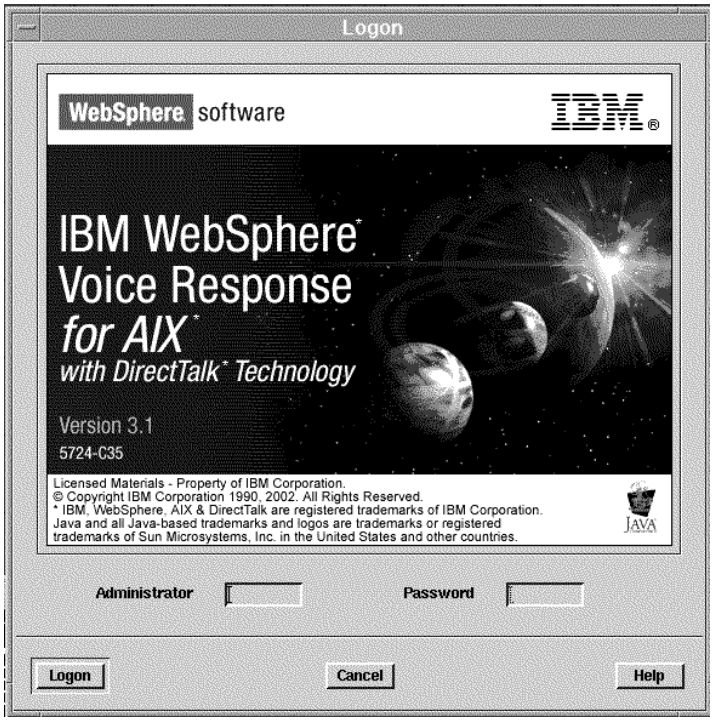


Figure 4-18 WebSphere Voice Response Logon window

6. Log on to the default administrator profile, named “admin”. Its password is also ”admin”. If the logon is successful, the Logon window is replaced with the Welcome window as shown in Figure 4-19. At this point, WebSphere Voice Response has been successfully installed and started. You should change the default administrator password for security reasons.

Figure 4-19 WebSphere Voice Response Welcome window

4.4.9 Configuration of packs

The next step is to configure the T1 trunk so that WebSphere Voice Response can receive calls from the PABX.

1. From the Welcome window, select **Configuration -> Pack Configuration -> Change** to display the Pack Configuration window as shown in Figure 4-20.

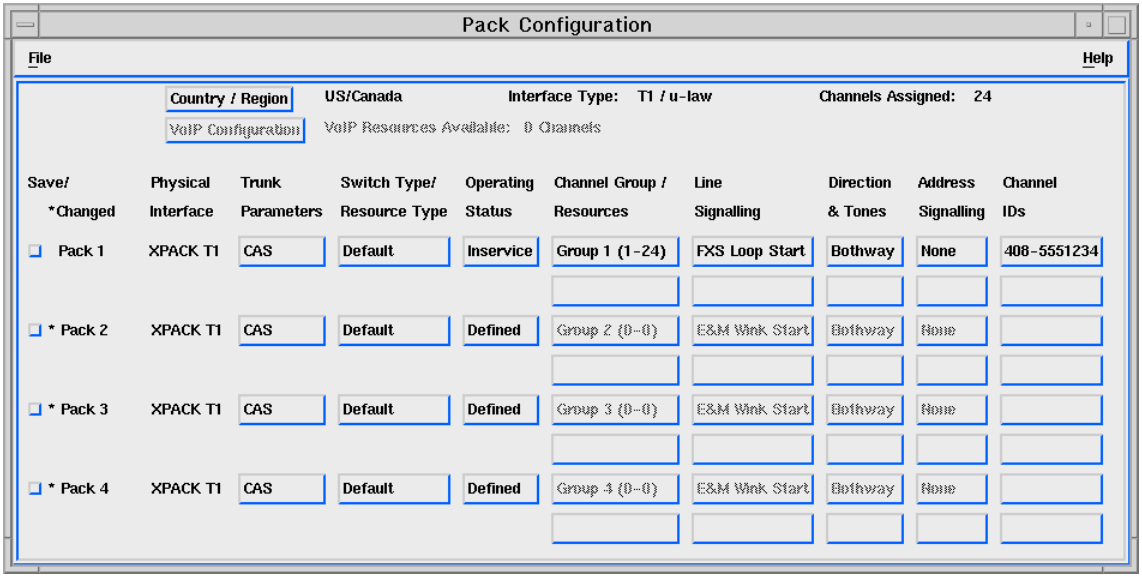


Figure 4-20 ITSO WebSphere Voice Response Pack Configuration window

Although the DTXA card can support up to four packs, only one is used in our installation.

The Country/Region column is set to US/Canada.

The Line Signalling column is set to FXS Loop Start, which is related to the PBX setting.

The Operating Status column is set to Inservice. This will allow the trunk to receive telephone calls immediately after a restart. This situation may not be suitable in your environment, because the back-end host may not be ready to handle incoming calls as yet after the WebSphere Voice Response restart.

2. To change the trunk interface parameter to CAS, click the **Trunk Parameter** button. The Trunk Interface Parameters window of Figure 4-21 on page 153 is displayed. This window also allows the number of channels assigned to be modified.

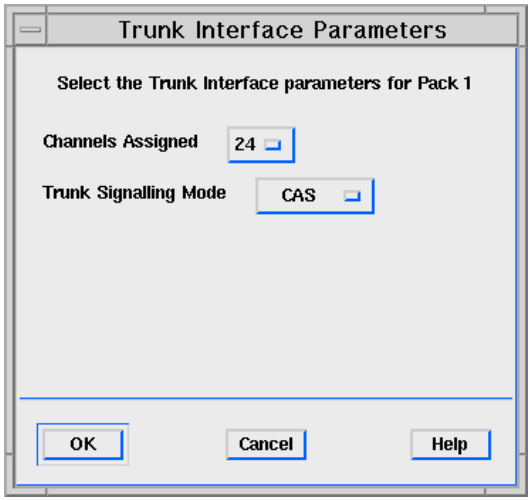


Figure 4-21 Trunk Interface Parameters window

3. Click **OK**. The following warning message shown in Figure 4-22 appears.

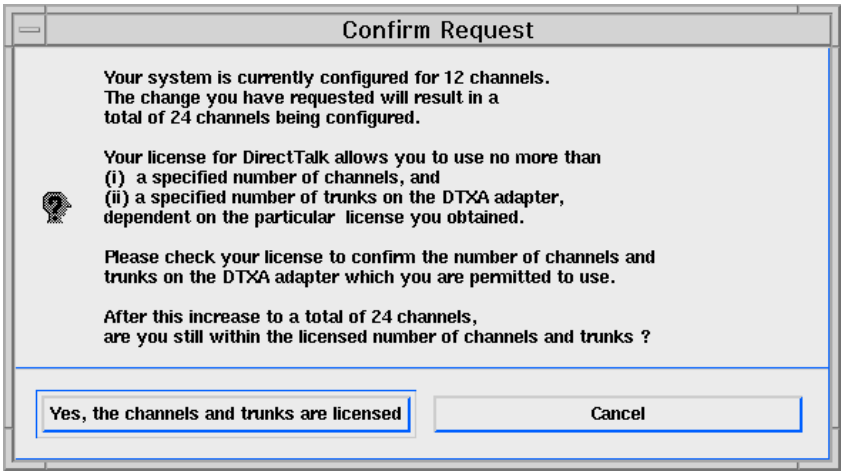


Figure 4-22 Changing the Channel Assigned field brings up this warning

4. Click **Yes, the channels and trunks are licensed**. As shown in Figure 4-23 on page 154, select **Channels groups** and increase the number of channels assigned from 12 to 24.

Figure 4-23 Channel Group window

5. Click **OK**. Click the **Channel IDs** field and you will see a window similar to Figure 4-24 on page 155.

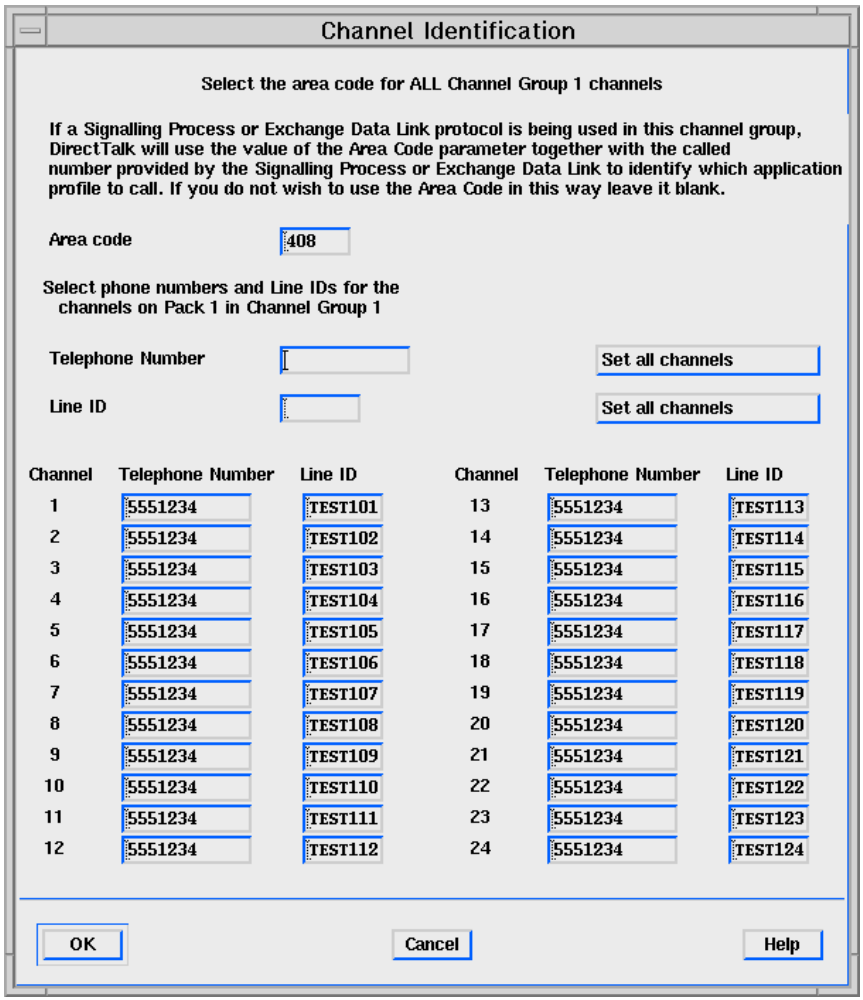


Figure 4-24 Channel Identification window

6. The default Channel Identification settings in Figure 4-24 were used in the ITSO setup. Click **OK**.

   The Channel IDs are left at their default values, although you may want to change them to suit your application environment. The Channel IDs and the application profile allow you to assign telephone channels to different voice applications.

   When all the changes to the Pack Configuration are completed, click the **Save/*Change** box next to Pack1. Then click **File** -> **Save**. The Confirm Request window of Figure 4-25 on page 156 appears.

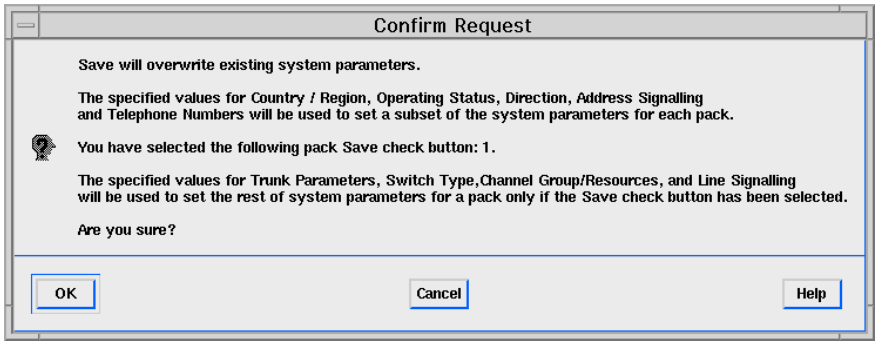


Figure 4-25 Confirmation of pack configuration changes

7. In most cases, changing the trunk configuration from the Pack Configuration window will enable the trunk to receive incoming calls from the PABX. However, in certain circumstances a more detailed configuration is desired and it is essential to change certain parameters associated with the trunk.
8. Finally, select **File** -> **Close**. You will see a window similar to Figure 4-26.

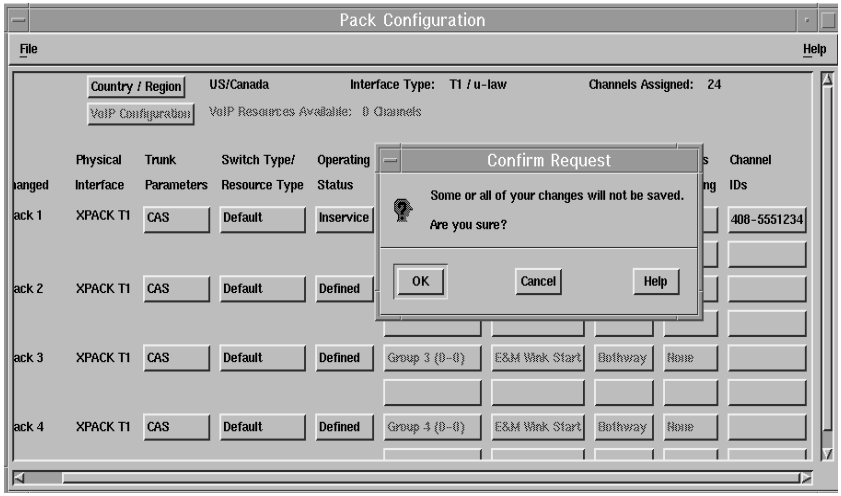


Figure 4-26 Confirm Request window

9. Select **OK** to exit the configuration.
10. From the Welcome window, click **Configuration -> System Configuration -> Change**. The System Configuration window is displayed (Figure 4-27 on page 157).

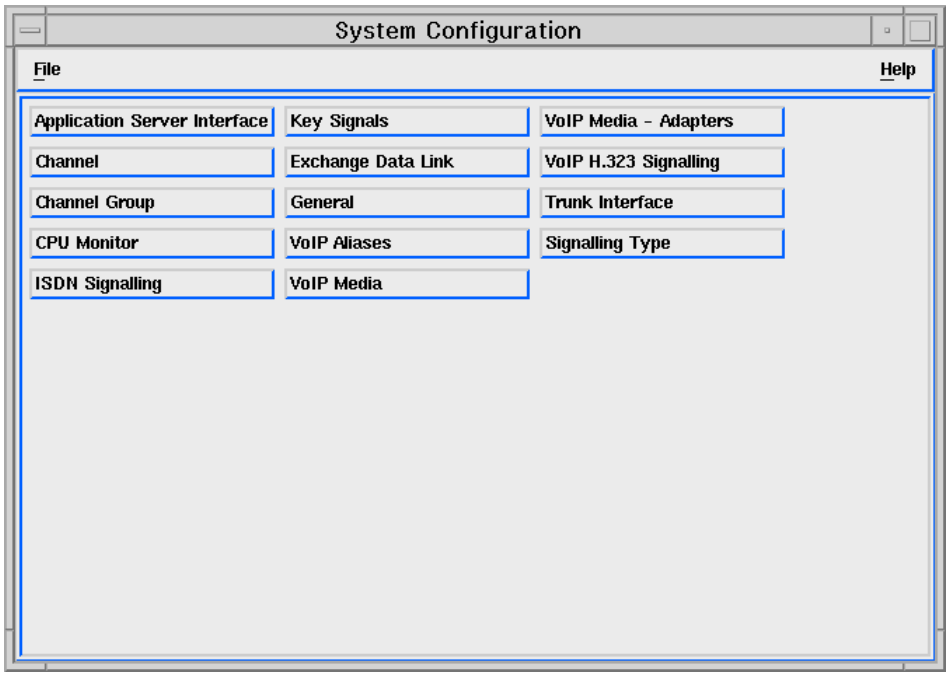


Figure 4-27 WebSphere Voice Response System Configuration window

11. In the configuration window, click **Trunk Interface**. The Trunk Interface window (Figure 4-28 on page 158) is displayed. The 16 buttons, labeled from 1 to 16, represent the maximum possible 16 trunks that can be supported by a RS/6000. This is equivalent to having four DTXA cards in one RS/6000 system. In a production environment, it is very unlikely that 16 trunks will be used on a single RS/6000.

    The other buttons on the Trunk Interface window represent the individual trunking parameters and protocol of each country that WebSphere Voice Response supports.

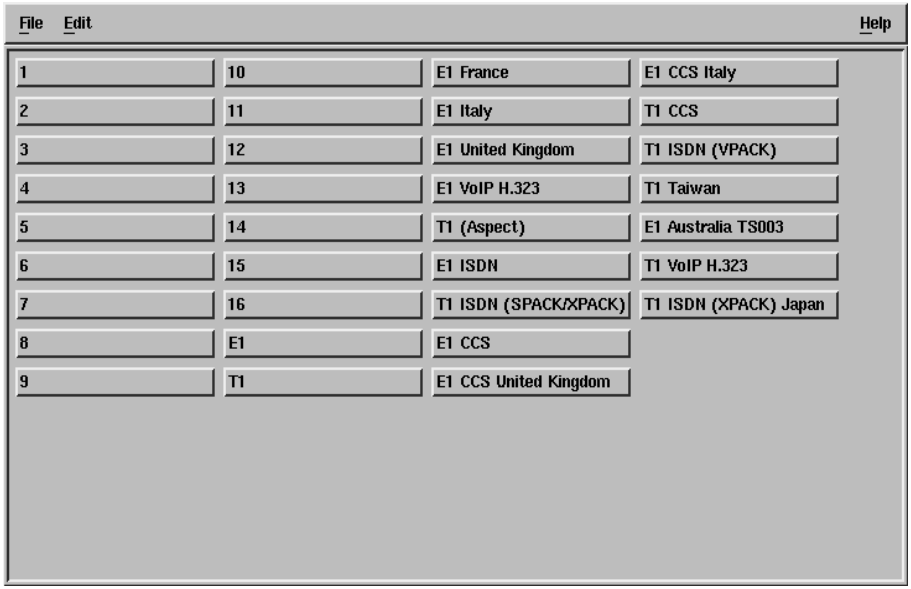


Figure 4-28 Trunk Interface window

12. Click **1** to view or change the parameter on the first trunk. The default values used in our implementation is shown in Figure 4-29 on page 159.

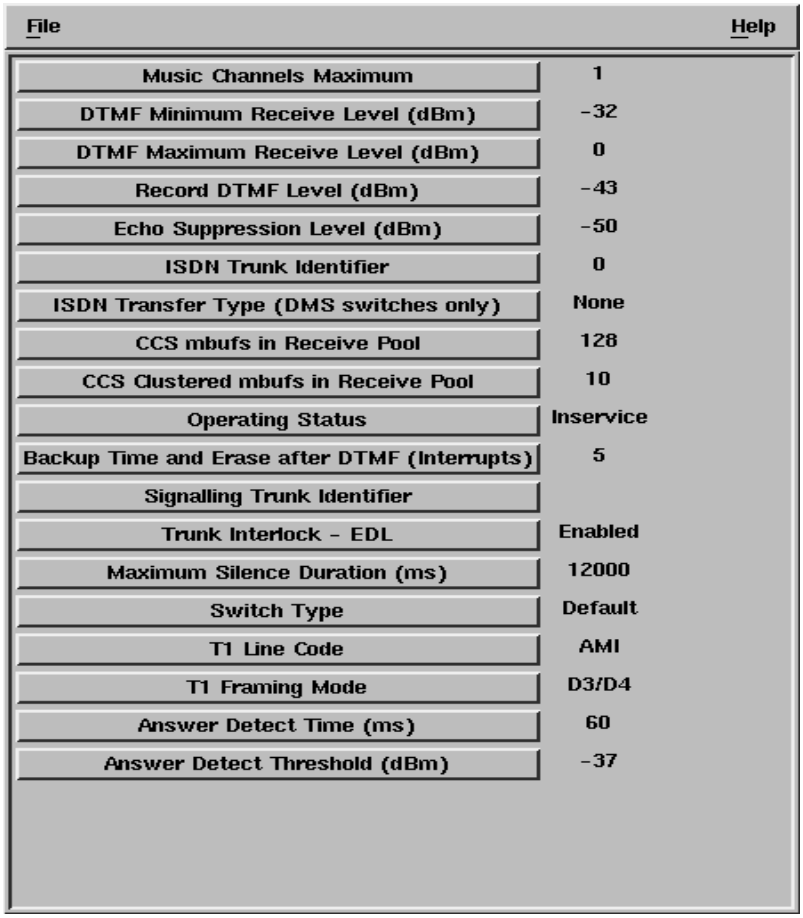


Figure 4-29 Trunk Interface/1 window showing the parameters settings used

13. Close all the configuration windows and stop WebSphere Voice Response. There are two ways to stop WebSphere Voice Response. In the Welcome window, click **Operations -> Immediate Shutdown** and **OK** to confirm. Observe the shutdown messages in the WebSphere Voice Response Status window of Example 4-4 on page 150. Wait for the Node Manager terminated message to appear. This signifies that WebSphere Voice Response has completely stopped. In the WebSphere Voice Response Status window of Example 4-4 on page 150, close the window by pressing Ctrl+C.

Note: It is recommended that you run **`DT_shutdown`** immediately after the shutdown. It will release some memory WebSphere Voice Response used.

Alternatively, WebSphere Voice Response can immediately be stopped by the **DT_shutdown** command issued from dtuser. This method of shutdown will also close the WebSphere Voice Response Status window.

4.5 Testing WebSphere Voice Response installation

To test the WebSphere Voice Response configuration, log on to WebSphere Voice Response as dtuser from the command line. Then restart WebSphere Voice Response. Wait for the Logon window of Figure 4-18 on page 151 to appear. At this point in the start process, the DTXA card is almost at its final stage of the initialization process. Log on as the administrator to bring up the Welcome window.

1. From the Welcome window, select **Operations -> System Monitor** to bring up the System Monitor window as shown in Figure 4-30. Because the Operating Status of the Pack Configuration is set to Inservice (Figure 4-20 on page 152), all 24 channels will be automatically put into the Inservice status. With this status, WebSphere Voice Response is ready to answer incoming calls.

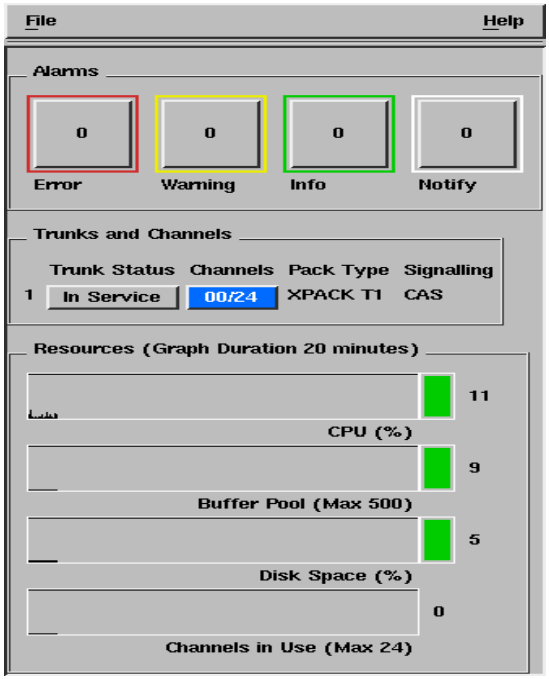


Figure 4-30 WebSphere Voice Response System Monitor window

2. Select the channels (00/24) button. The window shown in Figure 4-31 is displayed. From this window, we can enable or disable any channel and also monitor the status of these channels.

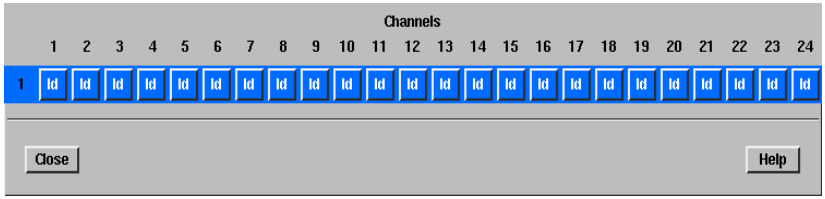


Figure 4-31 Channel Status window

3. We call 20011, channel 1 of the T1 trunk (it depends on the configuration of PBX). Since no other WebSphere Voice Response applications are imported, we heard the default Java sample menu application, which plays the following message:

Example 4-5 application message

```
"Hello. Welcome to the menu demonstration application.
Press 1 for the time.
2 for the date.
3 for the date and time.
Or press 0 to exit."
```

It means that the Java beans and VXML support has been installed. Otherwise, only the first sentence will be available without any of the choices of data and time.

The status of the first channel changes to active (Figure 4-32).

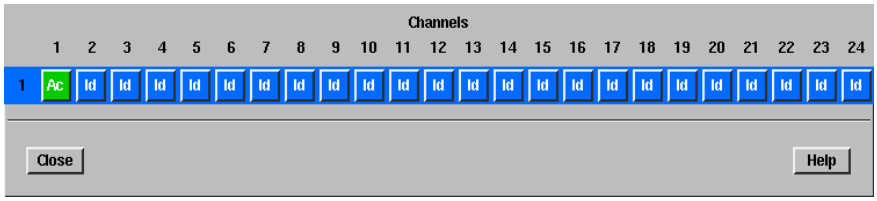


Figure 4-32 Channel Status window

The configuration completed so far has enabled WebSphere Voice Response to receive incoming calls and play the sample menu application. No other applications have been imported or installed yet. The following sections describe the installation of the speech recognition and text-to-speech technologies components.

4.6 Installation of WebSphere Voice Server

The server software and client software can be installed on the same or different RS/6000 systems. However, this is dependent on the capacity of the RS/6000 and whether it will meet your business objectives. In our implementation of WebSphere Voice Server, we installed both server and client software on the RS/6000.

Generally, the WebSphere Voice Server server software is installed first and then the client software. WebSphere Voice Response for AIX is the prerequisite for WebSphere Voice Server client. It can be independently installed up to a point where WebSphere Voice Server client software is required. You can continue to install WebSphere Voice Server client software on the WebSphere Voice Response RS/6000 system or install the WebSphere Voice Server server software. We installed the server software and then the client software on one RS/6000 system.

4.6.1 Installation of WebSphere Voice Server server

This section describes how we installed the WebSphere Voice Server server component onto the RS/6000 machine.

1. Iinstall the C++ set 5.0.2.0 using **smitty install_latest**.
2. Next, we applied the AIX Application Runtime PTF U480006 for C++ set using **`smitty update_all`** (for details on usage, refer to step 1 on page 134). Use the command **`lslpp -L`** to confirm the level of C++ set:
   - xlC.rte 5.0.2.1
   - xlC.aix43.rte 5.0.2.3

The WebSphere Voice Server server software is packaged on one CD. This contains the associated speech recognition and formant text-to-speech engine for a specific language. WebSphere Voice Server supports two type of text-to-speech formats: formant and concatenative.

The concatenative text-to-speech is packaged on one CD. Only one type of text-to-speech can be installed on one RS/6000 server. We installed the US English speech recognition and formant text-to-speech on one RS/6000 machine.

Before proceeding with the installation of WebSphere Voice Server software, the maintenance level 8 fixes to AIX 4.3.3 must be applied. After this is done, we used **smitty install_latest** to install the server software (for details on usage, please refer to step1 on page 134). Select all the filesets to install.

4.6.2 Grant an administrator to WebSphere Voice Server server

During the installation process, a user group is created. The user name and group name are both known as vvt. Any other user who wishes to use the WebSphere Voice Server facilities should belong to the vvt group. We added the dtuser to the vvt group using **smit** (for details on usage, please refer to step 2 on page 133) as shown in Figure 4-33.

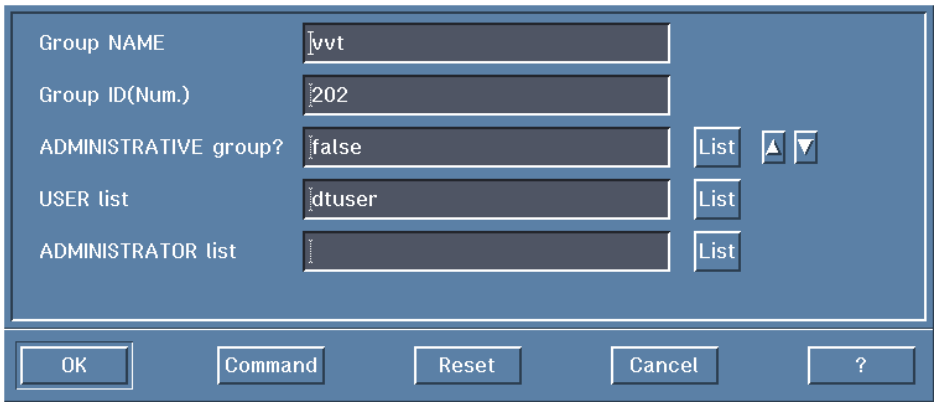


Figure 4-33 Add Users window

We use the same role to maintain both WebSphere Voice Response and WebSphere Voice Server.

Finally, we log on as dtuser and edit $VAE/sw/environ/.vaeprofile.user. Add the following statement:

```
. /var/vvt/bin/vvt.setenv
```

The statement is added just before the End User-Defined section. This will prevent an error message that reports that the vvt environment is not set when WebSphere Voice Response is started the first time.

Then log on again as dtuser to active the modification.

4.6.3 Configuring the WebSphere Voice Server server

The number of speech recognition engines or formant text-to-speech engines to be supported on the server must be configured. First, we configure one US English speech recognition engine and one US English formant text-to-speech engine using the vvt_config script provided with the product. You must ensure that your RS/6000 system is capable of supporting the number of engines you are going to configure. Configuring with too many engines will produce unpredictable results.

Figure 4-34 shows the responses to the questions asked by the vvt_config script.

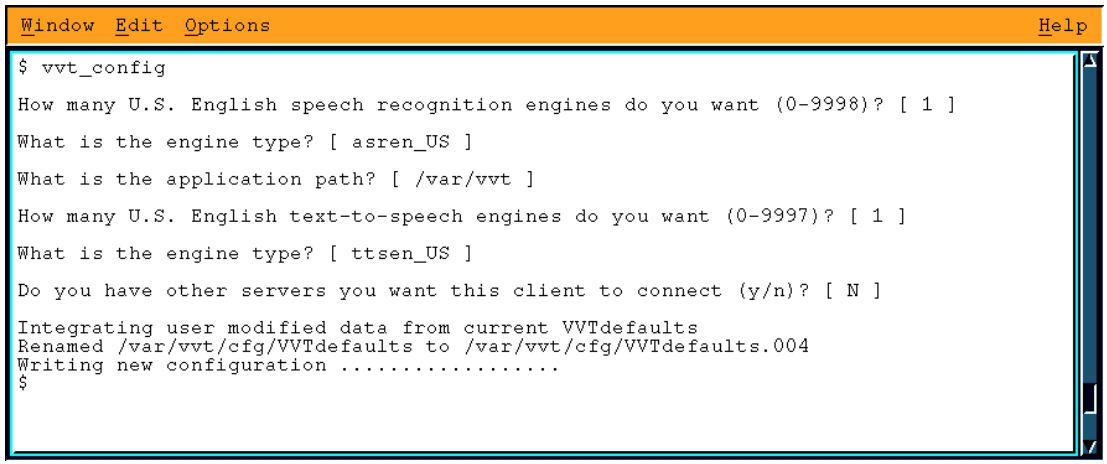


Figure 4-34 Configuration of a speech recognition and text-to-speech engine to support US English

The vvt_config script generates the parameter values for the VVTdefaults configuration file. This file is located in the /var/vvt/cfg directory. Every time the vvt_script is run, the previous version of the VVTdefaults file is saved. The file name of the saved version is appended with a suffix that increments by 1. Figure 4-34 shows the original version of the VVTdefaults is saved as VVTdefaults.004.

The listing of VVTdefaults generated during the implementation in the ITSO is shown in Figure 4-35 on page 165. A detailed description of each parameter is found in the *WebSphere Voice Server Version 3.1 - Use with WebSphere Voice Response for AIX Telephony Platform Administrator's Guide*, G210-1259.

```
# DO NOT MODIFY THIS LINE - THE CONFIGURATION UTILITY DEPENDS ON IT - BEGIN
# created on 2002/10/07 16:49:45 by dtuser
# $Id: vvtconfig.tcl,v 1.15.2.7.2.6 2002/08/20 17:22:38 reichel Exp $
#
# EVERY MODIFICATION BETWEEN THE TWO 'DO NOT MODIFY' LINES WILL BE LOST
# DURING RECONFIGURATION
#
Pm.useRouter: 1
asrp.0.Asr.engineType: asren_US
asrp.0.Asr.userId: vsen_us
asrp.0.Asr.enrollId: vse1
asrp.0.Asr.taskId: vebus
asrp.0.Asr.appPath: /var/vvt
asrp.0.Tts.languages: EN-US
asrp.0.Asr.baseformsSource: vv
Pm.numberOfAsrServers: 1
*.*.Asr.notifyWhenLoaded: 1
ttsp.0.Tts.engineType: ttsen_US
ttsp.0.Tts.languages: EN-US
Pm.numberOfTtsServers: 1
# DO NOT MODIFY THIS LINE - THE CONFIGURATION UTILITY DEPENDS ON IT - END
```

Figure 4-35 Contents of VVTdefaults

To put the engines into service, use the **pm** command to start Process Manager. It is not necessary to start the service at this point, since the client software has not been installed yet. But using this command, we can check the installation of WebSphere Voice Server server.

4.6.4 Starting server components

In our implementation where the client and server are both installed on the same RS/6000 system, the server can be started with the dtuser login.

1. Log in as dtuser. To start the server use the **pm** command. The response indicates that the command has successfully completed:

   ```
   Resource file  : /var/vvt/cfg/VVTdefaults
   pm started
   ```

   Since Process Manager is started without any command, it is started as a daemon. To verify that Process Manager has started the speech recognition and text-to-speech engines, we entered the **tsmcon** command. The output of the **tsmcon** command is shown in Figure 4-36 on page 166. It shows that the speech recognition engine, asren_US, and the text-to-speech engine are free and both are available to be used by any application.

```
Window  Edit  Options                                                    Help
/var/vvt/cfg
$ cd ..
$ ls
Version   bin       inc       log       tcl       tts
ViaVoice  cfg       lib       pcm       tmp
$ cd cfg
$ ls
VVTdefaults      VVTdefaults.001  VVTdefaults.002  VVTdefaults.003  VVTdefaults.004
$ pm
Resource file: /var/vvt/cfg/VVTdefaults
pm started
$ tsmcon
 Host          | Server.cloneId | Type     | Status              | Count  | Last Alloc
ated
================================================================================================
=====
 9.27.111.193 | asrp     .0000 | asren_US | available           |      0 | <not set>
 9.27.111.193 | ttsp     .0000 | ttsen_US | available           |      0 | <not set>
================================================================================================
=====
```

Figure 4-36 Output of tsmcon showing two engines are available to be used

2. To further verify that Process Manager has started the other relevant processes, we enter the **`pm list`** command. Its output is shown in Figure 4-37.

```
Window  Edit  Options                                                    Help
Resource file: /var/vvt/cfg/VVTdefaults
pm started
$ op^H^H
ksh: op^H^H:  not found.
$ pm list
Sending command "list" to localhost
  PID  |      PROCESS           | STATE         | RESTARTS | LAST STARTED
  18020| logger           .0000 | RUNNING       |      0   | Tue Oct  8 08:50:1
3 2002
  21592| tsmr             .0000 | RUNNING       |      0   | Tue Oct  8 08:50:1
4 2002
  19518| asrp             .0000 | RUNNING       |      0   | Tue Oct  8 08:50:1
4 2002
  14654| ttsp             .0000 | RUNNING       |      0   | Tue Oct  8 08:50:1
5 2002

Done
$
```

Figure 4-37 Output from pm list command showing the active processes

3. The server components have started successfully. The next step is to start the client components.

   If changes are made to the VVTdefaults file, you may stop and start the processes to update the changes. To stop the processes immediately, use the **`pm exit`** command. The messages will be displayed as shown in Figure 4-38 on page 167.

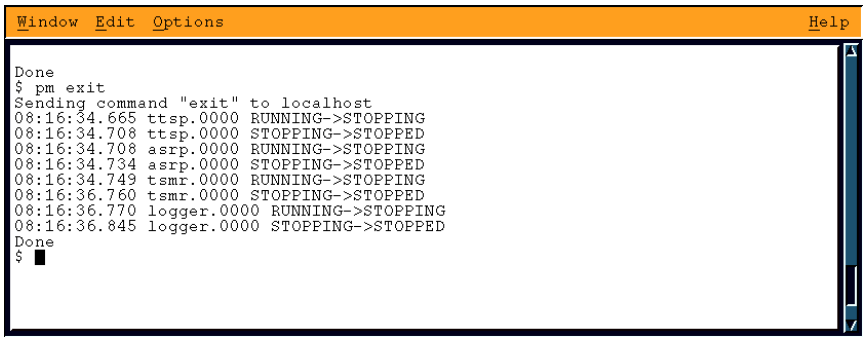


Figure 4-38 Stop Services window

The installation process for concatenative text-to-speech is similar to the method used to install formant text-to-speech. Concatenative text-to-speech is packaged onto one CD for each language.

4.6.5 Installation of the WebSphere Voice Server client

Since base components of DirectTalk for AIX are installed and running, we need to shut down WebSphere Voice Response in order to install the WebSphere Voice Server client software.

1. Log on as dtuser and type **`DT_shutdown`** and wait for the shutdown to complete.
2. Log on as root and use **`smitty install_latest`** to install the WebSphere Voice Server client software. Select all the filesets to install.

4.6.6 Setting environment

Next, set the WebSphere Voice Response environment and ownership of the filesets just installed. Perform the following instructions:

1. To set the environment, enter:

   ```
   . /usr/lpp/dirTalk/tools/vae.setenv
   ```

2. To confirm that the WebSphere Voice Response environment is correctly set, enter:

   ```
   echo $VAE
   ```

3. The response should be:

   /usr/lpp/dirTalk

4. Enter the following command to refresh the file ownership:

   `/usr/lpp/dirTalk/tools/vae.setuser`

5. Type in `dtuser` when prompted to enter the user ID. A list of messages similar to Figure 4-6 on page 138 will be displayed.

6. When the **`vae.setuser`** command is successfully completed, Log in as dtuser and start WebSphere Voice Response from another window.

7. Select option **1** to start WebSphere Voice Response processes. Wait for the WebSphere Voice Response Logon window to appear. Log on as admin. The Welcome window will appear.

4.6.7 Importing WebSphere Voice Server filesets

The next step is to import WebSphere Voice Server filesets on the client system. The filesets are contained in a standard WebSphere Voice Response file.

1. In the Welcome window, select **Applications -> Applications**.

2. In the Applications window, select **Applications -> Import -> Replace -> File**.

3. In the Search String field, type:

   `/usr/lpp/dirTalk/sw/viavoice/*`

4. Click **Search** button.

5. Highlight the file **ViaVoice.imp** and click **OK** button to import that file. See Figure 4-39 for details. ViaVoice.imp contains a custom server for WebSphere Voice Response on a WebSphere Voice Server client system.

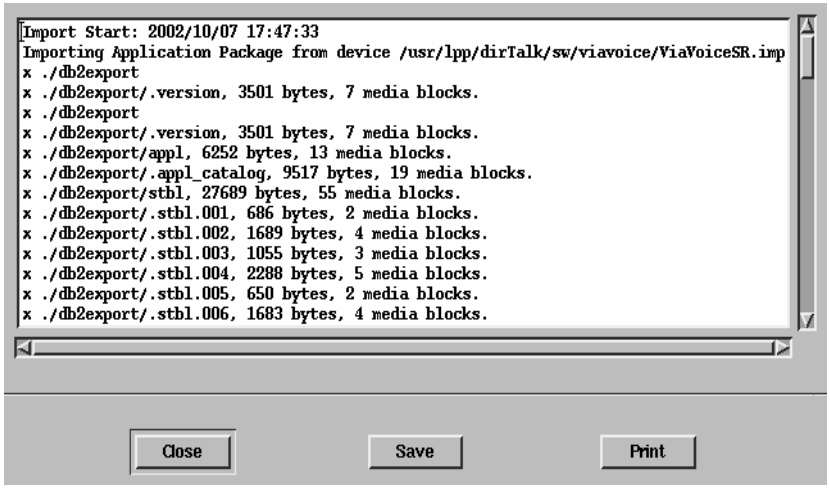


Figure 4-39 Importing WebSphere Voice Server custom servers

6. A window showing the status of the import function is displayed. When the import has completed, check for any error messages. Save the import report into a file in case you find any errors during the import process and you want to diagnose the cause of the errors. If the import is successful as shown in Figure 4-40, close the window.

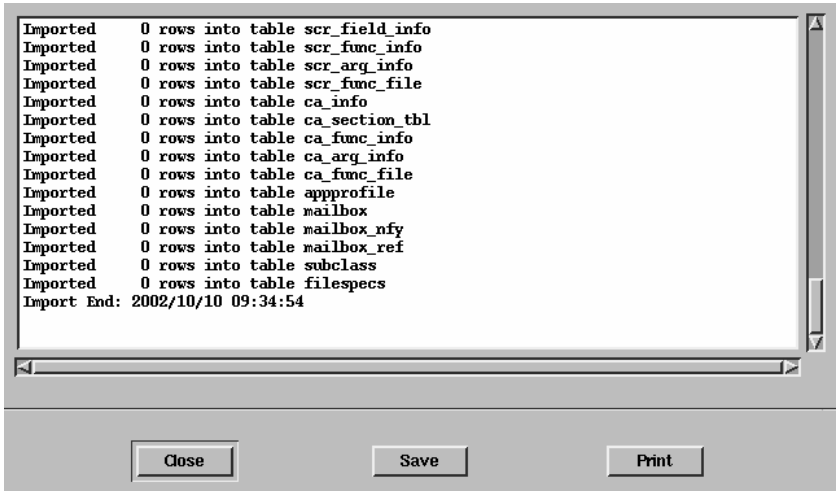


Figure 4-40 Import Configuration window

7. Repeat the import process for ViaVoiceSR.imp and ViaVoiceTTS.imp. ViaVoiceSR.imp contains the filesets for speech recognition. ViaVoiceTTS.imp contains the filesets for text-to-speech. When all the files have been imported, click **Cancel** to close the window.
8. To confirm that the filesets are correctly imported, select **Applications** from the Welcome window. A window similar to Figure 4-41 on page 170 is displayed. Confirm the three imported applications, ViaVoice, ViaVoiceSR and ViaVoiceTTS, are displayed.

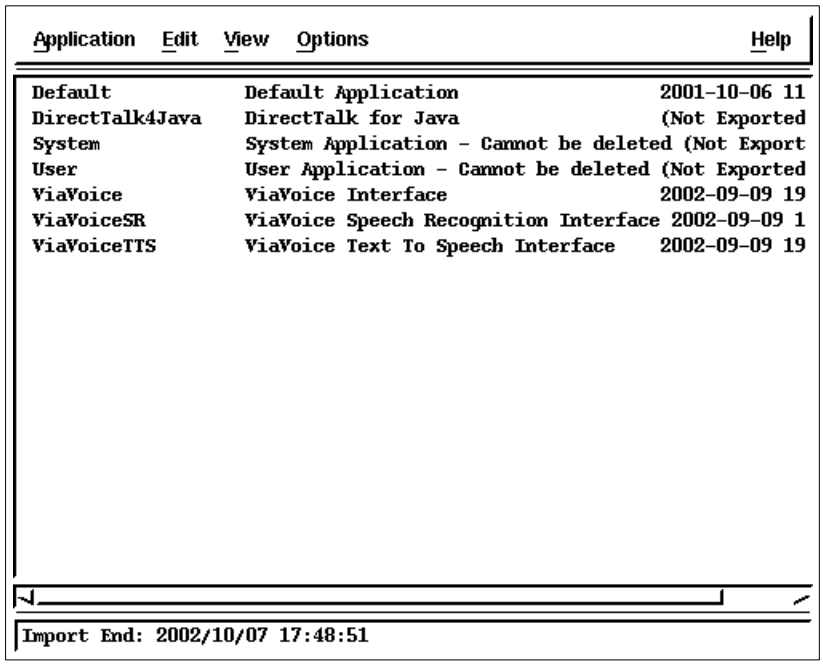


Figure 4-41 WebSphere Voice Response Applications window showing the imported files

9. To view the contents of each application, highlight the application and double-click. Figure 4-42 shows the contents of the imported ViaVoice application. The only component the application contains is the ViaVoice custom server.

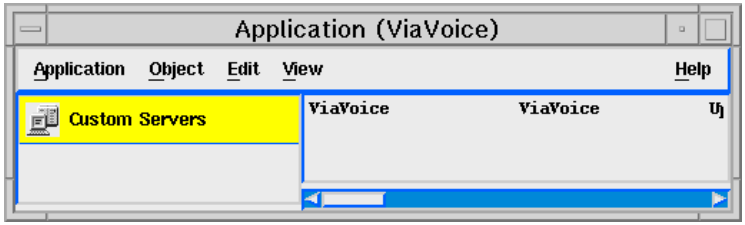


Figure 4-42 Contents of ViaVoice applications

10. Figure 4-43 on page 171 shows the contents of the imported ViaVoiceSR speech recognition application. The application provides a set of state tables that can be used for the speech recognition application.

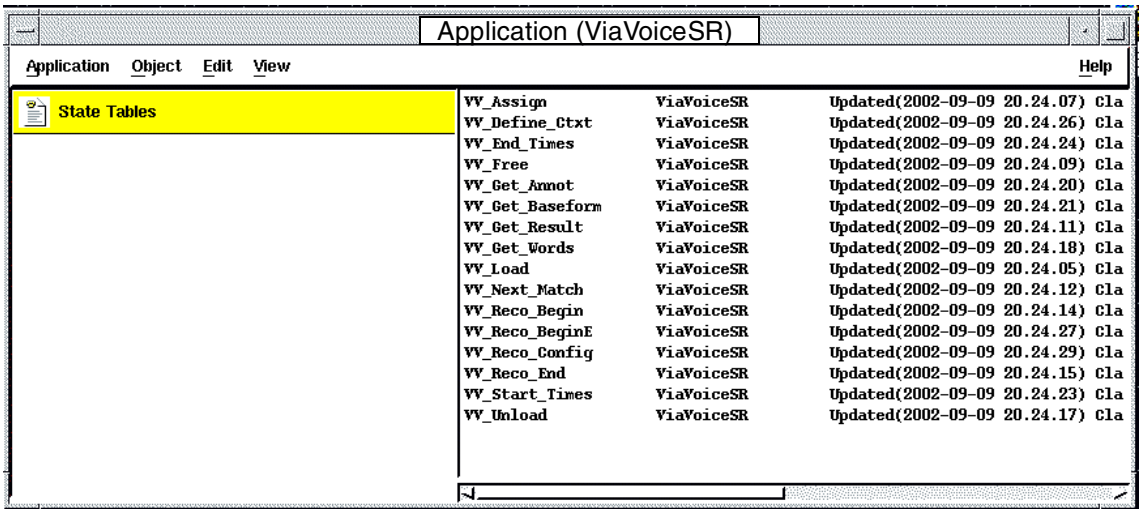


Figure 4-43 Contents of ViaVoiceSR application

11. Figure 4-44 shows the contents of the ViaVoiceTTS application. The set of state tables provides the interface to text-to-speech technologies.

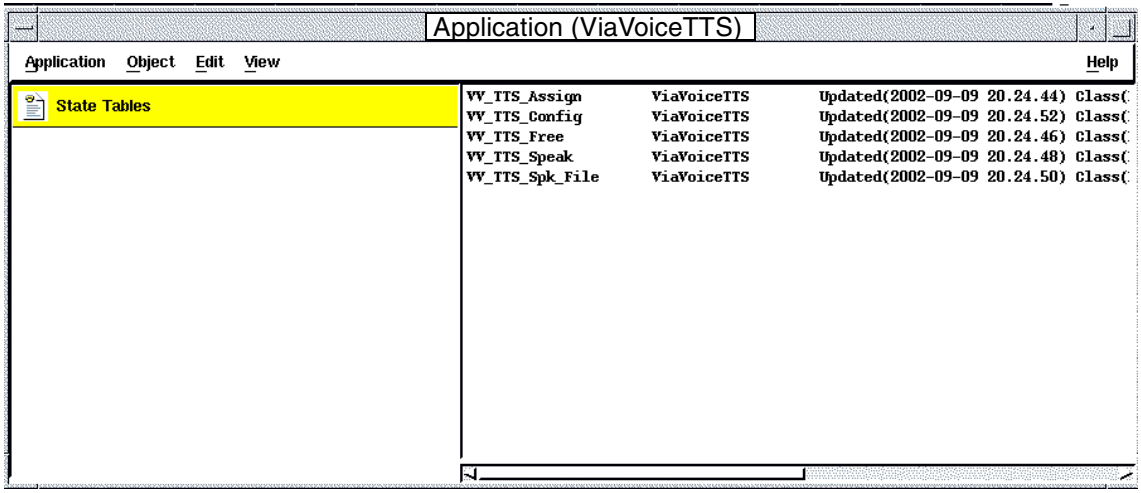


Figure 4-44 Contents of ViaVoiceTTS application

4.6.8 Importing Java components

The Java components for WebSphere Voice Server are included in the WebSphere Voice Response installation. To run Java voice applications with WebSphere Voice Server, the Java plug-in must be installed and the Java beans environment must be configured.

1. To install the Java plug-in, log on as dtuser and change directory to:

   /usr/lpp/dirTalk/DTBE/plugins

2. Type the following command:

   ```
   dtjstop
   dtjshost -exit
   dtjplgin dtjvv.zip
   ```

3. Shut down WebSphere Voice Response with the **DT_shutdown** command. When WebSphere Voice Response has completely shut down, restart WebSphere Voice Response using **vaeinit**. The speech recognition plug-in and text-to-speech plug-in are ready to be used.

4.6.9 Starting WebSphere Voice Server client

Perform the following steps to start the client:

1. Extra information is presented in the WebSphere Voice Response login window. The command:

   ```
   . /var/vvt/bin/vvt.setenv
   ```

   inserted earlier in the .profile file has set WebSphere Voice Response to operate in the dtuser environment. This is confirmed in the startup messages displayed in Figure 4-45 on page 173.

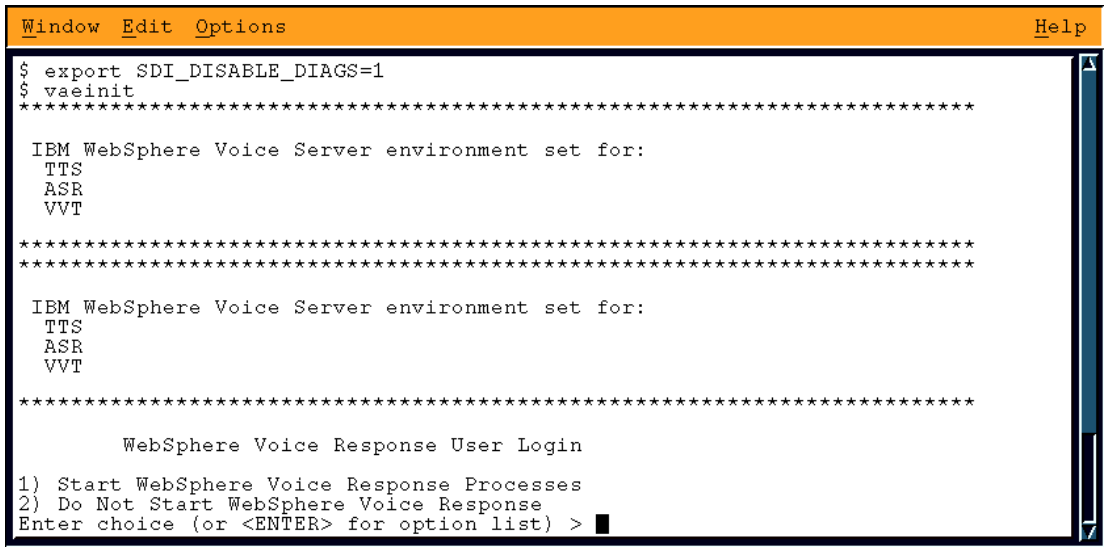


Figure 4-45 WebSphere Voice Response environment set for WebSphere Voice Server

2. When WebSphere Voice Response has completed its startup process, log on as Admin. In the Welcome window, click **Operations -> Custom Server Manager**.
3. Set the IPL status of the ViaVoice custom servers to Auto - Start On. The custom servers will automatically start every time WebSphere Voice Response restarts.
4. In the Run Status window, start the ViaVoice custom servers:

   Click **Run Status of ViaVoice** and set to Start.
5. The status will be displayed as WAITING when the custom servers have started. Figure 4-46 on page 174 shows both custom servers have been started and ready to be used.

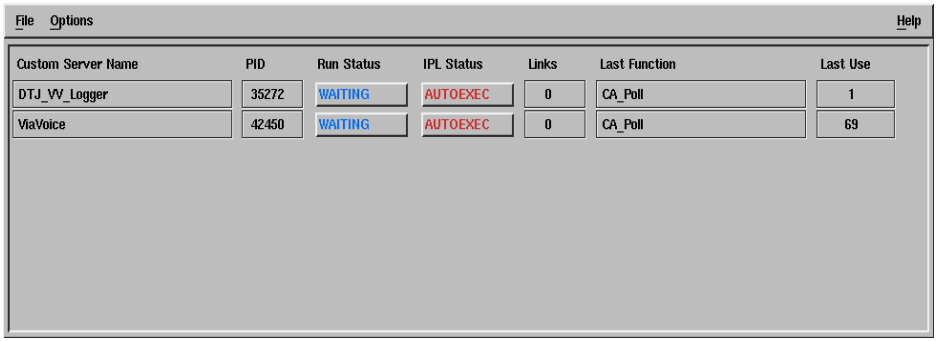


Figure 4-46 Custom servers DTJ_VV_Logger and ViaVoice have started and waiting to be used

6. To use speech recognition and text-to-speech, we make changes to the configuration file, default.cff, in the /var/dirTalk/DTBE/native/aix/ directory.

   Backup the default.cff file and overwrite it using default.sample.cff. Then comment out all the French configuration in the group definition at the end of the file:

   – # TTSService=WVS_TTSfr_FR
   – # TTSDefinition=fr_FR,TTSfr_FR
   – # RecoService=WVS_Recofr_FR
   – # RecoDefinition=fr_FR,Recofr_FR

   The default.cff file that was used in the ITSO is include in Appendix D, “Default.cff with English and German support” on page 553.

7. We configured the Weather.VXML sample application as the default application. The path name of the Weather.VXML application is:

   /var/dirTalk/DTBE/samples/Weather.VXML

   This path is a link to the actual file, which is located in the /usr/lpp/dirTalk/DTBE/samples/Weather.VXML directory.

   This application allows us to test the US English speech recognition and formant text-to-speech engines.

   There are a number of parameters in the default.cff file that can be modified to work within your installation environment. Since the WebSphere Voice Server server and client software are installed on the same machine, we set the Hostname definition to LocalHost.

8. To make the changes take effect, we run the **`dtjconf`** command. We must first stop the node by typing in the command **`dtjstop`**. Then, restart the node using **`dtjstart`**. The result is shown in Figure 5-48.

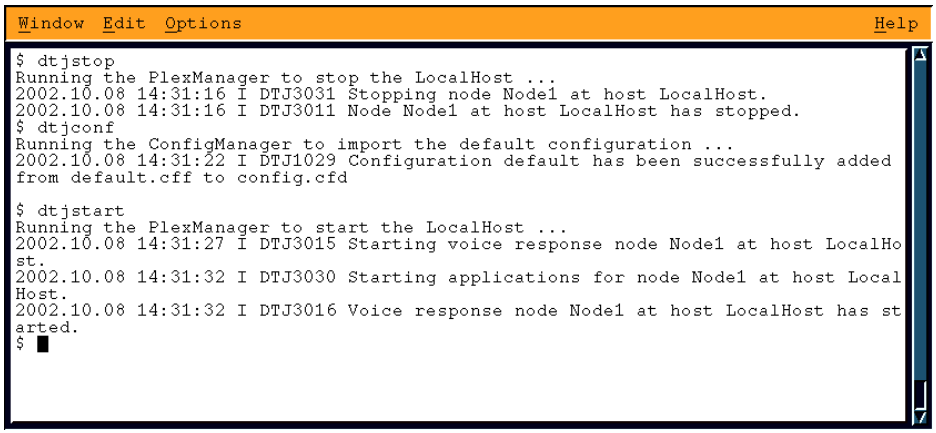


Figure 4-47 Import configuration window

9. Start WebSphere Voice Response, if it has not been started. The WebSphere Voice Response Status window will display more information than the first time WebSphere Voice Response was started. For instance the custom servers, DTJ_VV_Logger and ViaVoice, will be started. This log is part of the file named DTstatus.out, located in the /home/WebSphere Voice Response/current_dir/oamlog. The WebSphere Voice Response Status window displayed is the tail function of the file.
10. When WebSphere Voice Response is fully started, start the Java environment by entering **dtjstart**.
11. If any changes are made to default.cff, run the **dtjconf** command. Stop the node with the **dtjstop** command and then restart the node with the **dtjstart** command.
12. The version of WebSphere Voice Response Beans Environment can be displayed with the **dtjver** command. The version that was used in the ITSO implementation is shown in Figure 4-48.

```
The current version of WebSphere Voice Response Beans Environment is...
3.1 02/09/12 07:46:36
```

Figure 4-48 Output of dtjver command

4.7 Testing our implementation

To test the implementation, we used extension 24141 to call extension 22001, the first channel of the T1 trunk. The call was answered by the Weather.VXML application.

1. Start System Monitor of WebSphere Voice Response (refer to 4.5, “Testing WebSphere Voice Response installation” on page 160).
2. Start WebSphere Voice Server (refer to 4.6.4, “Starting server components” on page 165).
3. When prompted to select a language, we selected US English, then tested the speech recognition and text-to-speech functions. We used the WebSphere Voice Response System Monitor to track the telephone call.
4. We used the **tsmcon** command to track the usage of the speech recognition and text-to-speech engines. The output of the **tsmcon** command is shown in Figure 4-49.

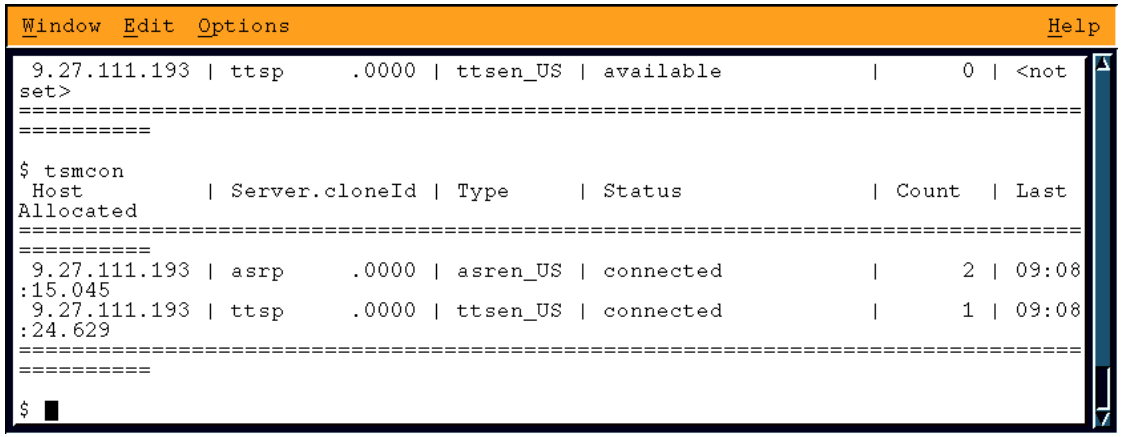


Figure 4-49 tsmcon commands to track the status of speech recognition and TTS engines

4.8 Call transfer application

We created another VoiceXML application to test call transfer functionality. This required the application to use the VoiceXML Transfer tags. The call transfer supported with this version is unsupervised. At the completion of the transfer, the T1 channel that connects the incoming call to WebSphere Voice Response is released and is available to receive the next call.

1. The VoiceXML application needed to be written. Example 4-6 on page 177 displays the call transfer application we used.

Example 4-6 VoiceXML transfer application

```
<VXML version=”1.0”>
   <form>
       <block> Please hold while your call is being transferred. </block>
     <transfer dest=”24151”/>
   </form>
</VXML>
```

2. We used WebSphere Voice Toolkit 3.1 to create the file and save the source code as Transfer.VXML at the /usr/lpp/dirTalk/DTBE/samples/.
3. Edit /var/dirTalk/DTBE/native/aix/default.cff and add the following content after the Application menu’s definition:

   ```
   AppName=Transfer
   Enabled=yes
   Parameter=URI,file:///usr/lpp/dirTalk/DTBE/samples/transfer.VXML
   AppClass=com.ibm.speech.VXML.DTViocelet
   ;
   ```

4. Change `NodeDefAppName=weather` to `NodeDefAppName=Transfer`.
5. Save the file and import configuration.
   a. Run the **`dtjconf`** command.
   b. Stop the node with the **`dtjstop`** command.
   c. Restart the node with the **`dtjstart`** command.
6. Test the application. We called extension 22001, the first channel of the T1 trunk. The Transfer.VXML application plays the synthesized speech:

   ```
   Please hold while your call is being transferred.
   ```

7. The call is transferred to extension 24151 and the T1 channel is released.

4.9 Add new languages

So far we have only installed languages on a new system. To install either single-byte or double-byte character languages onto an existing system is different. The languages supported by WebSphere Voice Response for AIX 3.1 are:

Single-byte languages include:

- Brazilian Portuguese
- Canadian French
- French
- German
- Italian

- Spanish (Castillian)
- UK English
- US English

Double-byte languages include:

- Japanese
- Simplified Chinese

4.9.1 Add single-byte languages

We used the German version of WebSphere Voice Server, to demonstrate how to add a single byte language to an existing system.

The German version of the Voice Server server was implemented onto the same machine running the English version. When anybody dials this system, they will now be able to select either the English or German language for their speech recognition or text-to-speech function.

We used the following steps to install the German language version:

1. Log on as dtuser and run **`pm exit`** to stop all of the running service for the Voice Server server.
2. Log on as root and install all of the language support filesets (both ASR and TTS) for German using **`smitty install_latest`**. The components lists as below:
   - viavoice.asr.de_DE
   - viavoice.asr_sample.de_DE
   - viavoice.asr_server.de_DE
   - viavoice.tts.de_DE.1at08KHz
   - viavoice.tts.de_DE.2at08KHz
   - viavoice.tts.de_DE.rte

   Wait for the end of the installation.
3. Run **`vvt_config`** to config WebSphere Voice Server to use both German and English voice components (see Figure 4-50 on page 179):

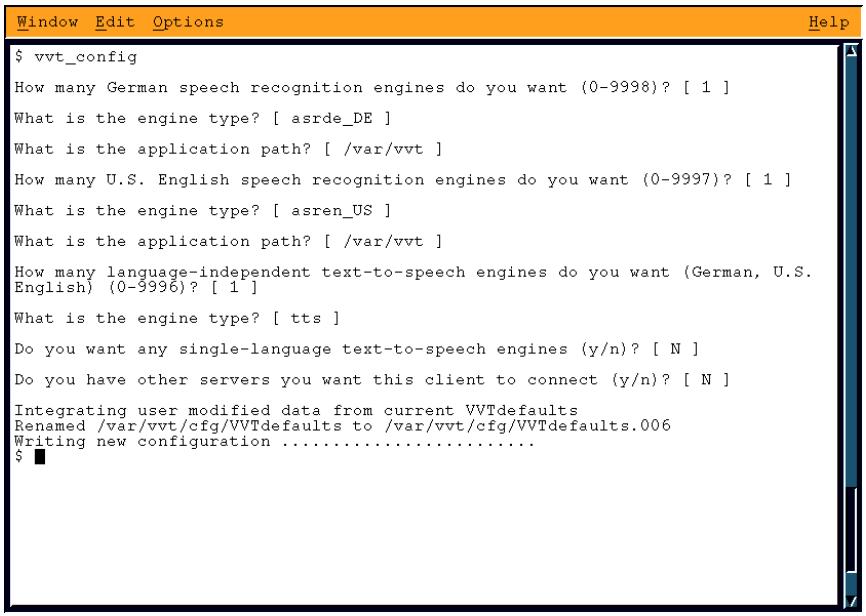


Figure 4-50 Config window of vvt_config

4. Run the service of the Voice Server server using the **pm** command. Then monitor the status of these services using **tsmcon** (see Figure 4-51).

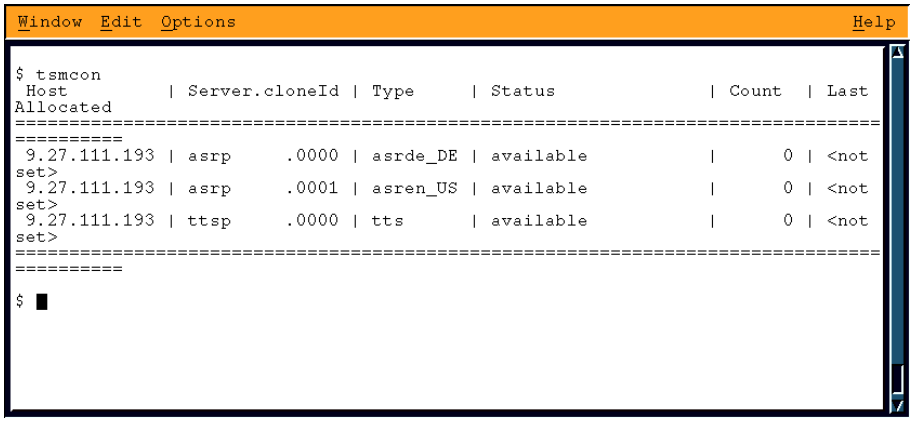


Figure 4-51 Service Status window

There are two ASR services launched. One is for English, another is for German. However, there is only one TTS service in the server side. All the text-to-speech engines of support languages are migrated to one service in WebSphere Voice Server 3.1. Its type is tts.

5. Stop the running node1 with the command **dtjstop**.
6. Modify /var/dirTalk/DTBE/native/aix/default.cff. (See Appendix D, “Default.cff with English and German support” on page 553.)
   - Uncomment:
     - #TTSService=WVS_TTSde_DE
     - #RecoService=WVS_Recode_DE
     - #RecoDefinition=de_DE,Recode_DE
   - Change:
     - TTSType=TTSde_DE
     - InitSessionString=EngineName=ttsde_DE,EngineFree=true
   - To:
     - TTSType=TTS
     - InitSessionString=EngineName=tts,EngineFree=true
   - Change:
     - TTSType=TTSen_US
     - InitSessionString=EngineName=ttsen_US
   - To:
     - TTSType=TTS
     - InitSessionString=EngineName=tts
   - Change:
     - TTSDefinition=de_DE,TTSde_DE
     - TTSDefinition=en_US,TTSen_US
   - To:
     - TTSDefinition=en_US,TTS
     - TTSDefinition=de_DE,TTS
7. Save the file.
8. Import the configuration using **dtjconf** and start the node1 with the command **dtjstart**.

We tested the server by calling it. The default weather application was running when we dialed in. To access the German engine, press 4. To test TTS ,press 2. For speech recognition, press 3. Example 4-7 is part of the source file.

Example 4-7 VXML file of de_DE

```
<?xml version="1.0" encoding="windows-1252"?>
<VXML version="1.0" lang="de_DE" application="WeatherRoot.VXML">
```

```
<form id="startingForm">
    <!-- This form plays the initial non-barge-in introduction voice segment
-->
    <block>
        <prompt bargein="false">
            <audio src="WelcomeWeather1.wav" />
        </prompt>
        <goto next = "#main_menu"/>
    </block>
</form>

<!-- These variables will be used in several forms -->
<var name="location" expr="'Germany'"/>
<var name="report" expr="'unset'" />
<var name="selection" expr="'unset'" />
<var name="counter" expr="0"/>

<menu id="main_menu">
```

This VXML file uses `lang="de_DE"` to indicate that WebSphere Voice Response will use the German speech recognition engine and German text-to-speech engine to parse the application. The default engine is for en_US. This is set in the default.cff file, located at /usr/lpp/dirTalk/DTBE/native/aix/default.cff. The specific parameter is:

NodeDefLocale=en_US

4.9.2 Add double-byte language

We used Simplified Chinese version of WebSphere Voice Server to demonstrate how to add a double-byte language to an existing system.

The Simplified Chinese version of Voice Server was implemented on the same machine running the English version. When anybody dials this system, they will now be able to select either the English or Simplified Chinese language for their speech-recognition or text-to-speech function.

We used the following steps to install the Simplified Chinese language version:

1. Log on as `dtuser` and run **`pm exit`** to stop all of the running service for WebSphere Voice Server.
2. We log on as `root` and install all the language support filesets (Both ASR and TTS) for Simplified Chinese using **`smitty install_latest`**. The components lists as below:
   - viavoice.asr.zh_CN
   - viavoice.asr_sample.zh_CN

- viavoice.asr_server.zh_CN
- viavoice.tts.zh_CN.1at08KHz
- viavoice.tts.zh_CN.1at08KHz_1_0
- viavoice.tts.zh_CN.2at08KHz
- viavoice.tts.zh_CN.2at08KHz_1_0
- viavoice.tts.zh_CN.rte

3. Wait for the end of the installation.
4. Run **vvt_config** to config WebSphere Voice Server to use both Simplified Chinese and English voice components (see Figure 4-50 on page 179):

```
$ vvt_config

How many U.S. English speech recognition engines do you want (0-9998)? [ 1 ]

What is the engine type? [ asren_US ]

What is the application path? [ /var/vvt ]

How many Simplified Chinese speech recognition engines do you
want (0-9997)? [ 1 ]

What is the engine type? [ asrzh_CN ]

What is the application path? [ /var/vvt ]

How many language-independent text-to-speech engines do you want (U.S.
English, Simplified Chinese) (0-9996)? [ 0 ]

How many U.S. English text-to-speech engines do you want (0-9996)? [ 1 ]

What is the engine type? [ ttsen_US ]

How many Simplified Chinese text-to-speech engines do you want (0-9995)? [ 1 ]

What is the engine type? [ ttszh_CN ]

Do you have other servers you want this client to connect (y/n)? [ N ]

Integrating user modified data from current VVTdefaults
Renamed /var/vvt/cfg/VVTdefaults to /var/vvt/cfg/VVTdefaults.020
Writing new configuration ..........................
$ _
```

Figure 4-52 Config window of vvt_config

5. Run the service of WebSphere Voice Server using the **pm** command. Then monitor the status of these services using **tsmcon** (see Figure 4-53).

```
$ tsmcon
 Host          ¦ Server.cloneId ¦ Type     ¦ Status              ¦ Count  ¦ Last
Allocated
==========================================================================
==========
 9.27.111.193 ¦ asrp      .0000 ¦ asren_US ¦ available           ¦      0 ¦ <not
set>
 9.27.111.193 ¦ asrp      .0001 ¦ asrzh_CN ¦ available           ¦      0 ¦ <not
set>
 9.27.111.193 ¦ ttsp      .0000 ¦ ttsen_US ¦ available           ¦      0 ¦ <not
set>
 9.27.111.193 ¦ ttsp      .0001 ¦ ttszh_CN ¦ available           ¦      0 ¦ <not
set>
==========================================================================
==========
```

Figure 4-53 Service Status window

There are two ASR services launched. One is for English, another is for Simplified Chinese. There are also two TTS services in the server side. All of the single-byte language text-to-speech engines must be separated with double-byte language engine.

Note: Simplified Chinese TTS engine also needs to be separated from the Japanese TTS engine.

6. Stop the running node1 with the command **dtjstop**.
7. Modify /var/dirTalk/DTBE/native/aix/default.cff. Uncomment:
   - #TTSService=WVS_TTSzh_CN
   - #RecoService=WVS_Recozh_CN
   - #RecoDefinition=zh_CN,Recozh_CN
8. Save the file.
9. Then import the configuration using **dtjconf** and start the node1 with the command **dtjstart**.

We tested the server by calling it. The default weather application was running when we dialed in. To access the Simplified Chinese engine, press 9. To test TTS, press 2. For speech recognition, press 3. Example 4-8 is the part of the source file.

Example 4-8 VXML file of zh_CN

```
<?xml version="1.0" encoding="gb2312"?>
<VXML version="1.0" lang="zh_CN" application="WeatherRoot.VXML">

<form id="startingForm">
    <!-- This form plays the initial non-barge-in introduction voice segment
-->
    <block>
        <prompt bargein="false">
            <!-- Welcome to IBM WebSphere Voice Server for Direct Talk. You can
use this application to test several different functions. -->
            <audio src="WelcomeWeather1.wav" />
        </prompt>
        <goto next = "#main_menu"/>
    </block>
</form>
```

This VXML file uses `lang="zh_CN"` to indicate that WebSphere Voice Response will use Simplified Chinese speech recognition engine and Simplified Chinese text-to-speech engine to parse the application. The default engine is for en_US. This is set in the default.cff file, located at /usr/lpp/dirTalk/DTBE/native/aix/default.cff. The specific parameter is:

```
NodeDefLocale=en_US
```

4.10 Integrating AIX and Windows 2000 system

One of our test configurations allowed us to integrate a mixed operating system environment with WebSphere Voice Server. This mixed environment consisted of both an AIX and Windows 2000 machine sharing a Voice Server deployment.

The AIX system comprised an RS/6000 44P Model 170. Installed on it was WebSphere Voice Response and the Voice Server client, so that it cooperated with the other Windows system.

The Windows 2000 machine was 8658-51Y Netfinity 5100. It had the Voice Server server component installed. It was configured to support US English, with one speech recognition engine and one formant text-to-speech engine.

The network diagram is illustrated in Figure 4-54 on page 185.

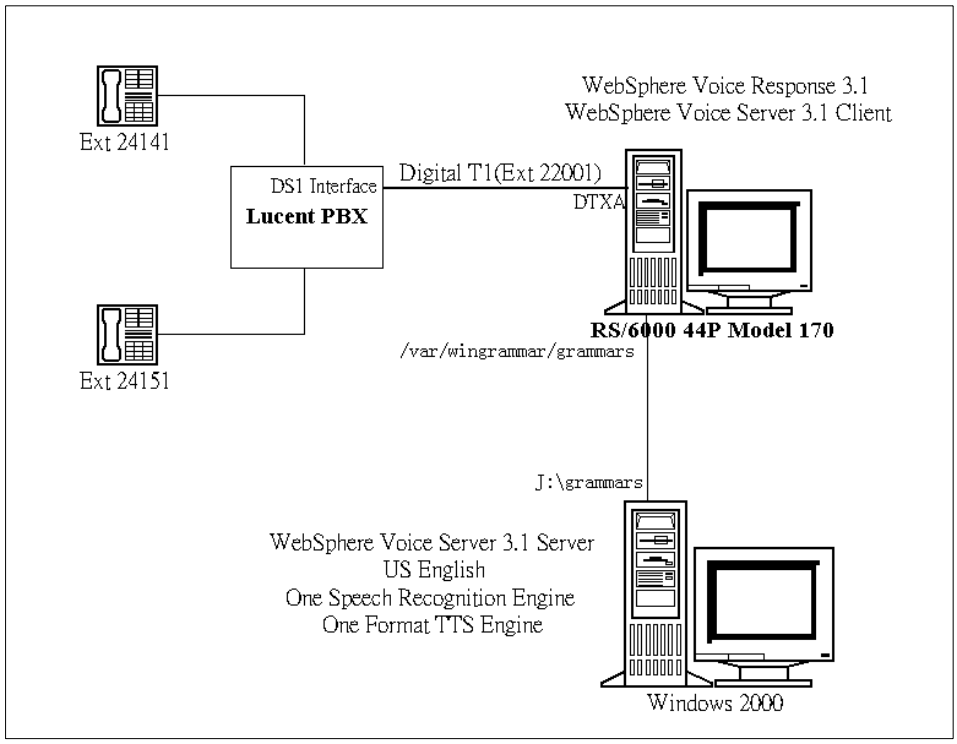


Figure 4-54 Distributed setup of WebSphere Voice Server

4.10.1 Setup

The AIX configuration

The AIX system used the folllowing settings. The important part of the configuration was the shared folder. Although the share is on an AIX system, it must be available for the Windows machine to connect to. This was done using SAMBA. (See Appendix C, “Integration of Windows 2000 and AIX” on page 543.)

- IP Address: 9.27.111.193
- Host Name: aixwvs3
- Share folder: /var/wingrammar

The Windows 2000 configuration

For the Windows-based machine, the three important settings were:

- IP Address: 9.24.104.191
- Host Name: m23wph67
- Map the AIX folder /var/wingrammar as: J:\

4.10.2 Installation

We installed the WebSphere Voice Server server for en_US on Windows 2000 machine. This was further configured to support one speech recognition engine and one formant text-to-speech engine.

4.10.3 Connection of AIX and Windows 2000

The WebSphere Voice Server's client and server must now be configured. On the AIX system, do the following:

1. Log on as dtuser.
2. Shut down WebSphere Voice Response:

   ```
   DT_shutdown
   ```

3. Shut down the Voice Server server in the AIX box:

   ```
   pm exit
   ```

4. Run **vvt_config** script in AIX box to configure the location of the speech recognition engine and text-to-speech engine as shown in Figure 4-55.

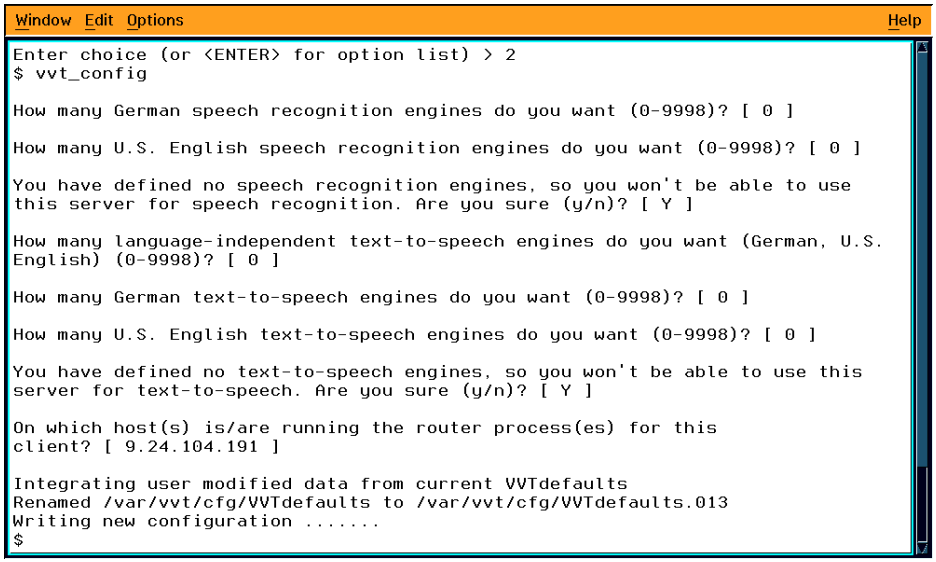


Figure 4-55 Window of vvt_config

We skipped the local speech recognition engine and text-to-speech engine settings as they will be linked to Windows 2000 box. Refer to the configuration file /var/vvt/cfg/VVTdefaults, as shown in Example 4-9 on page 187.

Example 4-9 VVTdefaults

```
# DO NOT MODIFY THIS LINE - THE CONFIGURATION UTILITY DEPENDS ON IT - BEGIN
# created on 2002/10/18 09:33:38 by dtuser
# $Id: vvtconfig.tcl,v 1.15.2.7.2.6 2002/08/20 17:22:38 reichel Exp $
#
# EVERY MODIFICATION BETWEEN THE TWO 'DO NOT MODIFY' LINES WILL BE LOST
# DURING RECONFIGURATION
#
*.*.TsmRouter.routerHost: 9.24.104.191
*.*.TsmRouter.routersToScan: 1
# DO NOT MODIFY THIS LINE - THE CONFIGURATION UTILITY DEPENDS ON IT - END
```

Note: There is one remote WebSphere Voice Server server linked in this VVTdefaults file. Its IP address is 9.24.104.191.

5. Modify the default.cff. It is found in /var/dirTalk/DTBE/native/aix/default.cff. (See Appendix E, “Default.cff with distributed system support” on page 561.)
   - If you have applied the German version, comment:

     ```
     TTSService=WVS_TTSde_DE
     RecoService=WVS_Recode_DE
     RecoDefinition=de_DE,Recode_DE
     ```

   - Change:

     ```
     TTSType=TTS
     InitSessionString=EngineName=tts,EngineFree=true
     ```

     To:

     ```
     TTSType=TTSde_DE
     InitSessionString=EngineName=ttsde_DE,EngineFree=true
     ```

   - Change:

     ```
     TTSType=TTS
     InitSessionString=EngineName=tts
     ```

     To:

     ```
     TTSType=TTSen_US
     InitSessionString=EngineName=ttsen_US
     ```

   - Change:

     ```
     TTSDefinition=en_US,TTS
     TTSDefinition=de_DE,TTS
     ```

     To:

     ```
     TTSDefinition=de_DE,TTSde_DE
     TTSDefinition=en_US,TTSen_US
     ```

– Change:

```
InitTechnologyString=LocalBaseDir=/var/vvt, ServerBaseDir=/var/vvt
```

To:

```
InitTechnologyString=LocalBaseDir=/var/wingrammar/grammars,
ServerBaseDir=j:/grammars
```

– Add at the end of the file:

```
HostName=m23wph67
  Enabled=yes
  IPName=9.24.104.191
  Node=Node1
;
```

6. Save the file.
7. Import the configuration using **dtjconf.**
8. Restart WebSphere Voice Response using **vaeinit**.
9. Stop WebSphere Voice Server server in Windows 2000 box:

```
pm exit
```

10. Run **vvtconfig.tcl** at Windows 2000 as shown in Figure 4-56.

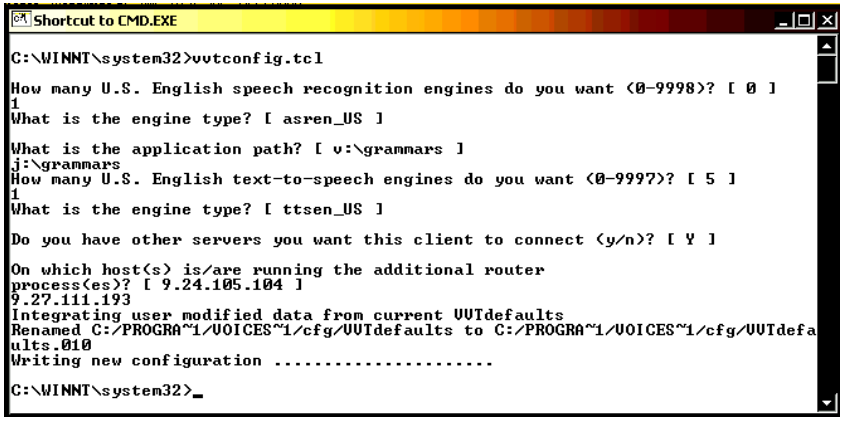


Figure 4-56 vvtconfig in Windows 2000

Its application path points to j:\grammars, which is the mapped AIX folder, and also adds AIX IP address to the router processes.

11. Start WebSphere Voice Server server in Windows 2000:

```
pm
```

Call the extension 22001, channel 1 of the T1 trunk. The Weather application should be running. Use the DTMF keys options to select the speech recognition and text-to-speech functions for English.

4.11 Managing the environment

In this section, we discuss the commands of WebSphere Voice Response and WebSphere Voice Server that allow us to manage each product within our environment.

4.11.1 Commands of WebSphere Voice Response

A comprehensive set of administrative script commands is available to manage the Java Bean WebSphere Voice Response environment. A detailed description of each command is listed in the *WebSphere Voice Response Application Development using Java and Voice XML, Version 3.1* manual. The commands are:

dtjconf	Imports the configuration default.cff file
dtjenv	Sets local environment variables
dtjflog	Formats log files
dtjplex	Performs numerous utility actions, such as importing or exporting voice segments
dtjqapps	Queries applications on Node1 of local host
dtjqhost	Queries the local host
dtjqnode	Queries all the nodes in the local host
dtjshost	Starts Remote Method Invocation (RMI) and the HostManager
dtjstart	Starts Node1 in the local host
dtjstop	Quiesces all applications and all nodes in the local host
dtjterm	Immediately stops all applications and all nodes in the local host
dtjver	Queries the current version of Java WebSphere Voice Response Beans environment

4.11.2 Commands of WebSphere Voice Server

The commands to manage the operation of speech recognition and text-to-speech resources are described in *WebSphere Voice Server Version 3.1 - Use with WebSphere Voice Response for AIX Telephony Platform Administrator's Guide*. Some of the more common commands are:

pm	Starts the Process Manager (PM)
pm exit	Stops the Process Manager
pm config	Refreshes the configuration file and restarts PM
pm list processname	Lists all processes under PM control
pm quit	Quiesces all telephony processes
pm restart processname	Stops and starts specified processes
pm start processname	Starts specified processes
pm stop processname	Quiesces specified running processes
listres	Performs numerous options
logtail	Monitors and displays system events and errors
setdbg	Sets the debug level for specified processes and host
tsmcon	Monitors and controls the speech resources on a host
text2pcm	Converts text into speech coded in .pcm or .wav format
trace	Captures trace files intended for IBM support organization

4.11.3 Some tips

To have the voice application using speech recognition and text-to-speech technologies favorably accepted by the end users, the system should be tuned to suit the environments before it is put into production. The grammar file that is developed on a PC using a microphone may perform differently in a production environment, since the frequency response of a telephone network is much narrower than that of a high-quality microphone.

The `listres --help` command shows the various options available for the `listres` command. See Figure 4-57 on page 191.

The **listres.tcl** command shows the list of configured resources in the VVTdefaults file. The list of resources used in the ITSO implementation is shown in Figure 4-57 on page 191. Some of these values will most likely require modification before the system is put into a production environment.

Changes to the VVTdefaults file will not take effect until the processes are stopped and started. Use the **pm exit** command to stop the processes and the **pm** command to start the processes.

Every time when you start WebSphere Voice Response, the hardware test is running by default. You can use **export SDI_DISABLE_DIAGS=1** to disable this test. This will also speed up the starting of WebSphere Voice Response.

.

```
Usage:
The listres.tcl utility displays the current values of the
specified resources.
USAGE:
   listres.tcl [-v] <pattern>
   listres.tcl [--help ] [pattern|"all"|"objects"]>
PARAMETERS:
   --help - Displays this help text.
   --help all - Displays description of all resources.
   --help objects - Displays list of all object names.
   --help <pattern> - Displays description of resources matching the pattern.
    -v    - Lists source of the resource value (file or defaults).
EXAMPLES:
   listres.tcl "*" // lists all explicitly defined resources
   listres.tcl Asr // lists all resources defined for *asr* objects
   listres.tcl --help asr // displays description of all resources
                          // defined for *asr* objects
```

Figure 4-57 Output of the listres.tcl --help command

5

Cisco telephony environment

This chapter gives a detailed step-by-step description of how to install and configure WebSphere Voice Server for Windows 2000 Version 2.0 for the Cisco environment. It explains how to configure Voice Server and its components. At the same time, we explain how we tested the software in a variety of ways.

5.1 Voice Server for Cisco Environment installation

In the Cisco telephony environment, the server does not need any specialized hardware cards to be installed. The telephony traffic uses the network connectivity to communicate. This is known as Voice over IP, or VoIP for short. Voice Server utilizes the H.323 VoIP standards to communicate between the telephony interface device, in this case a Cisco router, and itself.

Figure 5-1 depicts a simple Voice Server configuration, using one Voice over IP gateway and one computer. A more complex configuration may have more than one gateway and more than one Voice Server computer.

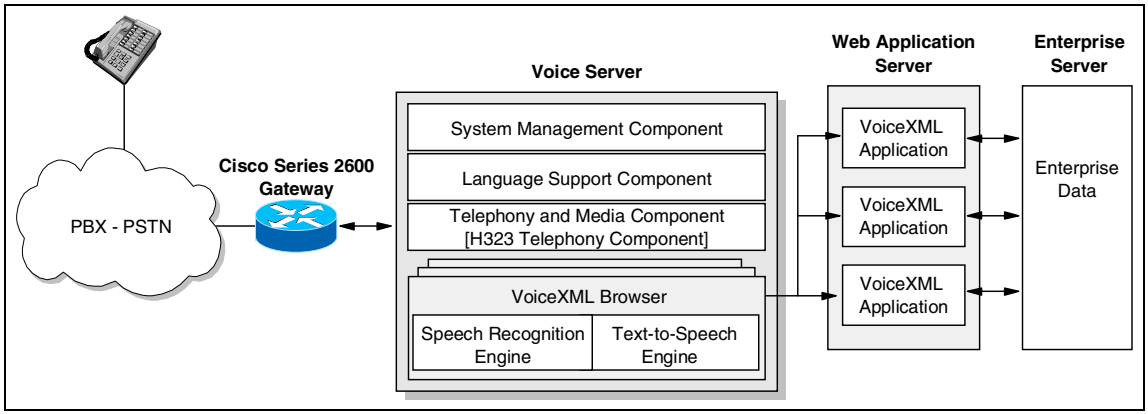


Figure 5-1 Voice Server in a Cisco environment

5.1.1 Requirements for Voice Server in a Cisco environment

Each computer that will have Voice Server needs to have at least the minimum hardware and software prerequisites stated in the following sections.

Hardware

The computer must be equipped with:

- Intel Pentium 1 GHz or faster processor
- CD-ROM drive
- 256 MB RAM (minimum)
- 430 MB disk space (minimum), which includes:
  - 30 MB for installing Sun Java Runtime Environment (Sun JRE) 1.3.1 or higher
  - 100 MB in the Windows system directory

  - 200 MB in the installation destination directory for *each* Voice Server language-support selected
  - Additional disk space in the installation destination directory for caching, logging, and tracing, as needed
- Monitor, keyboard, mouse, and their required cords and cabling

The telephony network must be equipped with:

- Cisco H.323 V2.0 compliant VoIP gateway:
- Cisco 2600 Series Router

  Note: The VoIP gateway should support the following:

  - G.711 uncompressed audio standard
  - DTMF detection and extraction
  - In-band and out-of-band DTMF
- Ethernet 10/100 network connection supporting VoIP
- PSTN/GSM telephone connection
  - ISDN PRI T1/E1
  - CAS T1
- Telephone
- VoIP LAN

  Note: A VoIP LAN with Quality of Service (QoS) is recommended.

Software

On each computer where you install Voice Server you will need:

- Microsoft Windows 2000 Server Edition (French, German, Japanese, or US English only) with Service Pack 2. We successfully tested Service Pack 3.
- WebSphere Voice Server for Windows 2000 Language Support (in French, German, Japanese, UK English, or US English).

  Important: If you plan to operate Voice Server in more than one language, all Language Support CDs must be installed *before* installing WebSphere Voice Server for use with Cisco telephony platform.

  At the time of writing, the support languages for Voice Server in a Cisco environment were:

  - US English
  - GB English, UK
  - German
  - French
  - Japanese

Note: Please refer to Chapter 10, “Voice Server language component” on page 453 for more information regarding language support installation.

- Sun Java Runtime Environment (Sun JRE) 1.3.1 (included on the Voice Server installation CD). This software must be installed prior to the Voice Server software.
- Adobe Acrobat Reader Version 5 or later (included on the Voice Server installation CD).
- HTTP 1.1 client.

For the Cisco 2600 Series gateway, you will need Cisco IOS Version 12.1(5) XM (XM4 recommended).

Before installing WebSphere Voice Server for Cisco 2.0, the system must have Java Runtime Environment 1.3.1 or higher installed, as well as Windows 2000 at Service Pack 2. Failure to do this will stop Voice Server installing.

Language support must be installed *before* the Voice Server component. A language cannot be installed later onto Voice Server. At the time of writing, the support languages for Voice Server in a Cisco environment were:

- US English
- GB English, UK
- German
- French
- Japanese

Note: Page link to language support installation.

5.1.2 Voice Server installation

In our case we had an executable file to install Voice Server. You should have a CD labeled WebSphere Voice Server for Windows 2000 - Use with Cisco telephony platform.

Step 1: Start install

Either run the file or insert the CD. You will be presented with a window like the one shown in Figure 5-2 on page 197. Click **Next**.

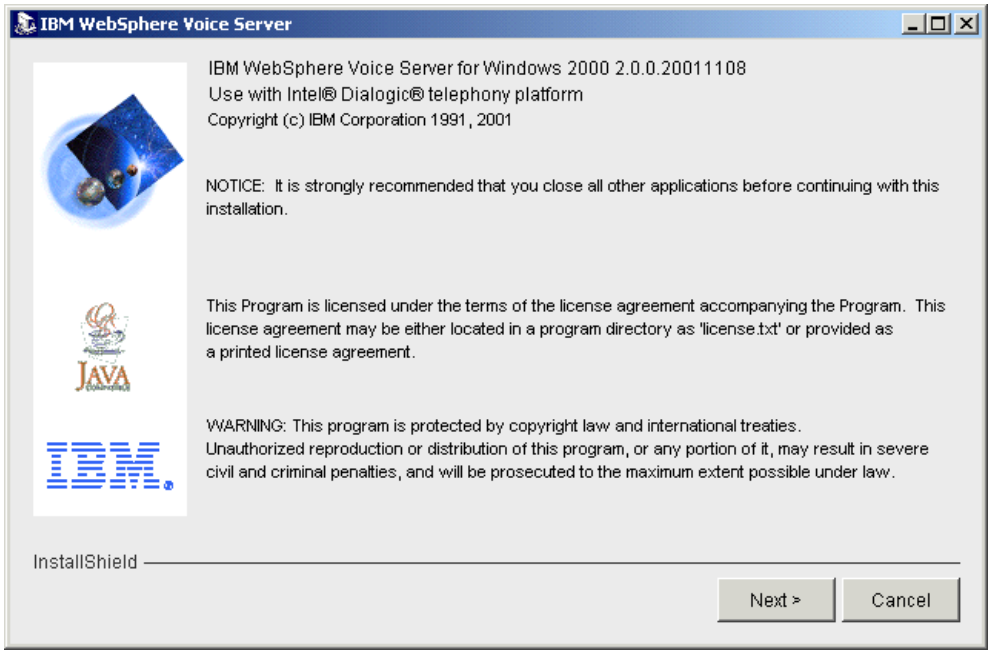


Figure 5-2 Welcome window

Step 2: Prerequisite check

Voice Server will check for the necessary prerequisites before it will continue. If one does not pass, the installation will fail. Correct the problem and try again. If all pass, click **Next**, as shown in Figure 5-3 on page 198.

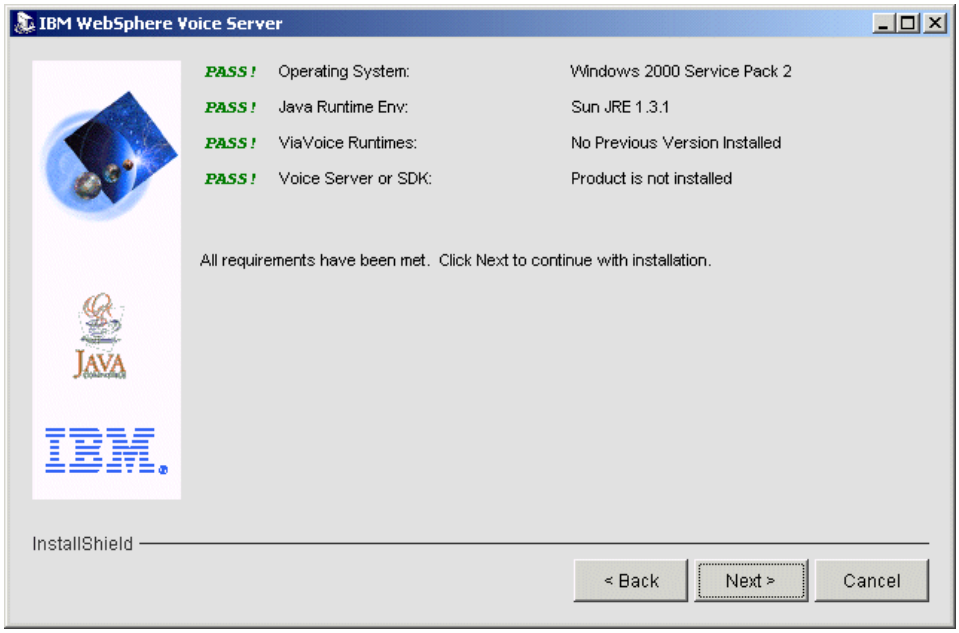


Figure 5-3 Prerequisite window

Step 3: License agreement

The license agreement appears. Read and accept it as in Figure 5-4 on page 199. Click **Next**.

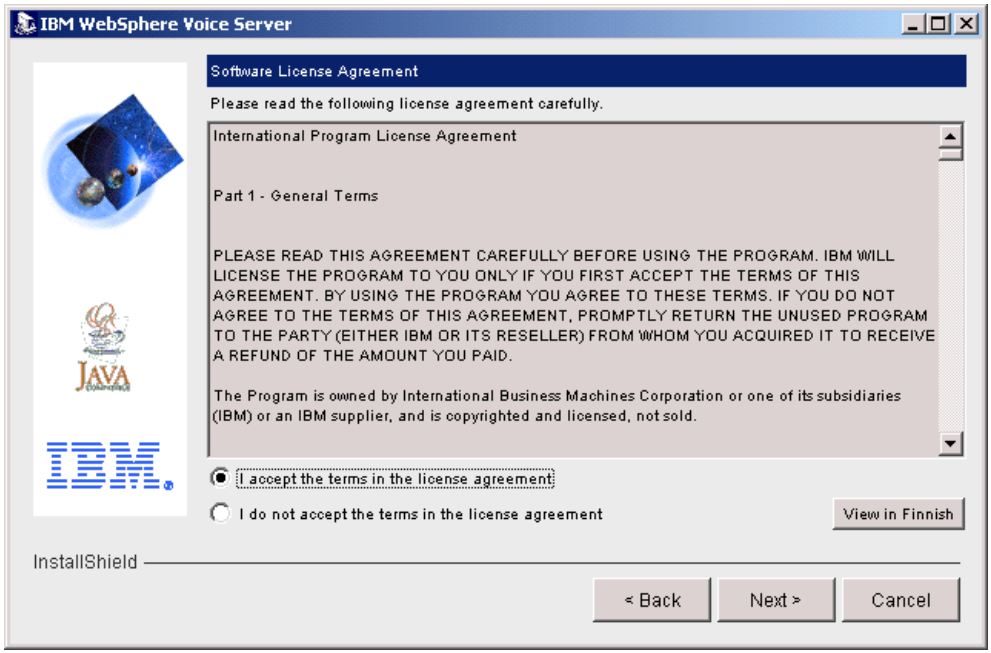


Figure 5-4 License agreement

Step 4: README file

The README file is displayed, as illustrated in Figure 5-5 on page 200. Read for any last-minute installation changes. Click **Next**.

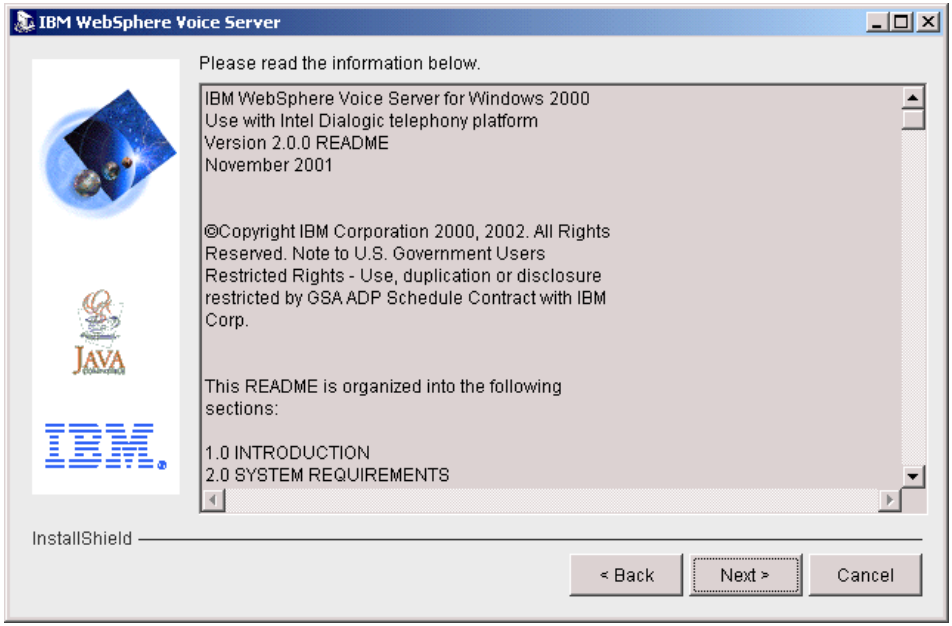


Figure 5-5 README welcome

Step 5: Creation of destination directory

Voice Server needs to be installed into a directory. A default is provided. We took this option. Click **Next** after setting the destination directory, as shown in Figure 5-6 on page 201.

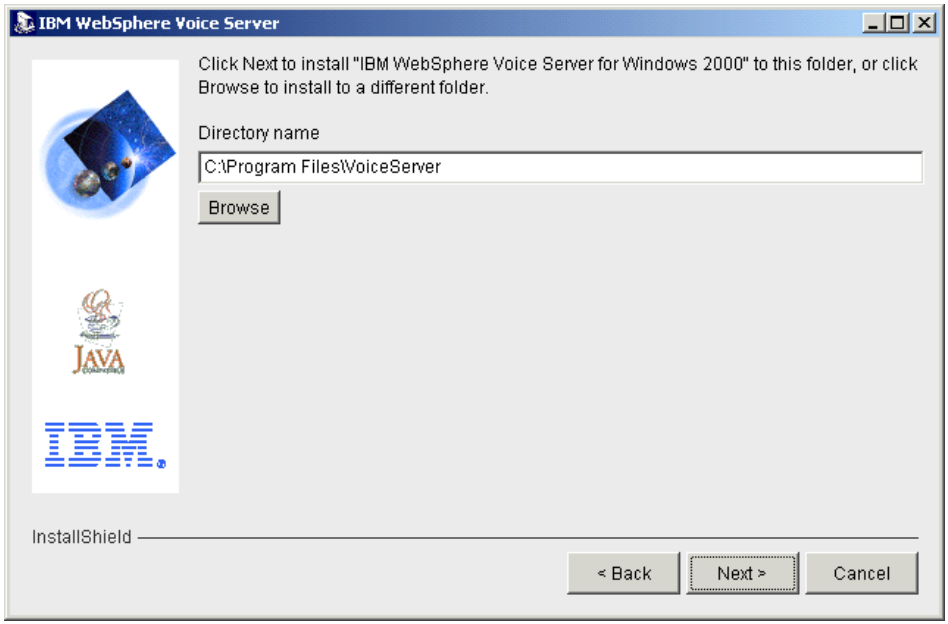


Figure 5-6 Destination directory window

A warning appears advising the default directory does not exist. Click **Yes** to create it, as in Figure 5-7.

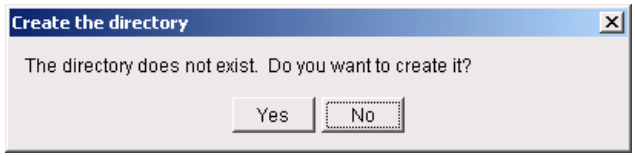


Figure 5-7 Directory creation

Step 6: Language support

This window displays all the supported languages for the Dialogic environment. Within this list you will see three languages highlighted: US English, UK English and German. These are the languages we installed during the Language Support components step. Click **Next**, as in Figure 5-8 on page 202.

Note: Once Voice Server is installed, additional languages cannot be installed afterwards, that is, on top of the server.

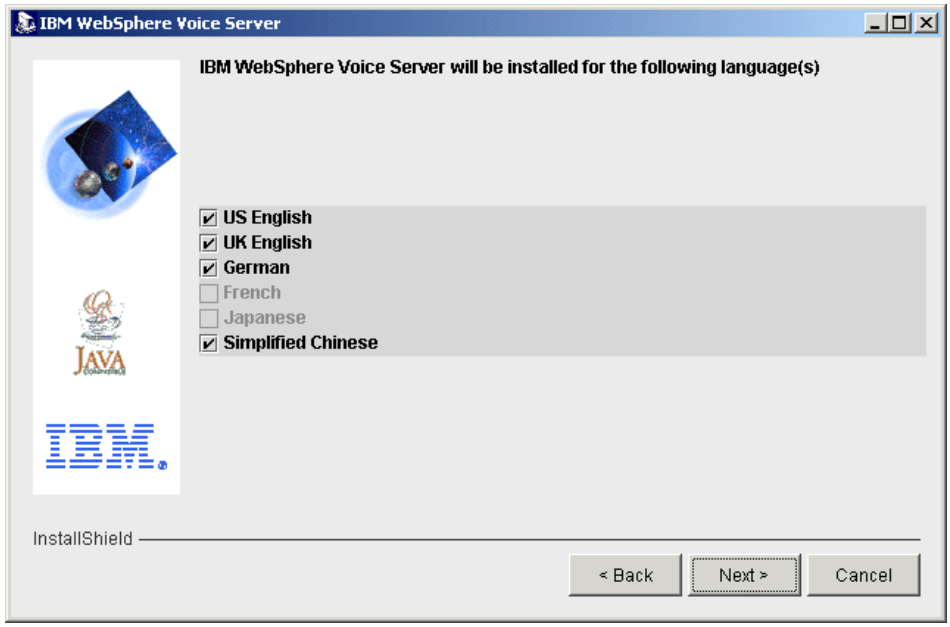


Figure 5-8 Language window

Step 7: Voice Server installing

For each language, Voice Server installs that version of itself. In our scenario, US English, UK English, and German versions of Voice Server were installed. Figure 5-9 on page 203 shows US English, and Figure 5-10 on page 203 shows UK English.

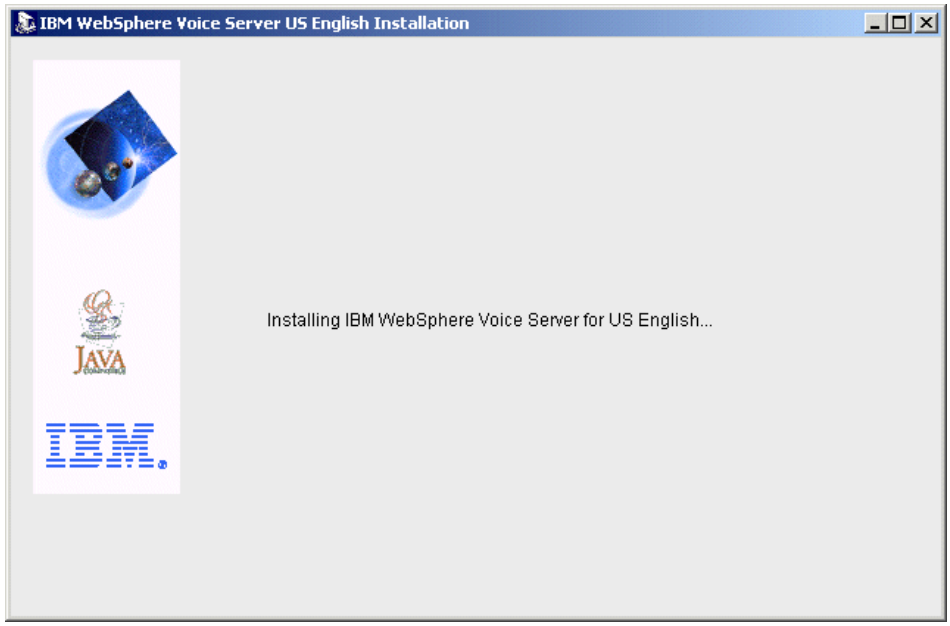


Figure 5-9 US English Voice Server

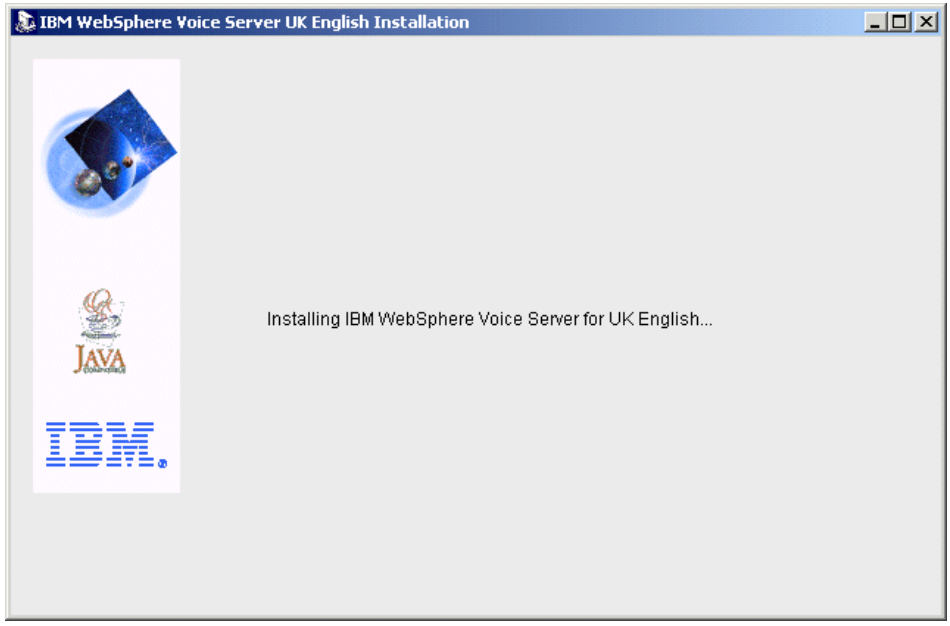


Figure 5-10 UK English Voice Server

Step 8: Completion

Figure 5-11 shows the completion of the Voice Server installation. Click **Finish** to wrap it up. At this point you should reboot your machine before continuing.

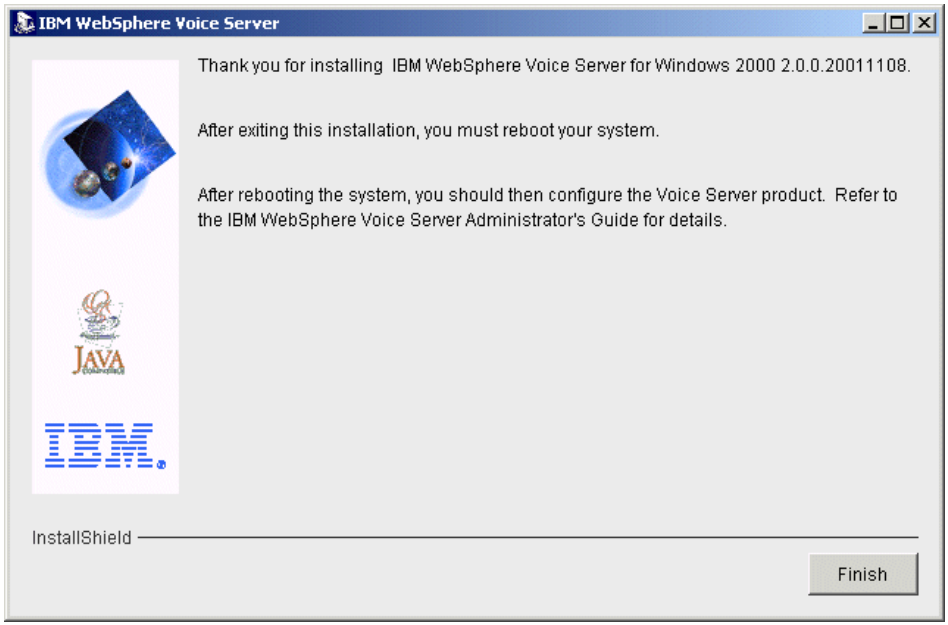


Figure 5-11 Completion window

5.2 Telephony environment configuration

This version of Voice Server requires a Cisco 2600 Series gateway to function. Our recommendation is that you work very closely with your Cisco specialist to configure this gateway correctly. Failure to do this may not allow some or all of the telephony functions to work with Voice Server. The impact of this would ultimately affect your customers trying to call in.

The telephony environment tested was an Avaya DEFINITY. We had a T1 trunk connected from the PBX directly into our Cisco 2600 series gateway. The T1 was configured to handle CAS loopstart.

Our Cisco 2600 had the following features:

- IOS 12.1(5)XM (This IOS level supports call transfer)
- IP Plus
- 16 MB flash and 48 MB of RAM

Example 5-1 shows the final configuration file for the Cisco gateway in our test environment.

Example 5-1 Cisco configuration

```
!
! Last configuration change at 18:28:25 UTC Fri Dec 7 2001
! NVRAM config last updated at 18:29:35 UTC Fri Dec 7 2001
!
version 12.1
no parser cache
no service single-slot-reload-enable
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname cisco-gw
!
no logging buffered
logging rate-limit console 10 except errors
no logging console
enable secret 5 $1$9VC1$tjO8Yb5WNO4K3mQyJduxb0
enable password vs
!
!
!
voice-card 1
!
ip subnet-zero
!
!
no ip finger
no ip domain-lookup
!
no mgcp timer receive-rtcp
call rsvp-sync
!
!
!
!
!
!
!
controller T1 1/0
 ds0-group 0 timeslots 1-24 type fxo-loop-start
!
!
interface FastEthernet0/0
 ip address 10.1.1.99 255.255.255.0
 duplex auto
```

```
 speed 100
!
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
!
ip classless
no ip http server
!
snmp-server packetsize 4096
!
voice-port 1/0:0
 timeouts wait-release 3
 connection plar 7101
!
dial-peer cor custom
!
!
!
dial-peer voice 7101 voip
 destination-pattern 7101
 session target ipv4:10.1.1.17
 dtmf-relay h245-signal h245-alphanumeric
 codec g711ulaw
 no vad
!
dial-peer voice 1000 pots
 application session
 incoming called-number 7101
 destination-pattern 24151
 no digit-strip
 port 1/0:0
!
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password vs
 login
!
ntp clock-period 17178863
ntp server 10.1.1.3
end
```

More detailed information about the Cisco gateway and configuration information can be found at these Web addresses:

- Data Sheet

  http://www.cisco.com/warp/public/cc/pd/rt/2600/prodlit/2600_ds.htm

- Digital T1/E1 packet voice trunk network modules

  http://www.cisco.com/univercd/cc/td/doc/pcat/dit1e1p1.htm

- Understanding high density voice network modules

  http://www.cisco.com/warp/public/788/products/hdv_netmod.html

Configuration planning is covered in Appendix A,"Cisco VoIP Gateway Configuration" in the *WebSphere Voice Server Version 2.0 - Use with Cisco Telephony Platform Administrator's Guide*, G210-1262.

5.3 Checking your installation

Once you have the Cisco gateway configured, Voice Server needs to be configured to accept and handle the incoming calls. In our test lab, the telephony environment was an isolated system. We also had a separate network dedicated to the VoIP traffic we generated.

You can control the behavior of Voice Server by means of System Management parameters in the file sysmgmt.properties. When Voice Server is installed, a sample sysmgmt.properties file is included. There are two types of configuration parameters:

- General
- VoiceXML browser

The configuration file documents all possible parameters in the comment sections of the file (lines whose first character is #). Only those parameters that must be explicitly assigned (that is, parameters that are required and have no default value) are required in this file. Review the parameters in more detail to understand what they do.

The file is located in the Voice Server subdirectory. Example 5-2 includes the system management file of our tested system.

Example 5-2 System management file

```
################################################################################
## Licensed Materials - Property of IBM
##      5724-B59
## (C) Copyright IBM Corp. 2000, 2002 All Rights Reserved
## US Government Users Restricted Rights - Use, duplication or
```

```
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
##############################################################################

#------------------------------------------------------------------------------
# This file contains the IBM WebSphere Voice Server Version 2.0
# System Management configuration parameters.
#
# The file defines 2 types of configuration parameters:
#
#    1. General
#    2. VoiceXML Browser
#
#------------------------------------------------------------------------------

#------------------------------------------------------------------------------
#
# General Configuration Parameters
#
#   ALARM_FILE_MAX_SIZE
#   AUTO_START_ON_INIT
#   BROWSER_CACHE_MAXSIZE
#   BROWSER_CACHE_PATHNAME
#   BROWSER_DIRECTORIES_KEPT
#   CALLMANAGER_HOST (Dialogic only)
#   HTTP_PROXY
#   LOG_FILE_MAX_SIZE
#   NUMBER_INBOUND_BROWSERS
#   SYS_MGMT_OUTPUT_PATHNAME
#
#------------------------------------------------------------------------------
ALARM_FILE_MAX_SIZE=1
AUTO_START_ON_INIT=false
BROWSER_CACHE_MAXSIZE=1
BROWSER_CACHE_PATHNAME=cache
LOG_FILE_MAX_SIZE=1
NUMBER_INBOUND_BROWSERS=1
SYS_MGMT_OUTPUT_PATHNAME=logs


#------------------------------------------------------------------------------
#
# VoiceXML Browser Configuration Parameters
#
#  NOTE: The "n" in INBOUND_*n parameters identifies the VoiceXML browser
#  instance. Valid values are 0 through (NUMBER_INBOUND_BROWSERS - 1) and
#  correspond to the VoiceXML browser naming convention vvi0"n".
#
#   INBOUND_BARGE_INn
#   INBOUND_CALLED_NUMBERn
```

```
#   INBOUND_DUPLEXn
#   INBOUND_LOCALEn (*)
#   INBOUND_SITEn
#   INBOUND_TIMEOUTn
#   INBOUND_URLn
#
# (*) NOTE: Valid locales are:
# en_US for US English
# en_GB for UK English
# fr_FR for French
# de_DE for German
# ja_JP for Japanese
# zh_CN for Simplified Chinese (Dialogic only)
#
#-------------------------------------------------------------------------------
INBOUND_URL0=file:///Program
Files/VoiceServer/samples/en_US/AudioSample/AudioSample.VXML
```

5.3.1 Specific parameters

In this section, we discuss specific parameters that effect the operation of our telephony environment.

AUTO_START_ON_INIT

The VoiceXML browsers will not start automatically when Voice Server is powered up because the preconfigured value of the AUTO_START_ON_INIT parameter is false. Edit the sysmgmt.properties file and set the AUTO_START_ON_INIT parameter to true to start the VoiceXML browsers automatically when the Voice Server is powered up.

NUMBER_INBOUND_BROWSERS

The parameter NUMBER_INBOUND_BROWSERS specifies the number of browsers you have running to handle inbound calls. For each inbound browser, there must be a corresponding INBOUND_URL. This tells the browser where the VoiceXML file is located.

Note: If the number specified in the NUMBER_INBOUND_BROWSERS does not match the total number of INBOUND_URLs, Voice Server will not start.

INBOUND_URL

When the Voice Browser is started, the application it runs is set by the INBOUND_URL parameter. The default setting points to the sample application called AudioSample.VXML. This is a simple VoiceXML application that prompts the caller to record some speech and then plays the recorded audio back to the caller. It is located in < install-directory>/samples/< locale> /AudioSample.VXML, where < locale> is one of the following 5-character language and country codes:

- de_DE for German
- fr_FR for French
- ja_JP for Japanese
- zh_CN for Simplified Chinese
- en_GB for UK English
- en_US for US English

Note: The URL must not be on a network drive. Any loss of the network drive will cause the Voice Browser to fail.

If you configured Voice Server in a client/server configuration, you should modify the sample application AudioSample.VXML to indicate which client's browser is accepting a call. You can do this by adding, for example, a prompt to say "This is the client one". This scheme can be used to verify that each client is successfully communicating with the server. If you installed VoiceXML browsers on the server, you should confirm that those browsers also accept calls.

INBOUND naming convention

The INBOUND_*n parameters are configured using a naming convention that is applied to the parameter names for VoiceXML browsers. Table 5-1 illustrates the naming convention.

Table 5-1 Browser naming convention

VoiceXML Browser vvi00	**VoiceXML Browser vvi01**	**VoiceXML Browser vvi02**	**VoiceXML Browser vvi0n**
INBOUND_BARGE_IN*0*	INBOUND_BARGE_IN*1*	INBOUND_BARGE_IN*2*	INBOUND_BARGE_IN*n*
INBOUND_DUPLEX_IN*0*	INBOUND_DUPLEX_IN*1*	INBOUND_DUPLEX_IN*2*	INBOUND_DUPLEX_IN*n*
INBOUND_URL*0*	INBOUND_URL*1*	INBOUND_URL*2*	INBOUND_URL*n*

5.3.2 Validating the configuration file

A system management component is included with Voice Server. This provides basic system management and system administration functions. It is responsible for starting, stopping, and monitoring VoiceXML browsers. It also provides logging and error data in the SM log and alarm files. There is one System Management component on every computer on which Voice Server components are installed.

To validate the sysmgmt.properties configuration file before you start up the VoiceXML browsers, issue the **SMConfig** command from a Windows command prompt.

Each time **SMConfig** detects an invalid parameter in the configuration file, it will display an error message on the console and stop running. Correct any errors and re-run **SMConfig** until a `No Errors` status message is returned to the console, indicating that the configuration file passed all syntax validation.

Note: Only after the system reboots and the Voice Server service is running in Windows can you use the System Management component to start the VoiceXML browsers.

5.3.3 Using Voice Server commands

Once the system management component is started and the VoiceXML browsers are operational, you can monitor their operation by means of System Management commands.

Note: You must log on with an Administrator ID at each client computer in order to use these commands for VoiceXML browsers running on that computer.

The system management commands are listed in Table 5-2.

Table 5-2 System management commands

Command	Description
vvsm start	Starts all configured VoiceXML browsers. This forces the system to read the sysmgmt.properties configuration file; it is ignored if the VoiceXML browsers are already running.
vvsm getstatus	Returns the status of all VoiceXML browsers running on this computer.
vvsm locale	Returns a list of installed language locales.

Command	Description
`vvsm help`	Displays a list of all available commands with a brief description of each.
`vvsm stop now`	Requests an immediate stop to all VoiceXML browsers. This request will not wait until the current calls end before shutting down the VoiceXML browsers. It will terminate all on-going calls immediately.
`vvsm stop`	Requests all VoiceXML browsers to gracefully shut down. This request is asynchronous (that is, the command ends before the VoiceXML browsers have stopped). Browsers that are in the “received call” state will not terminate until the call dialog ends, or a 10-minute timeout is reached.

5.4 The final test

Once the parameters have been set, checked and verified, reboot the machine. The machine will start with Voice Server running. Use the system management commands to verify that Voice Server is waiting on call status, as shown in Figure 5-12.

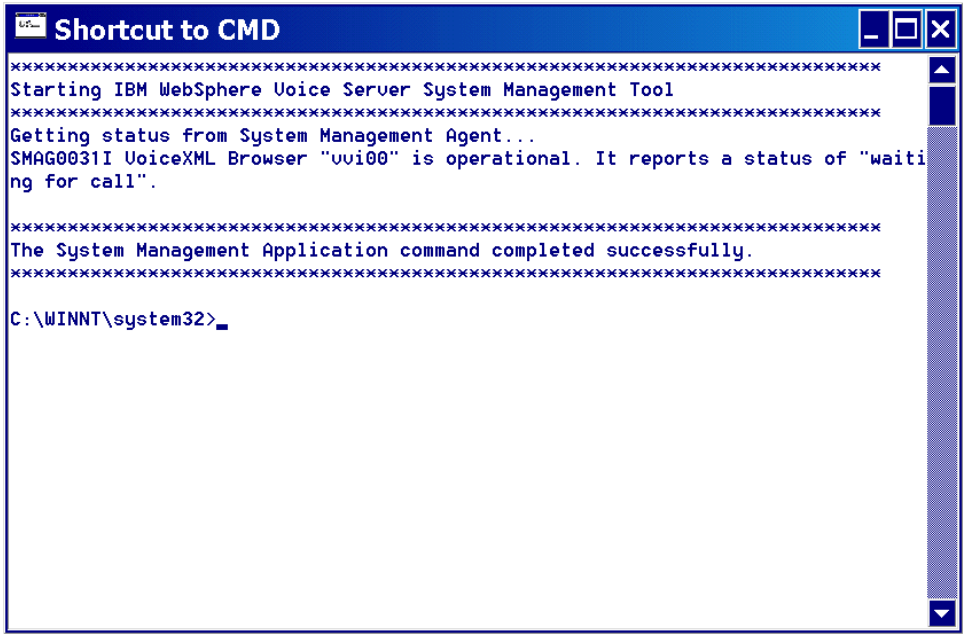


Figure 5-12 System management window

Place a call to Voice Server. You should hear your application running!

5.5 Testing on a PC

Testing of Voice Server for Cisco was also achieved using Microsoft NetMeeting. This application comes with Windows 2000 or can be downloaded from the following Web site:

http://www.microsoft.com/windows/netmeeting/download/

For users of Windows XP, click **run** in the Start button and type `conf`. In both cases if this is the first time you start NetMeeting, it will need to be configured.

NetMeeting is a Voice over IP (VoIP) H.323-compliant client application. This means you can call and interact with Voice Server, without the need of Cisco hardware for testing purposes.

The PC that will be running the application will need a sound card with both input and output capabilities. A good quality microphone headset is also required to effectively test Voice Server with NetMeeting. A low-budget microphone will pick up ambient room noise and make it hard for Voice Server to distinguish your voice input. Figure 5-13 on page 214 shows a ready-to-use NetMeeting session.

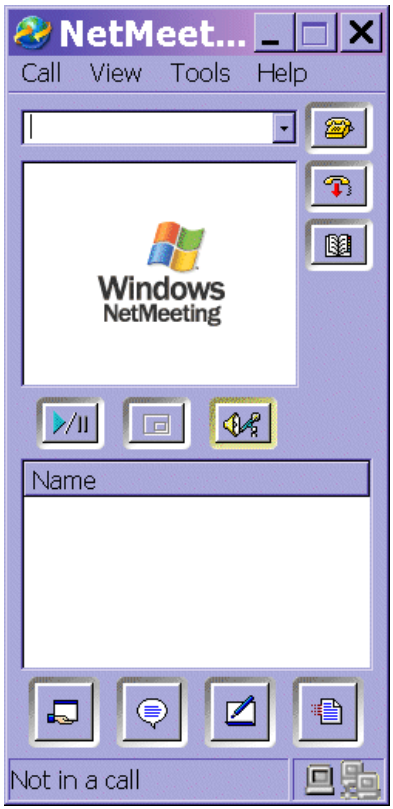


Figure 5-13 Active NetMeeting console

NetMeeting needs to know the IP address of the PC running Voice Server. In our environment, Voice Server was running on IP address 9.24.104.155. Once entered, click the phone button. NetMeeting will attempt to call the server, as shown in Figure 5-14.

Figure 5-14 Calling Voice Server

Once communication is established, the Voice Browser will start running the VXML application through NetMeeting. At any time you can adjust the playback and recording level within NetMeeting, as illustrated in Figure 5-15 on page 215. This may be necessary with record level as the microphone maybe set too high, causing recognition to be inaccurate.

Figure 5-15 Audio and recording adjustment

5.6 Voice Server considerations

During our testing and configuring of the Cisco implementation of Voice Server, we took the following into account:

- Number of phone lines

  The Cisco 2600 is the only supported gateway. The maximum number of phone calls this unit can support is two trunks, as follows:

  – 2 x T1 48 lines
  – 2 x E1 60 lines

- Call transfer

  The Cisco gateway uses tromboning for call transfer. An incoming call when transferred needs to use a second line out, thus tying up two phone lines. The two lines will remain in use until the call has been terminated. Hence, if call transfer is implemented, the maximum number of lines is halved. For two T1s it drops to 24, and for two E1s it has 30 lines.

6

Dialogic environment

This chapter gives a detailed step-by-step description of how to install and configure WebSphere Voice Server for Windows 2000 Version 2.0 for the Intel Dialogic telephony platform. It explains the Voice Server deployment configurations and describes the different components of Voice Server. We explain how we tested the software in a variety of ways.

6.1 Voice Server in the Dialogic environment

Voice Server on the Dialogic platform incorporates the use of specialized telephony cards, manufactured by Dialogic. These cards are connected directly to the telephony interface. Calls are then managed by the Dialogic software to pass incoming calls to the Voice Server application. Voice Server then controls the interaction between the end user and the applications it runs. These cards can allow multiple concurrent calls. Voice Server is also scalable, starting from a basic analog configuration to high-end solutions where specialized T1/E1 lines are connected.

6.1.1 System configurations

In the Dialogic-based Voice Server system, there are three possible configuration schemes. This feature gives Voice Server the benefit of distributing the workload over several machines rather than just one. The three schemes are:

1. A **stand-alone system** (Figure 6-1). All Voice Server components are installed and reside on one machine. This machine has the Intel Dialogic platform—the server computer containing the Dialogic hardware and software.

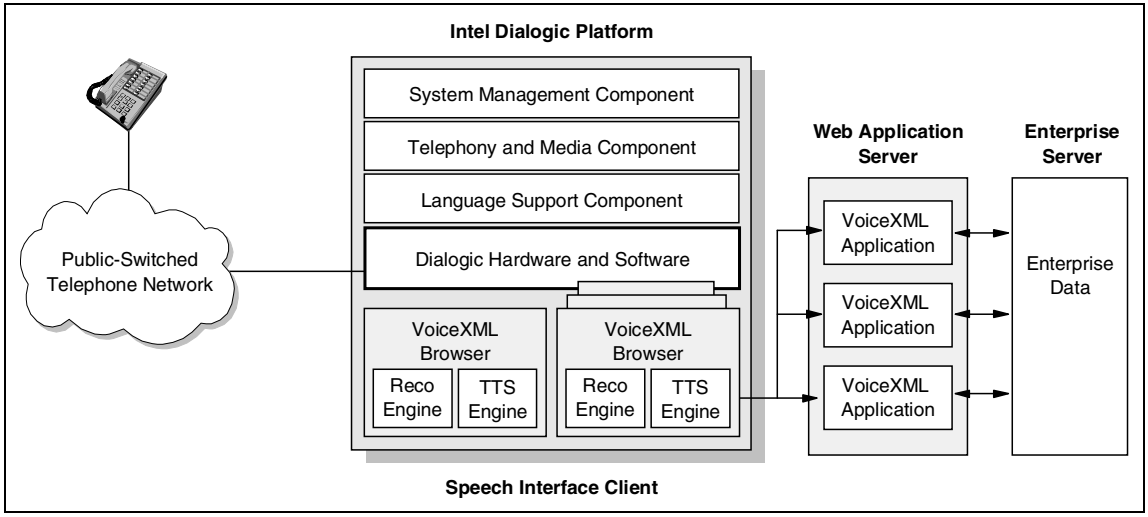


Figure 6-1 Stand-alone configuration

In a stand-alone system, Dialogic hardware, software, and all Voice Server components are installed and running on a single system. In this case we have two browsers running, meaning two phone calls can be handled concurrently.

2. **Full distribution, client/server system** (Figure 6-2). VoiceXML browsers are installed on one or more speech interface client computers but not on the server computer. The server contains the Intel Dialogic hardware and software and is called the telephony server.

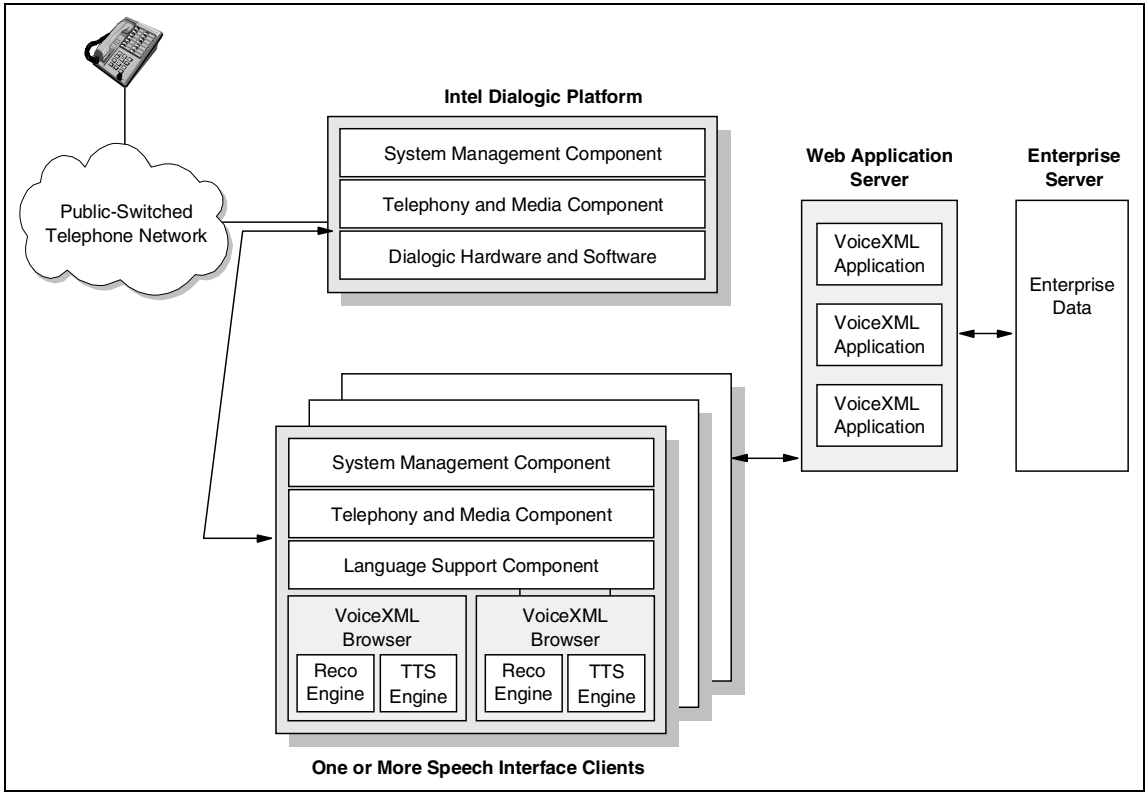


Figure 6-2 Full distribution configuration

The recommended Full Distribution configuration is a client/server system in which Voice Server's VoiceXML browsers run solely on speech interface client machines.

3. **Mixed distribution, client/server system** (Figure 6-3 on page 220). VoiceXML browsers are installed on both the server and on one or more speech interface client computers.

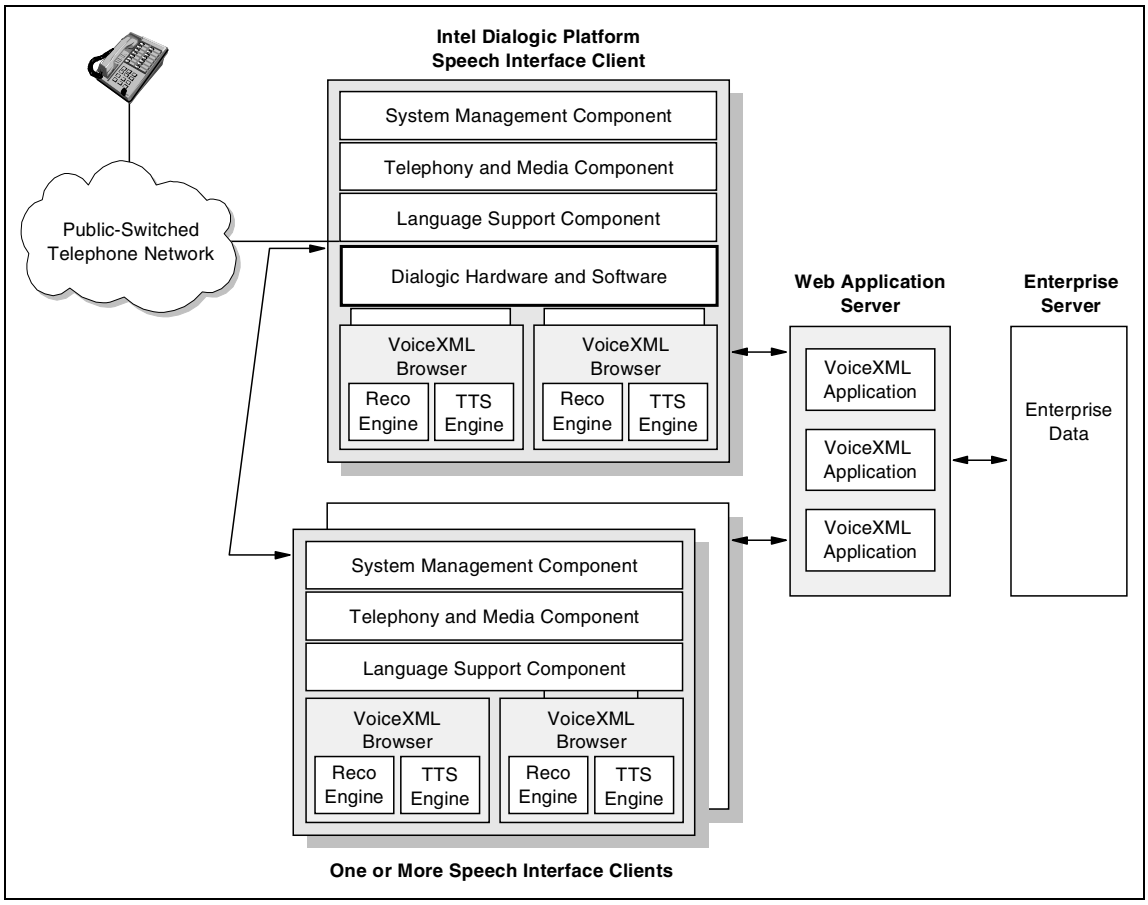


Figure 6-3 Mixed distribution configuration

This mixed distribution configuration is a client/server system in which Voice Server's VoiceXML browsers are distributed among speech interface client computers and Voice Server's Intel Dialogic platform server.

Note: In this type of configuration, the Language Support component must reside on all computers housing the VoiceXML browsers.

Consideration must also be made for licensing when planning a distributed solution. Each client is classed as an implementation of Voice Server, regardless of it being a Voice Server component or the Voice browser component. That being the case, the license is charged on a CPU basis. So if the client has two processors, you pay for two licenses. If a server has one processor, it has one license. This is further documented in the licensing agreement for the software.

6.2 Hardware and software prerequisites

In each of the three configurations, the hardware and software prerequisites are important, since they may vary slightly. Each configuration is listed below with both the Voice Server requirements for the server and the client. You must ensure these are met before installing Voice Server.

Note: If your computer is running a version of Norton AntiVirus older than Version 6.0, either upgrade to the latest version or uninstall Norton AntiVirus before installing Voice Server. Versions of Norton AntiVirus that predate Version 6.0 cause conflicts with Voice Server during system startup.

6.2.1 Prerequisite hardware/software for stand-alone configuration

Each computer needs to have at least the minimum hardware and software prerequisites stated in the following sections.

Hardware

In the stand-alone configuration, all Voice Server components are installed on one machine. Ensure there are adequate hardware resources, namely CPU speed and memory, that can cope with the demand from Voice Server. The minimum requirements are:

- Intel Pentium 500 MHz as a base processor speed plus:
  - 10 MHz for each telephony channel to be activated
  - 80 MHz for each VoiceXML browser to be started
- CD-ROM drive
- 400 MB RAM plus:
  - 13 MB RAM for each telephony channel to be activated
  - 36 MB RAM for each VoiceXML browser to be started
- 210 MB disk space (minimum), which includes:
  - 25 MB for installing the Sun Java Runtime Environment (Sun JRE) 1.3.1
  - 40 MB in the Windows system directory
  - 100 MB in the installation destination directory for *each* Voice Server language-support selected
  - Additional disk space in the installation destination directory for caching, logging, and tracing, as needed
- 240–600 MB disk space, depending on the protocols you install, for the software supplied with the Dialogic board

- Monitor, keyboard, mouse, and their required cords and cabling
- One of the following Dialogic board models:
  - D/120JCT-LS (a 12-port analog board)
  - D/240JCT-T1 (a 24-port digital board)
  - D/480JCT-1T1 (a dual-span 24-port digital board)
  - D/480JCT-2T1 (a dual-span 48-port digital board)
  - D/600JCT-1E1 (a dual-span 30-port digital board)
  - D/600JCT-2E1 (a dual-span 60-port digital board)
- Telephone (for testing) and connections from the Dialogic board to the PSTN

Software

In the stand-alone configuration, the following are required:

- Microsoft Windows 2000 Server Edition (French, German, Japanese, Simplified Chinese, or US English) with Service Pack 2.
- WebSphere Voice Server for Windows 2000 language support (in French, German, Japanese, Simplified Chinese, UK English, or US English).

Note: If you plan to operate Voice Server in more than one language, all language support CDs must be installed *before* installing WebSphere Voice Server for use with Intel Dialogic telephony platform. A description of the installation of language support is included in Chapter 10, “Voice Server language component” on page 453.

- Adobe Acrobat Reader Version 5 (included on the Voice Server installation CD).
- Dialogic System Release 5.01 (for North America) or 5.1 (for China, Europe, or Japan)
- Intel GlobalCall Protocols Release 1.00 for Windows

You can download the Dialogic products from

http://developer.intel.com/design/telecom/support/

6.2.2 Prerequisite hardware/software for full distribution configuration

Each computer needs to have at least the minimum hardware and software prerequisites stated in the following sections.

Hardware

For the server with the Intel Dialogic platform, you need a computer with:

- Intel Pentium 500 MHz as a base processor speed plus 10 MHz for each telephony channel to be activated
- CD-ROM drive
- 400 MB RAM plus 13 MB RAM for each telephony channel to be activated
- 80 MB disk space (minimum), which includes:
  - 25 MB for installing the Sun Java Runtime Environment (Sun JRE) 1.3.1
  - 20 MB in the Windows system directory
  - Additional disk space in the installation destination directory for caching, logging, and tracing, as needed
- 240–600 MB disk space, depending on the protocols you install, for the software supplied with the Dialogic board
- Monitor, keyboard, mouse, and their required cords and cables
- One of the following Dialogic board models:
  - D/120JCT-LS (a 12-port analog board)
  - D/240JCT-T1 (a 24-port digital board)
  - D/480JCT-1T1 (a dual-span 24-port digital board)
  - D/480JCT-2T1 (a dual-span 48-port digital board)
  - D/600JCT-1E1 (a dual-span 30-port digital board)
  - D/600JCT-2E1 (a dual-span 60-port digital board)
- Telephone (for testing) and connections from the Dialogic board to the PSTN

For each speech interface client, you need a computer with:

- Intel Pentium 500 MHz as a base processor speed plus 80 MHz for each VoiceXML browser to be started
- CD-ROM drive
- 400 MB RAM plus 36 MB for each VoiceXML browser to be started
- 210 MB disk space (minimum), which includes:
  - 25 MB for installing the Sun Java Runtime Environment (Sun JRE) 1.3.1
  - 40 MB in the Windows system directory
  - 100 MB in the installation destination directory for *each* Voice Server language support selected
- Additional disk space in the installation destination directory for caching, logging, and tracing, as needed
- Monitor, keyboard, mouse, and their required cords and cabling

- LAN connections among all configured server and client computers

Software

On the Intel Dialogic platform server computer, you need the following:

- Microsoft Windows 2000 Server Edition (French, German, Japanese, Simplified Chinese, or US English) with Service Pack 2.
- Sun Java Runtime Environment (Sun JRE) 1.3.1 (included on the Voice Server installation CD). This software must be installed prior to the Voice Server software.
- Adobe Acrobat Reader Version 5 (included on the Voice Server installation CD).
- Dialogic System Release 5.01 (for North America) or 5.1 (for China, Europe, or Japan)
- Intel GlobalCall Protocols Release 1.00 for Windows

You can download these Dialogic products from

http://developer.intel.com/design/telecom/support/

On each speech interface client computer, you will need:

- Microsoft Windows 2000 Server Edition (French, German, Japanese, Simplified Chinese, or US English) with Service Pack 2.
- WebSphere Voice Server for Windows 2000 language support (French, German, Japanese, Simplified Chinese, UK English, or US English).
- Sun Java Runtime Environment (Sun JRE) 1.3.1 (included on the Voice Server installation CD). This software must be installed prior to the Voice Server software.
- Adobe Acrobat Reader Version 5 (included on the Voice Server installation CD).

6.2.3 Prerequisite hardware/software for mixed distribution configuration

Each computer needs to have at least the minimum hardware and software prerequisites stated in the following sections.

Hardware

Here the telephony server on the Dialogic platform also acts as a voice browser client. Care must be taken to accommodate the extra CPU and memory requirements for each browser active on the computer.

For the server running the Intel Dialogic platform, you need a computer with:

- Intel Pentium 500 MHz as a base processor speed plus:
  - 10 MHz for each telephony channel to be activated
  - 80 MHz for each VoiceXML browser to be started
- CD-ROM drive
- 400 MB RAM plus:
  - 13 MB RAM for each telephony channel to be activated
  - 36 MB RAM for each VoiceXML browser to be started
- 210 MB disk space (minimum), which includes:
  - 25 MB for installing the Sun Java Runtime Environment (Sun JRE) 1.3.1
  - 40 MB in the Windows system directory
  - 100 MB in the installation destination directory for *each* Voice Server language support selected
  - Additional disk space in the installation destination directory for caching, logging, and tracing, as needed
- 240–600 MB disk space, depending on the protocols you install, for the software supplied with the Dialogic board
- Monitor, keyboard, mouse, and their required cords and cabling
- One of the following Dialogic board models:
  - D/120JCT-LS (a 12-port analog board)
  - D/240JCT-T1 (a 24-port digital board)
  - D/480JCT-1T1 (a dual-span 24-port digital board)
  - D/480JCT-2T1 (a dual-span 48-port digital board)
  - D/600JCT-1E1 (a dual-span 30-port digital board)
  - D/600JCT-2E1 (a dual-span 60-port digital board)
- Telephone (for testing) and connections from the Dialogic board to the PSTN

For each speech interface client, a computer with:

- Intel Pentium 500 MHz as a base processor speed plus 80 MHz for each VoiceXML browser to be started
- CD-ROM drive
- 400 MB RAM plus 36 MB RAM for each VoiceXML browser to be started
- 210 MB disk space (minimum), which includes:
  - 25 MB for installing the Sun Java Runtime Environment (Sun JRE) 1.3.1
  - 40 MB in the Windows system directory

  - 100 MB in the installation destination directory for *each* Voice Server supported language selected
  - Additional disk space in the installation destination directory for caching, logging, and tracing, as needed
- Monitor, keyboard, mouse, and their required cords and cabling
- LAN connections among all configured server and client computers

Software

On the Intel Dialogic platform server, you need the following:

- Microsoft Windows 2000 Server Edition (French, German, Japanese, Simplified Chinese, or US English) with Service Pack 2.
- WebSphere Voice Server for Windows 2000 language support (in French, German, Japanese, Simplified Chinese, UK English, or US English).
- Adobe Acrobat Reader Version 5 (included on the Voice Server installation CD).
- Dialogic System Release 5.01 (for North America) or 5.1 (for China, Europe, or Japan)
- Intel GlobalCall Protocols Release 1.00 for Windows

You can download these Dialogic products from

`http://developer.intel.com/design/telecom/support/`

On each speech interface client computer, you will need:

- Microsoft Windows 2000 Server Edition (French, German, Japanese, Simplified Chinese, or US English) with Service Pack 2.
- WebSphere Voice Server for Windows 2000 language support (in French, German, Japanese, Simplified Chinese, UK English, or US English).
- Adobe Acrobat Reader Version 5 (included on the Voice Server installation CD).

6.2.4 Language support component

Before the actual Voice Server code can be loaded, the language support component must first be installed. The language support component provides speech recognition and text-to-speech operations. Voice Server needs to install a different version of itself for each language. This means to have both French

and US English, the language component for each must be installed. Also, in the case of US English and UK English, these are considered two different languages, and so their associated language support must be installed for each. On the Dialogic platform, one or more of the following languages are supported:

- French
- German
- Japanese[1]
- Simplified Chinese[2]
- UK English
- US English

Important: If you plan to operate Voice Server in more than one language, the language support CDs must be installed *before* all other Voice Server components on all speech interface client computers (where VoiceXML browsers will operate).

6.3 Dialogic environment

Voice Server requires certain prerequisite software and hardware before installation. We proceeded in the following order:

1. Install Dialogic software.
2. Shut down the computer and install the Dialogic card.
3. Configure the card.
4. Test the card.

This enables the telephony hardware to integrate with the Voice Server. There are two applications: Dialogic System Release 5.x.x for Windows 2000 and GlobalCall Protocol Package 1.00 for Windows 2000. These are available either from a CD supplied with the card or by downloading from the following Web site:

http://developer.intel.com/design/telecom/support/

There are several versions of the Dialogic software. Tested versions were 5.0.1, 5.1 and 5.1 with Service Pack 1, 5.1.1 and 5.1.1 with Service Pack 1(for Japanese, Release 5.1 is required as a minimum).

[1] Japanese language support can be implemented only in the Japanese language version of Windows 2000.
[2] Simplified Chinese language support can be implemented only in the Chinese language version of Windows 2000.

6.3.1 Installing Dialogic System Release 5.1.1

In this section, complete the following instructions to install the Dialogic software.

Step 1: Starting the Dialogic software installation

In these steps, the Dialogic software is installed. In our document, Version 5.1.1 will be installed. Some of the windows may vary slightly if you install a different version of the Dialogic software. Once downloaded and unzipped, run the setup.exe program by double-clicking it. Click **Next**. The setup window is shown in Figure 6-4.

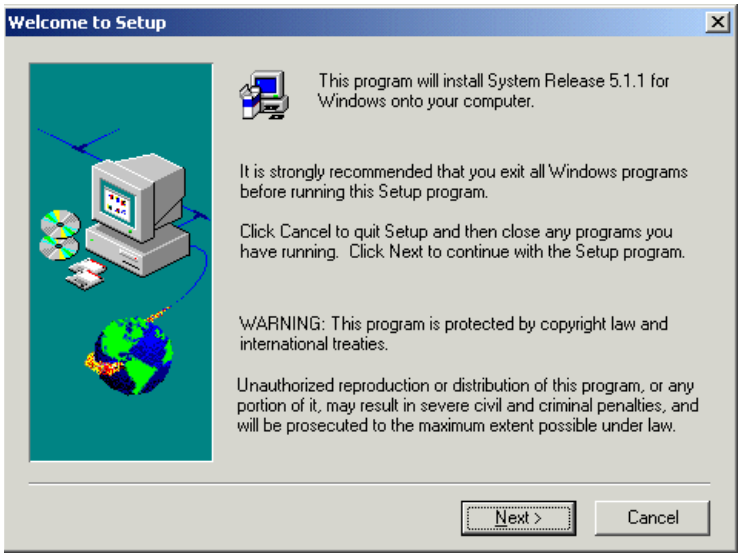


Figure 6-4 Setup window

Step 2: Release Guide

The Release Guide can be viewed at this point, as shown in Figure 6-5. We choose **No**. Click **No**.

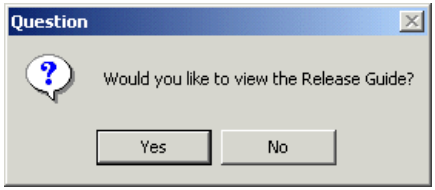


Figure 6-5 Release Guide window

Step 3: License

The license agreement appears as shown in Figure 6-6. You are required to read this. Click **Next** when completed.

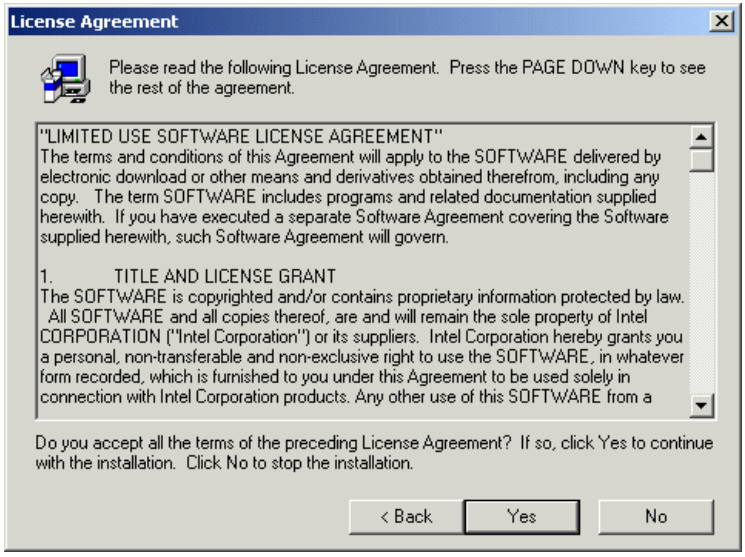


Figure 6-6 license agreement

Step 4: Registration

You are required to enter a name and company as shown in Figure 6-7 on page 230. Click **Next** when completed.

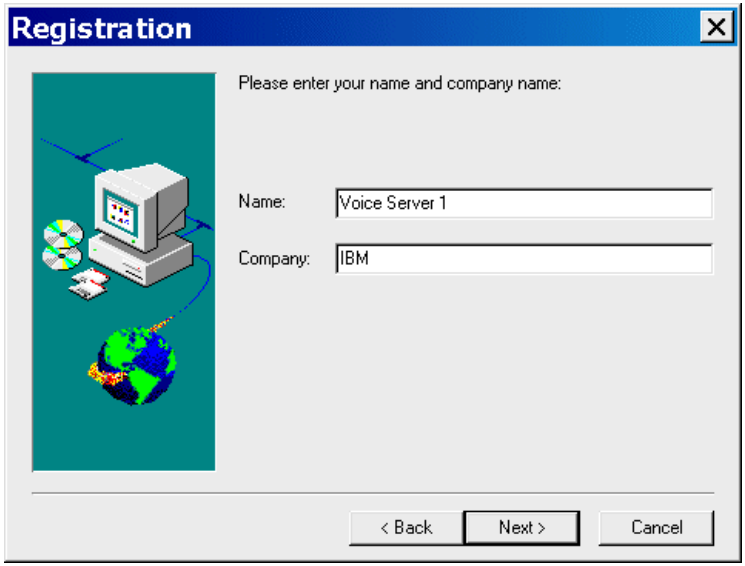


Figure 6-7 Registration window

Step 5: Setup options

The setup options give you four choices. The *WebSphere Voice Server Version 2.0 - Use with Intel Dialogic Telephony Platform Administrator's Guide*, G210-1263 states to install everything. This will use around 600 MB of hard drive space. Alternatively, we successfully installed using the minimum custom install. Click **Custom** on the options window shown in Figure 6-8 on page 231, then click **Next**.

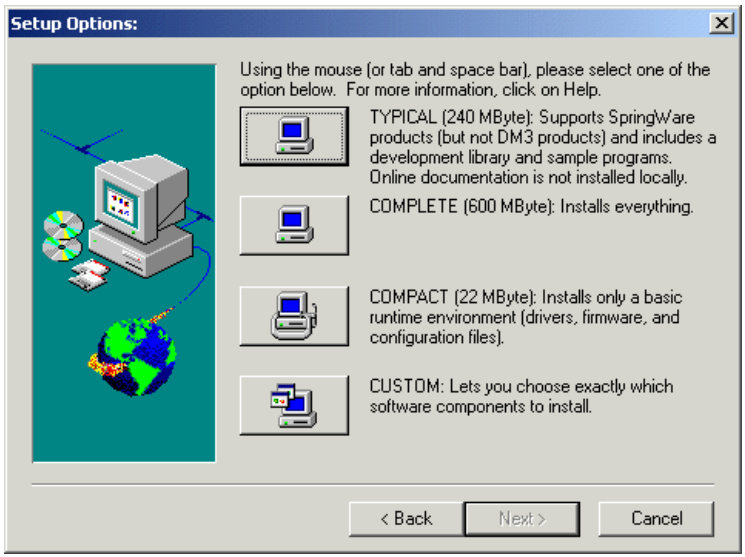


Figure 6-8 Options window

Step 6: Selection options

We chose to install the following options:

- Dialogic Drivers, Firmware and Configuration Files
- Dialogic Development SDK
- Sample Programs
- Springware TAPI Service Provider
- Online Documentation
- Performance Counters for Win NT Perf. Monitor
- GlobalCall API Package
- SCx
- Continuous Speech Processing (scroll down for this option)

Select these as illustrated in Figure 6-9 on page 232, and then click **Next**.

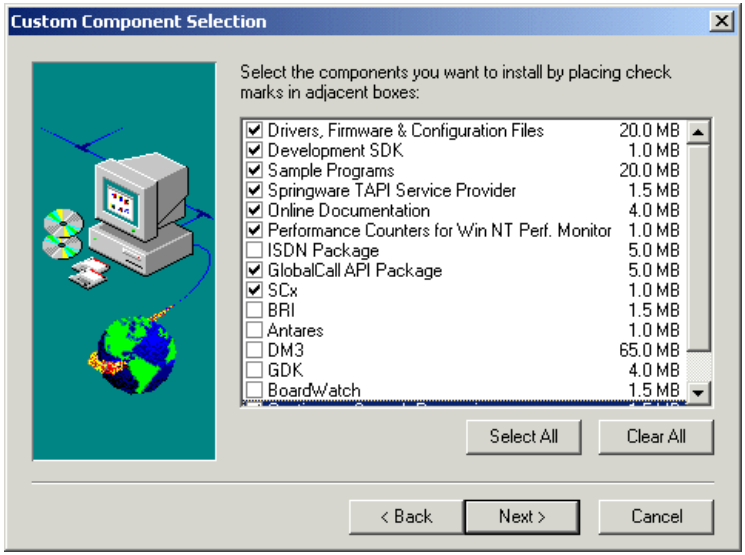


Figure 6-9 Custom component selection

Step 7: Online documentation

For better use of the documentation, select **Install Documentation Locally** as shown in Figure 6-10. Click **Next**.

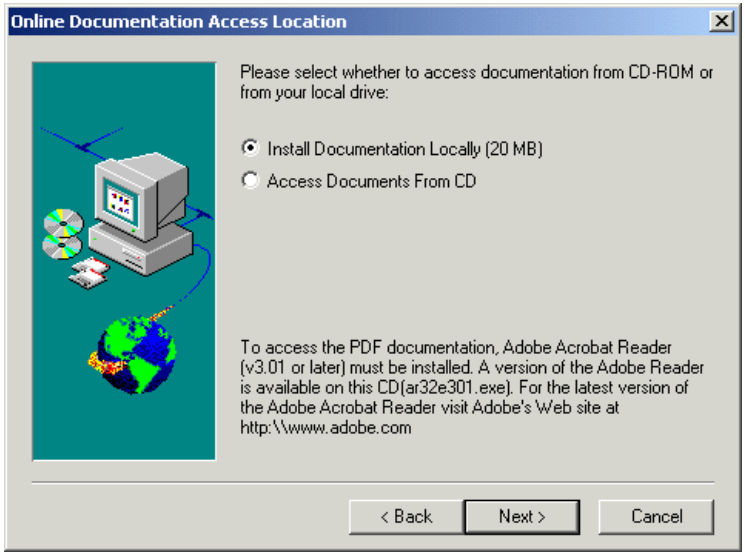


Figure 6-10 Documentation window

Step 8: Installation destination

The default directory path is displayed, as demonstrated in Figure 6-11. You may change this. We used the default. Click **Next**.

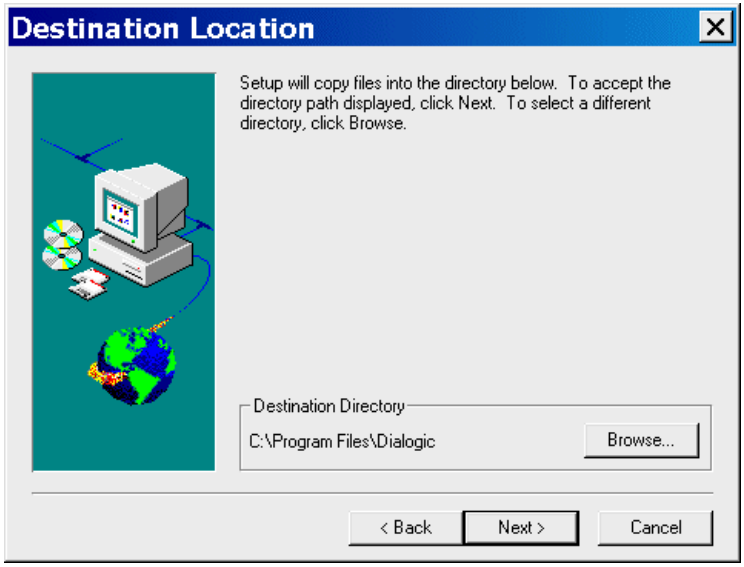


Figure 6-11 Destination window

Step 9: Program folder

The default program group folder is displayed in Figure 6-12 on page 234. We used the default. Click **Next**.

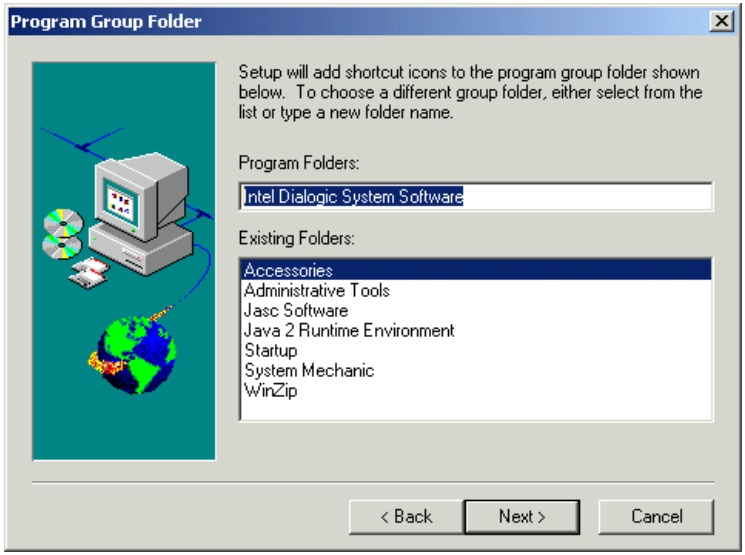


Figure 6-12 Program folder window

Step 10: Setup summary

Before the final installation, a summarization of the options is presented as shown in Figure 6-13. If these are correct, continue by clicking **Next**.

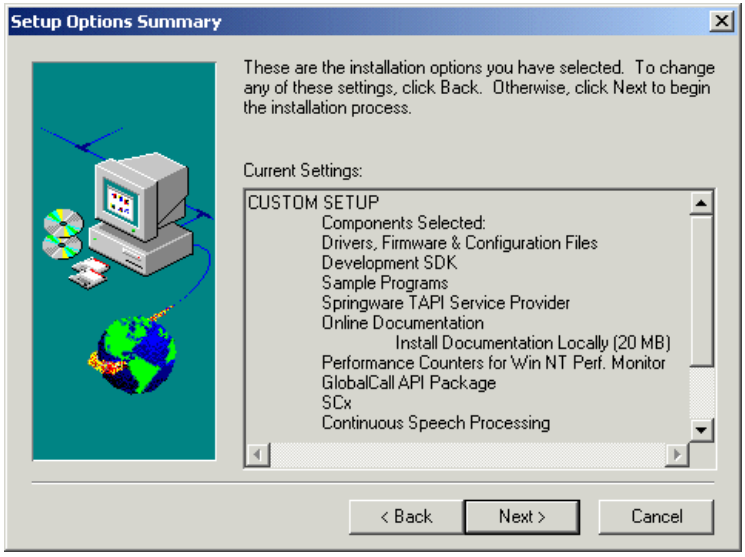


Figure 6-13 Summary window

Step 11: Dialogic Configuration Manager (DCM)

Once installation is complete, you are able to read the Release Guide and run the Dialogic Configuration Manager. (DCM). Since we have not installed the card yet, deselect this option as shown in Figure 6-14. Click **Next**.

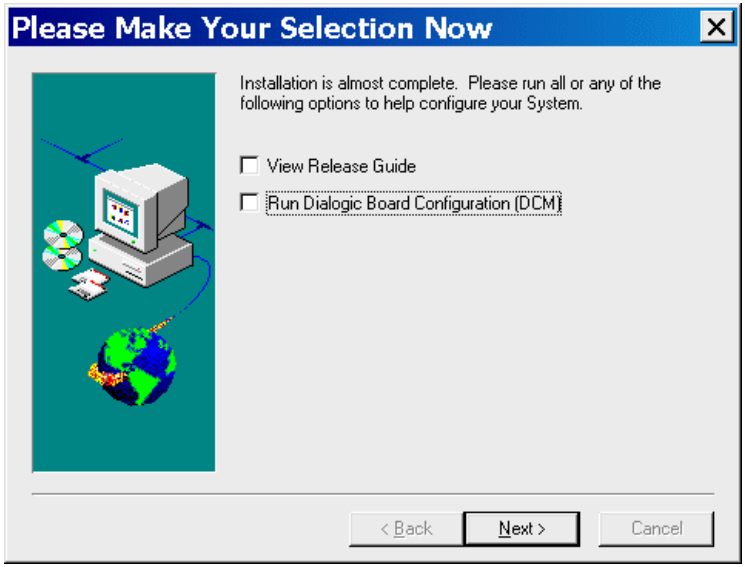


Figure 6-14 DCM window

Step 12: Performance monitor

The Dialogic software installs a performance monitor. This will not work until you reboot the system. A warning message advises you of this (see Figure 6-15). Click **OK**.

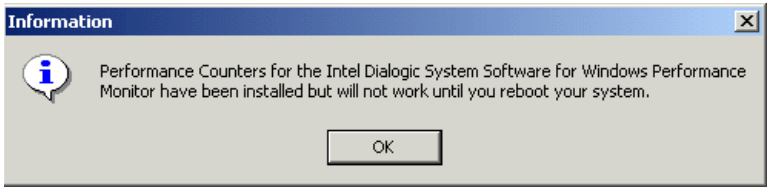


Figure 6-15 Monitor warning

Step 11: Rebooting

To ensure the Dialogic software is fully functional, you need to reboot, as shown in Figure 6-16 on page 236.

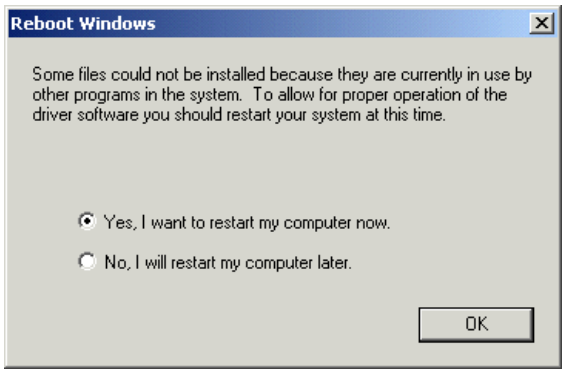


Figure 6-16 Reboot window

After rebooting, you can either:

- Install Service Pack 1 for 5.11
- Continue to the GlobalCall Protocol installation

6.3.2 Installing Dialogic System Release 5.1.1 Service Pack 1

We installed the Dialogic System Release 5.1.1 Service Pack 1. The Service Pack was downloaded from the Dialogic Web site: http://developer.intel.com/design/telecom/support/.

These are the steps to complete the installation of Service Pack 1.

Step 1: Run setup.exe

Once downloaded and unzipped, run the setup.exe program. Click **Next** as in Figure 6-17 on page 237.

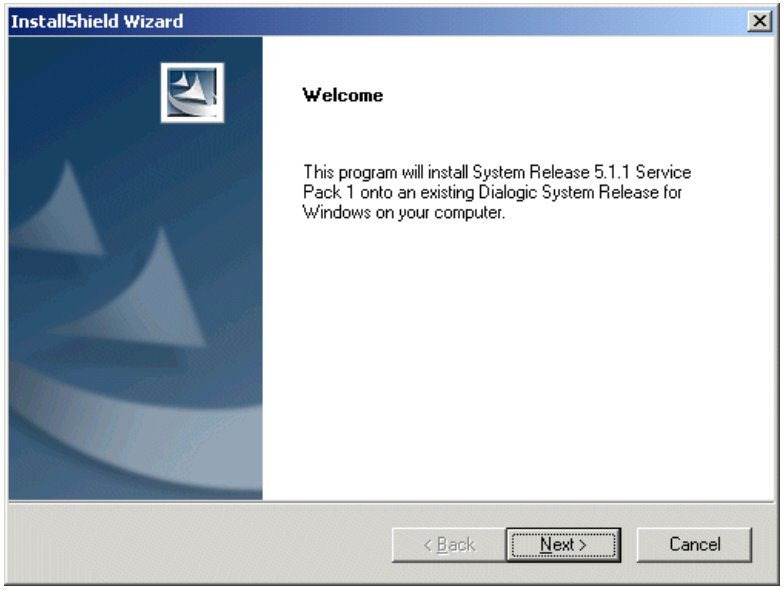


Figure 6-17 Install window

Step 2: License agreement

The license agreement is presented as shown in Figure 6-18 on page 238. Click **Yes**.

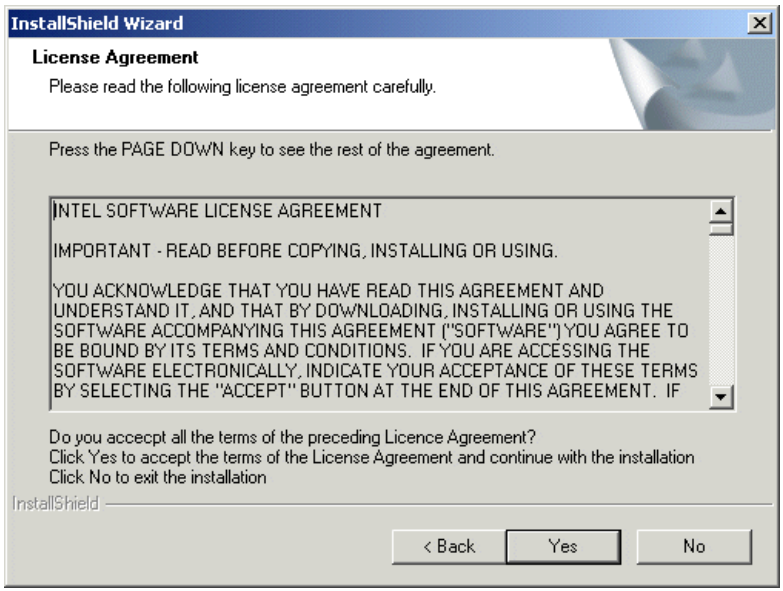


Figure 6-18 License window

Step 3: Registration

Ensure you use the same name and company you entered when installing the base code of 5.1.1 as shown in Figure 6-19 on page 239. Click **Next**.

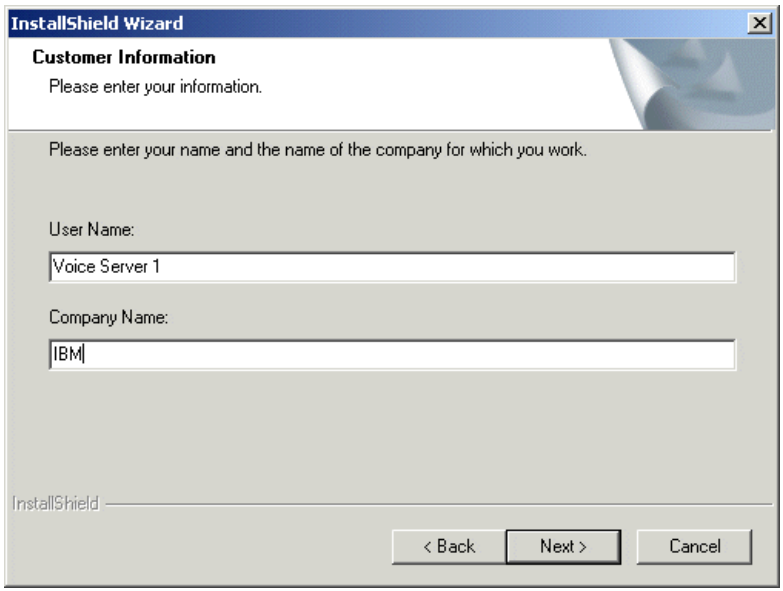


Figure 6-19 Registration window

Step 4: Reboot

Once completed the reboot option appears, as illustrated in Figure 6-20 on page 240. The system must be rebooted to continue. Click **Finish**.

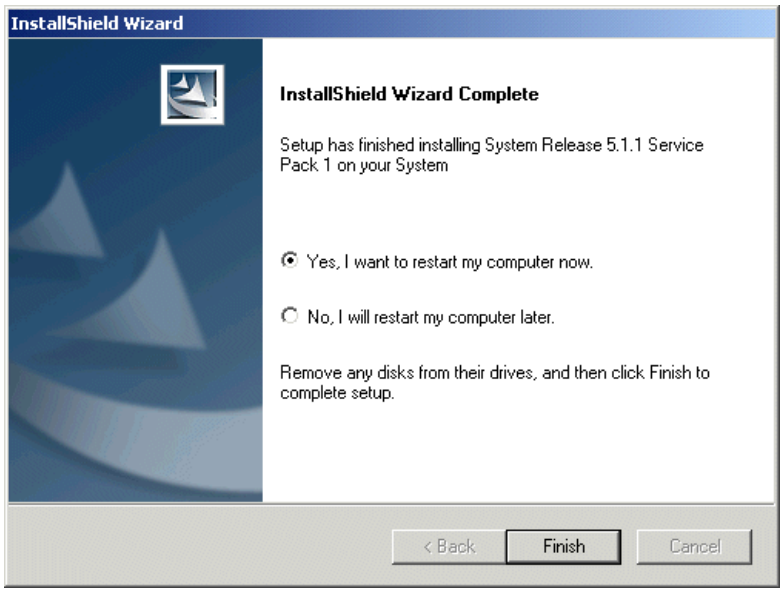


Figure 6-20 Reboot window

6.3.3 Installing GlobalCall Protocols 1.00

In this section, complete the following steps to install the GlobalCall Protocol.

Step 1: Run setup.exe

This is a step-by-step guide to install GlobalCall Protocols 1.00 for Windows. If you have downloaded the GCP 1.00 from http://developer.intel.com/design/telecom/support/, you will first need to unzip it and then run the setup.exe program in the Windows subdirectory, as shown in Figure 6-21 on page 241. Click **Next**.

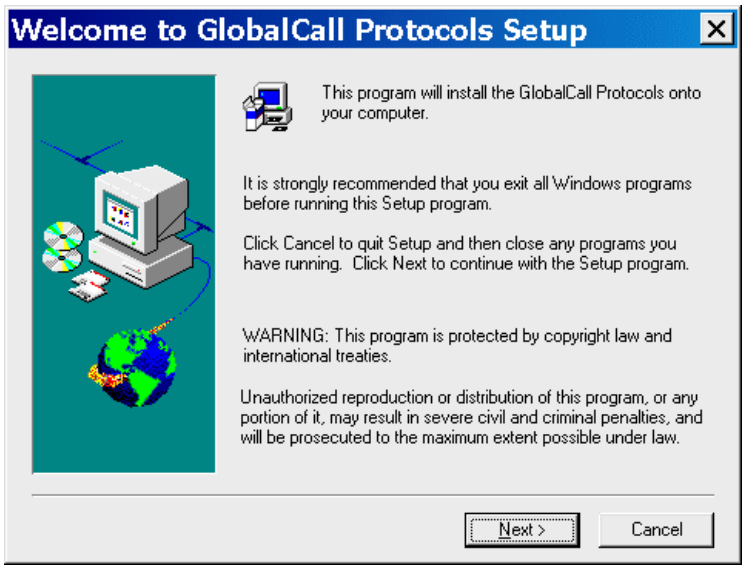


Figure 6-21 Setup window

Step 2: Setup options

There are two options. The *WebSphere Voice Server Version 2.0 - Use with Intel Dialogic Telephony Platform Administrator's Guide*, G210-1263 states that you should do a complete install, meaning all protocols. We tested both the complete and custom installations in our environment. Here we show you how to do a custom install. Click **Custom** as shown in Figure 6-22 on page 242. Then click **Next**.

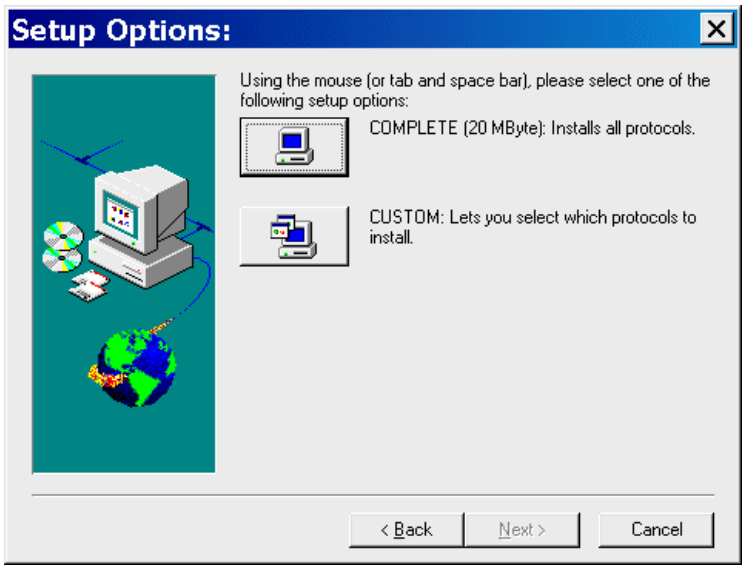


Figure 6-22 Option window

Step 3: Custom selection

Testing was done in North America, so we selected the **North America Protocols** as illustrated in Figure 6-23 on page 243. If in a different region, select the protocol for your country. Click **Next**.

Note: Global Protocols does not include default configuration files for all countries. In this situation, a configuration file needs to be created using your country settings. Consult your telephone specialist for region-specific settings.

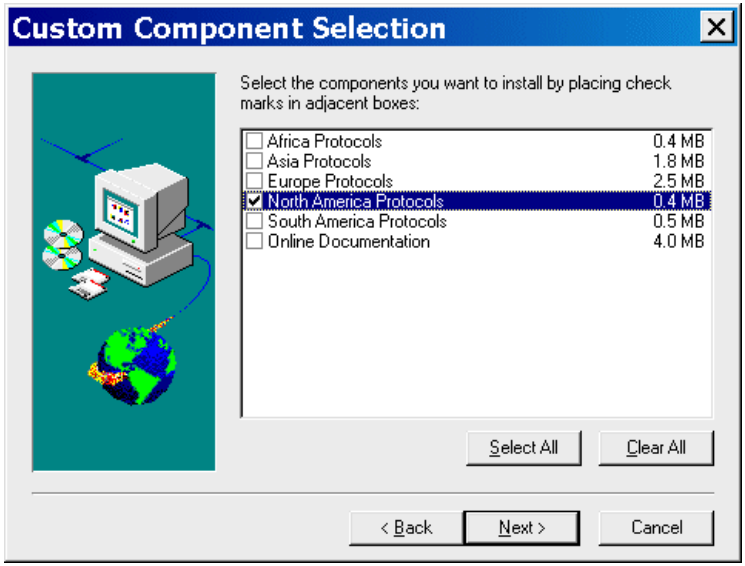


Figure 6-23 Protocol window

Step 4: Country protocols

The North America Protocols had three internal protocols. Choose the appropriate protocols for your region. We selected **North America Analog** and **United States**, as shown in Figure 6-24 on page 244. Click **Next**.

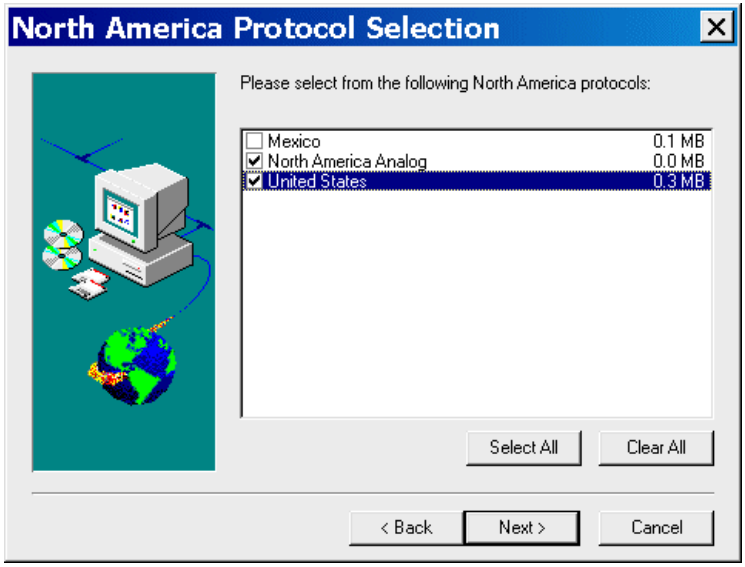


Figure 6-24 North America protocols window

Step 5: Destination location

The default path for the installation is shown in Figure 6-25. We kept the default. Click **Next**.

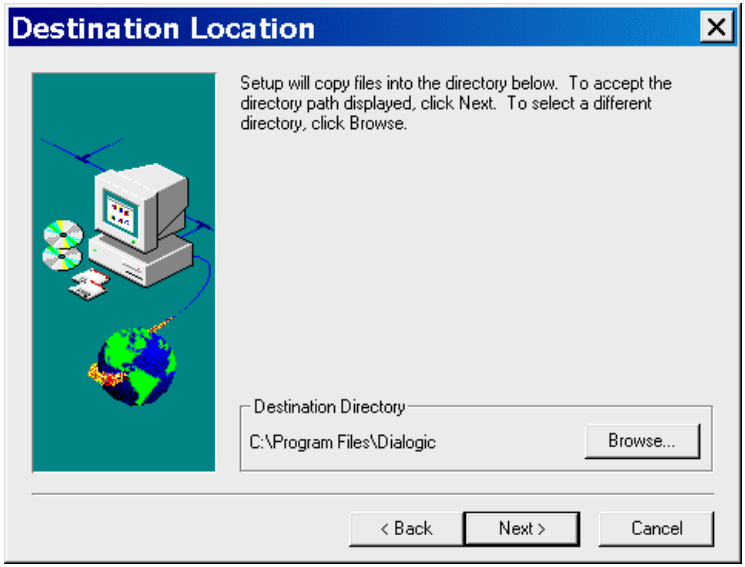


Figure 6-25 Destination window

Step 6: Setup summary

The setup options summary lists your selections as shown in Figure 6-26. Once the necessary parts are confirmed, click **Next** to complete the install.

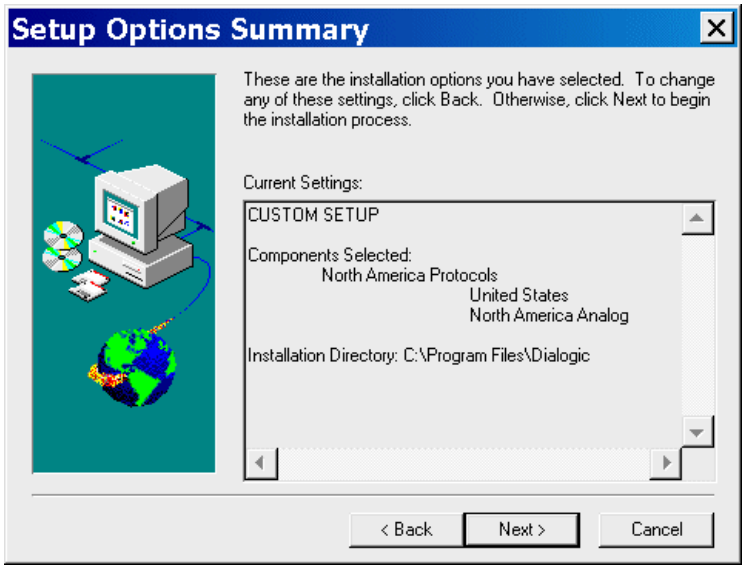


Figure 6-26 Summary window

Step 7: Reboot

Once complete, the Setup Complete window will display with the default setting of Yes to reboot, as shown in Figure 6-27 on page 246. We suggest that you reboot your machine. Click **Next**.

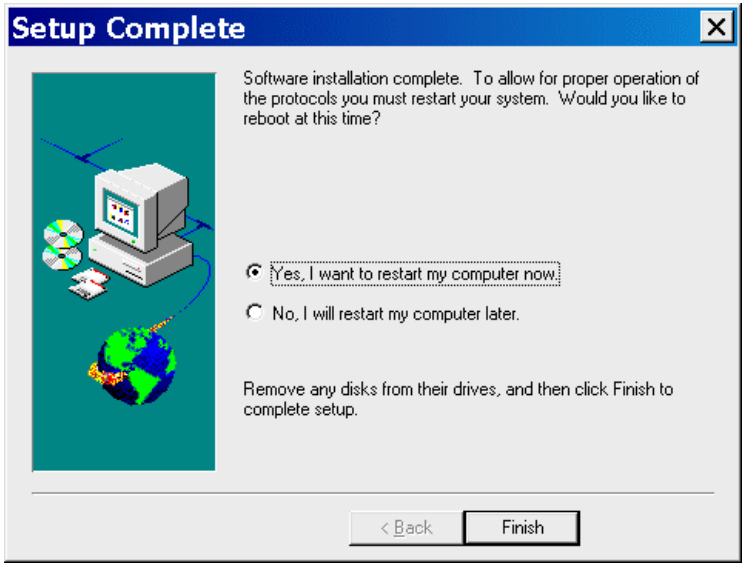


Figure 6-27 Reboot window

6.3.4 Dialogic card installation

Voice Server for the Dialogic environment requires a Dialogic board to interface with the telephony environment. The boards supported for Voice Server are:

- D/120JCT-LS (a 12-port analog board)
- D/240JCT-T1 (a 24-port digital board)
- D/480JCT-1T1 (a dual-span 24-port digital board)
- D/480JCT-2T1 (a dual-span 48-port digital board)
- D/600JCT-1E1 (a dual-span 30-port digital board)
- D/600JCT-2E1 (a dual-span 60-port digital board)

Dialogic has released another card called the D/41JCT-LS. It is a 4-port analog board. Although it is not included in the officially supported card list, we successfully tested it with Voice Server.

This is a step-by-step procedure to install the Dialogic board into your Intel machine. This procedure is for a D/120JCT-LS, the card we used in our scenario.

Step 1: Installing hardware

Power off the machine and unplug the power supply. Install the Dialogic card into a free PCI slot, preferably in the lowest numbered slot. Figure 6-28 on page 247 shows an installed D/120JCT-LS Dialogic card.

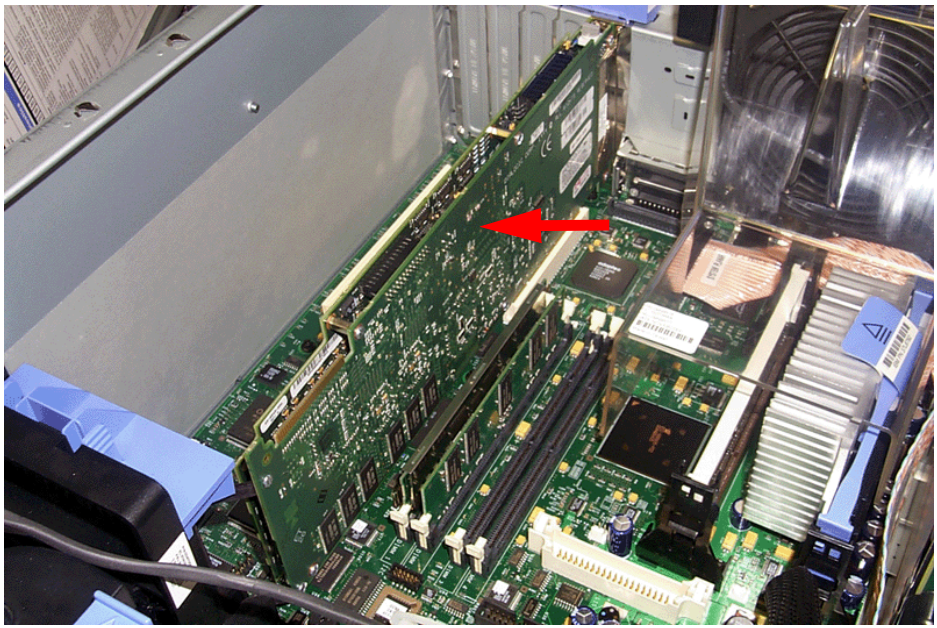


Figure 6-28 Photo of installed Dialogic card

Step 2: Hardware detection

Power on the machine and log in as the administrator. The Hardware wizard should appear indicating new hardware has been found, as shown in Figure 6-29 on page 248. Click **Next**.

Figure 6-29 Hardware wizard

Move through the hardware detection windows until you reach the device driver window, as illustrated in Figure 6-30 on page 249. Windows 2000 will try to determine what drivers match the card. Click **Next**.

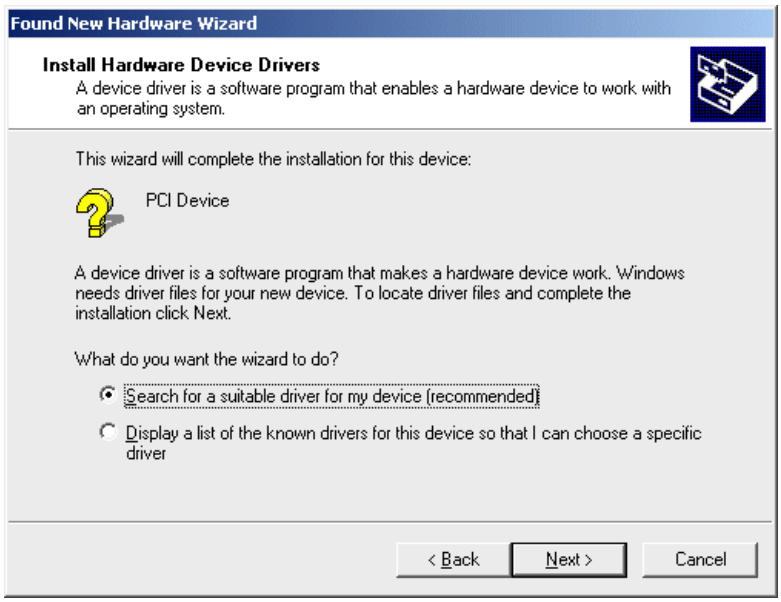


Figure 6-30 Device driver window

The Dialogic software provides a set of drivers. We select the option **Specify a location**, as shown in Figure 6-31. Click **Next**.

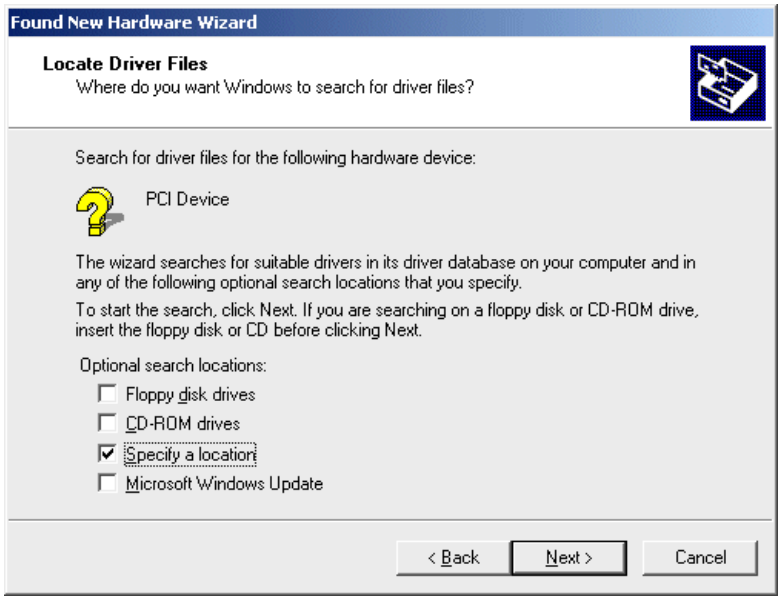


Figure 6-31 Specify location window

We need to manually point to the driver, so click **Browse**, as shown in Figure 6-32.

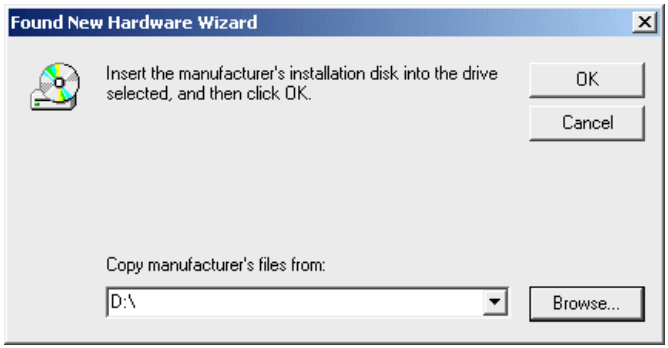


Figure 6-32 Browse window

The drivers are located in the DRVR subdirectory of the installed Dialogic software. In our case this is the default folder. Highlight this folder and click **Open**, as in Figure 6-33.

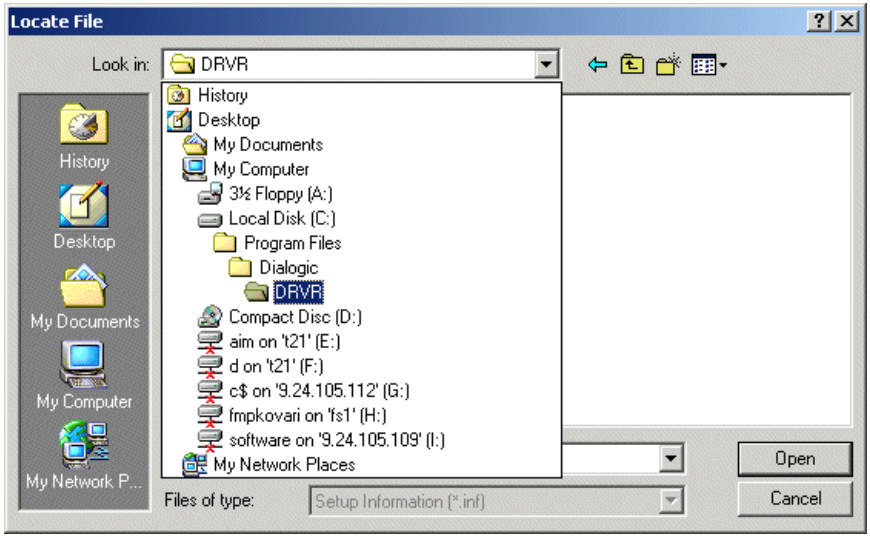


Figure 6-33 Dialogic driver location window

You are presented with a set of drivers. For the D/120JCT card, select the **dlgcsram_nt4.inf** file, then click the **Open** button, as shown in Figure 6-34 on page 251.

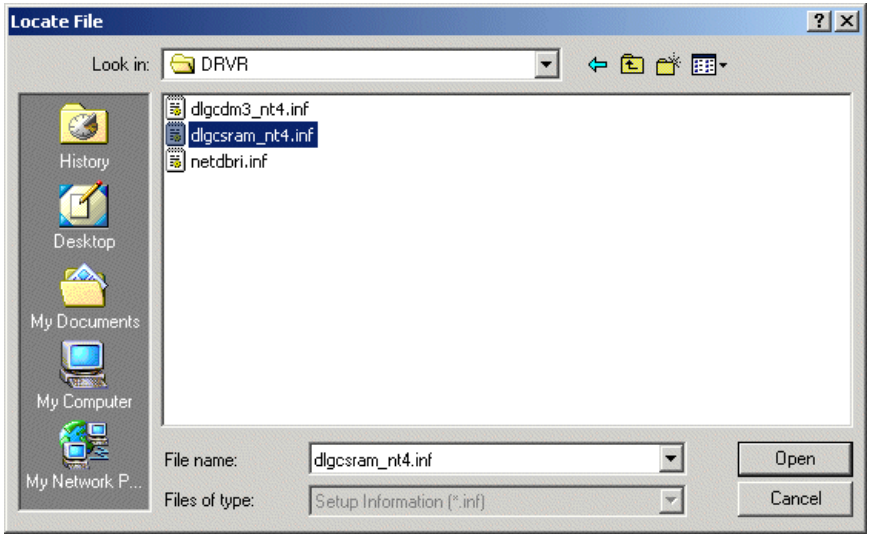


Figure 6-34 Dialogic driver window

A confirmation window will appear, as in Figure 6-35. Click **OK** if correct.

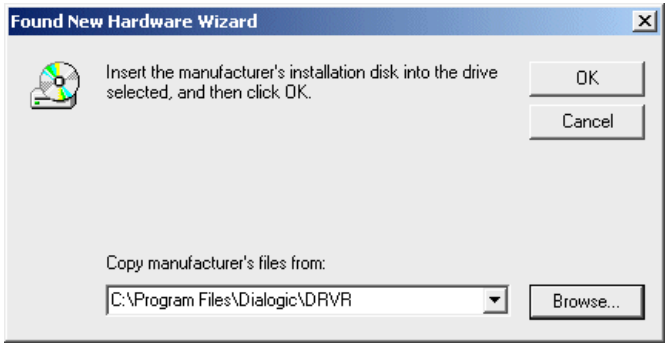


Figure 6-35 Driver confirmation window

The Hardware wizard will indicate the driver it found (Figure 6-36 on page 252). Click **Next**.

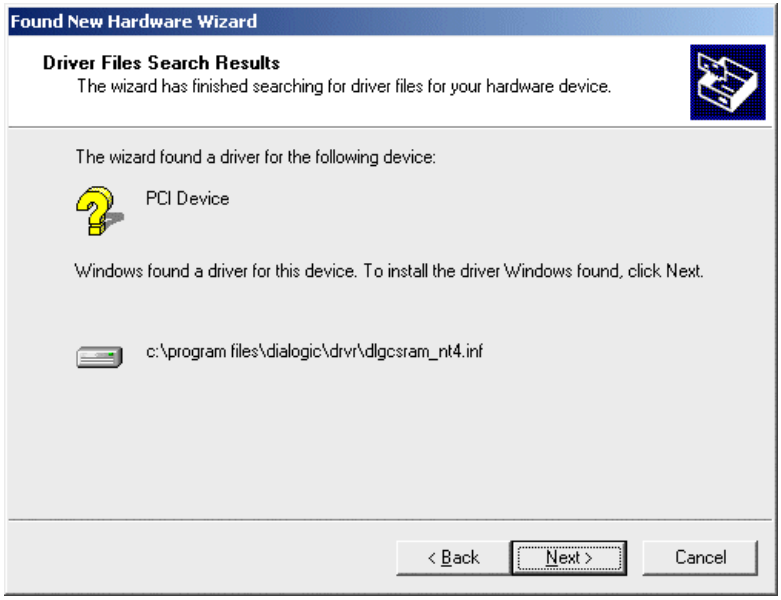


Figure 6-36 Drive found window

The Hardware wizard will present a completed window that looks like Figure 6-37 on page 253. Click **Finish**.

Figure 6-37 Finish window

6.3.5 Dialogic card configuration

Before installing Voice Server, the Dialogic card needs to be configured. These settings can be broken down into two sections, global settings and specific country settings.

To configure the Dialogic card, use the Dialogic Configuration Manager (DCM) in the Dialogic application folder. First the application must connect to the machine specified by the host name given to the machine. This is shown in Figure 6-38.

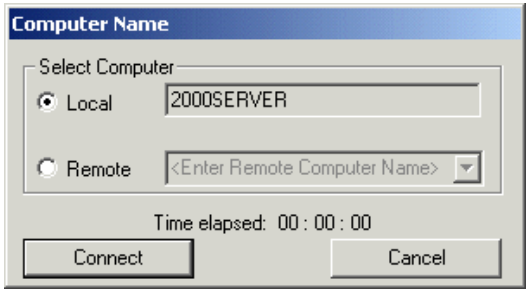


Figure 6-38 DCM connection window

The DCM window will show you the installed Dialogic card and the Time-Division-Multiplex (TDM) Bus. Figure 6-39 shows the DCM window.

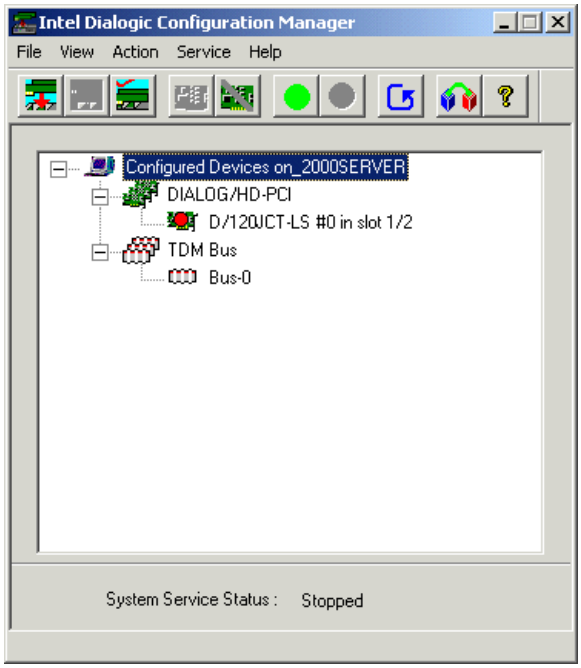


Figure 6-39 DCM window

Right-click the installed card. In this case it is a D/120JCT-LS. Select **Configure Device**, as illustrated in Figure 6-40 on page 255.

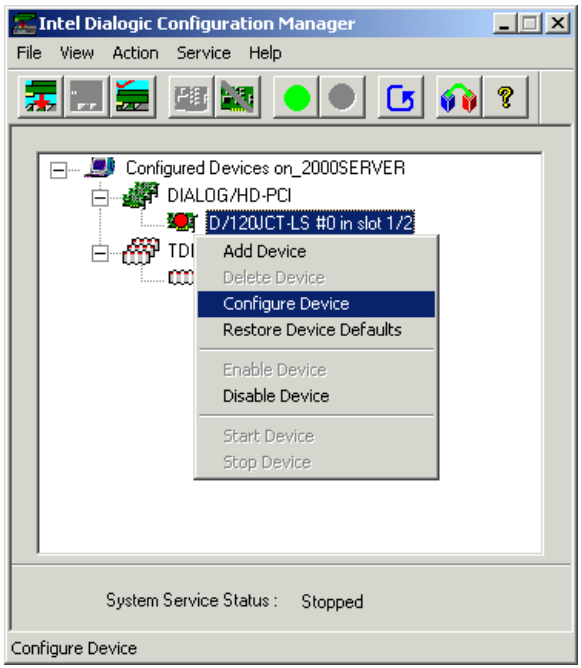


Figure 6-40 Configure Device option window

The properties window for the selected card is displayed. Changes are required on two tabs, Misc and Country. Set the parameter values for the D/120JCT-LS board as shown in Table 6-1.

Table 6-1 DCM settings table

At this tab	Ensure that this parameter	Has this value
Misc	FirmwareFile	d120csp.fwl
	EC_Resource	OFF
	CSPExtraTimeSlot	ON
Country	Country	*your country*

Figure 6-41 shows an updated Misc tab with the correct settings.

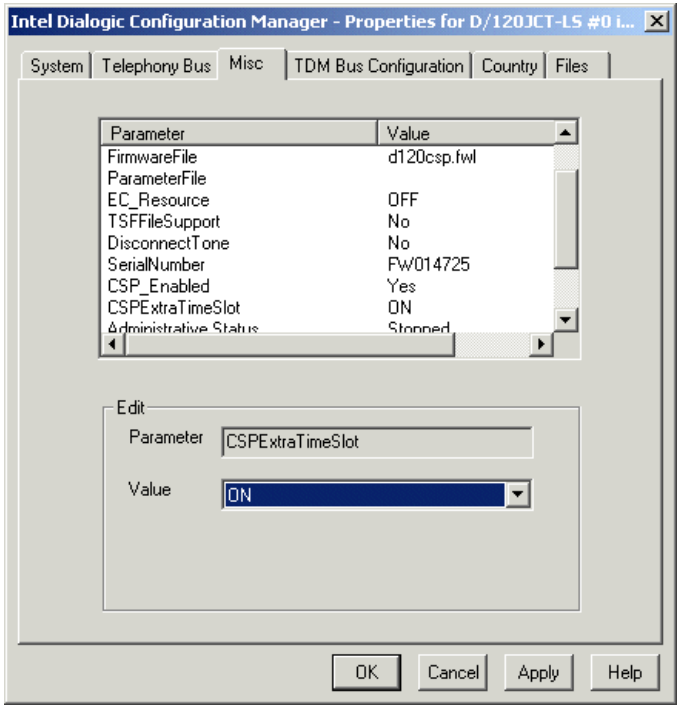


Figure 6-41 Misc parameter window

Click **Apply** to save the settings. Then select the **Country** tab and ensure you have your country selected. Click **OK**, as in Figure 6-42 on page 257.

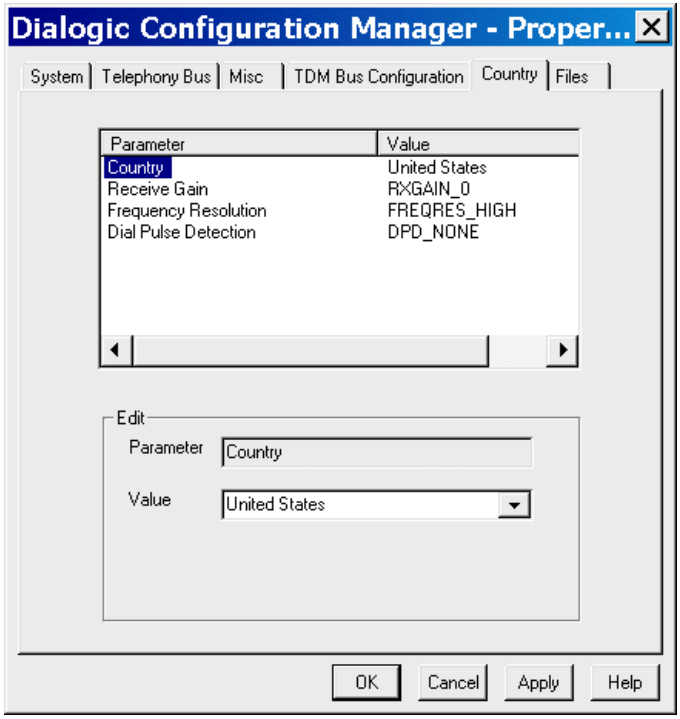


Figure 6-42 Country parameter window

Now right-click the Bus-0 under the TDM Bus and select **Configure Device**, as illustrated in Figure 6-43 on page 258.

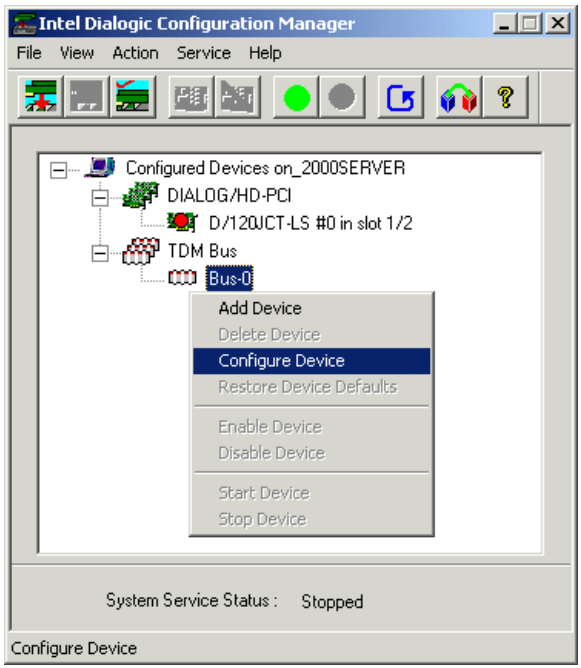


Figure 6-43 TDM Bus selection

The Bus configuration window will appear. One parameter needs to be changed here, as shown in Table 6-2.

Table 6-2 TDM Bus parameter

At this tab	Ensure that this parameter	Has this value
TDM Bus Configuration	TDM Bus Type (User defined)	SCBus

Click **OK** when done, as shown in Figure 6-44 on page 259.

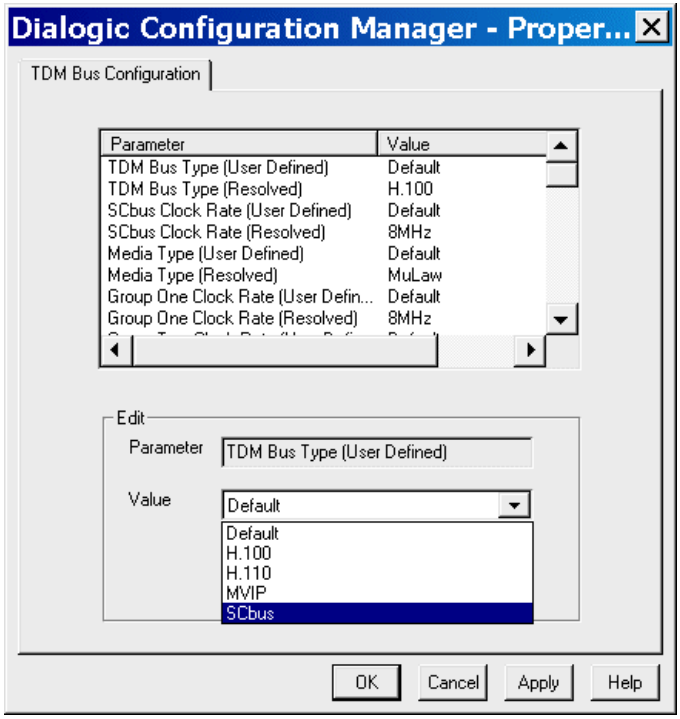


Figure 6-44 TDM Bus configuration window

Start the card to test the configuration by clicking the green circle, as in Figure 6-45 on page 260. Once completed, the card will be active. The card has a green circle on it to indicate this.

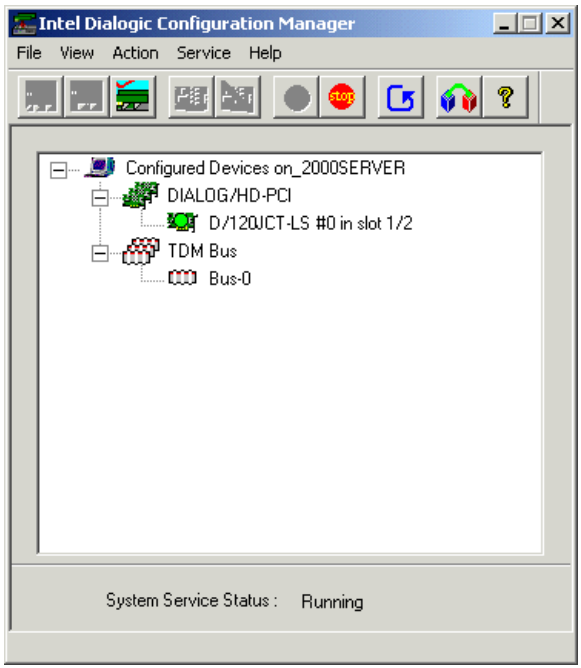


Figure 6-45 Started DCM window

6.3.6 Card configuration parameters for D/120JCT-LS

There are several parameters that need to be changed before the card can function. The first is in the file na_an_io.cdp. This file is found in the Dialogic software installation directory, in our case \Dialogic\cfg.

The prefix of the file name is na for North America. If you installed GlobalCall Protocols for a different country, you will need to edit the xx_an_io.cdp, where the xx is the two-character country code for the country you specified in the Country tab in the DCM properties. You can see a list of two-character country codes in the Appendix of the *WebSphere Voice Server Version 2.0 - Use with Intel Dialogic Telephony Platform Administrator's Guide*, G210-1263.

In this same file, change the assigned values so that they are identical to the statements below for your location. Then save the file.

For all countries

```
$6 timeout for automatic hang-up during speech (outbound) (s) : 0
$13 timeout for automatic hang-up during speech (inbound) (s) : 0
$100 CO_DT1 : nnn nn nnn nn
```

The value *nnn nn nnn nn* represents the frequencies your telephone switch uses for dial tones. Consult your telephony specialist to determine the appropriate values. These values must also be entered into your VVTDefaults file for the *.*.TelChannel.*.dialtoneFrequencies resource. (The values we used were 334 40 459 40.) This file, VVTdefaults, is present once Voice Server is installed.

For Canada and the United States only

```
$31 consider dial tone 1 as busy during speech (in-out) (1=YES, 2=NO) : 1
```

For European countries only

```
$30 detect disconnection on loop drop & reversal (in-out) (1=YES, 0=NO) : 0
$31 consider dial tone 1 as busy during speech (in-out) (1=YES, 2=NO) : 0
```

6.3.7 Testing the card

We tested the card by running a sample application. Make sure that the telephone cables are connected to the card. In our scenario, two analog lines were connected to the Dialogic card. Start the Horoscope program found in the sample directory in the Dialogic folder, as shown in Figure 6-46 on page 262.

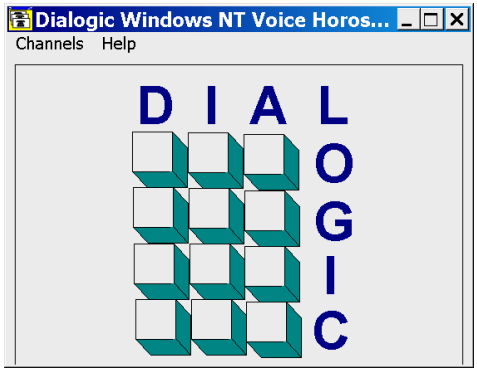


Figure 6-46 Horoscope window

The Horoscope program opens so incoming calls can be received. To do this click **Channels -> Open**, as shown in Figure 6-47.

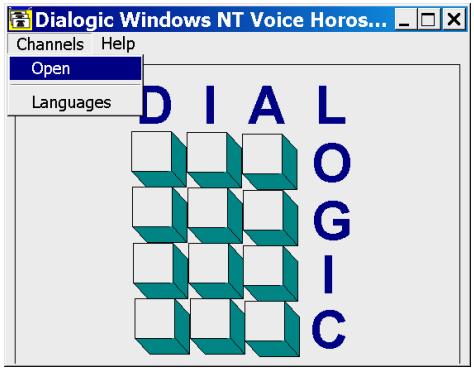


Figure 6-47 Channel opening window

Once the channels are opened, calls can be made. Four lines are ready and waiting. When a call is received, the phone appears off the hook, as shown in Figure 6-48 on page 263.

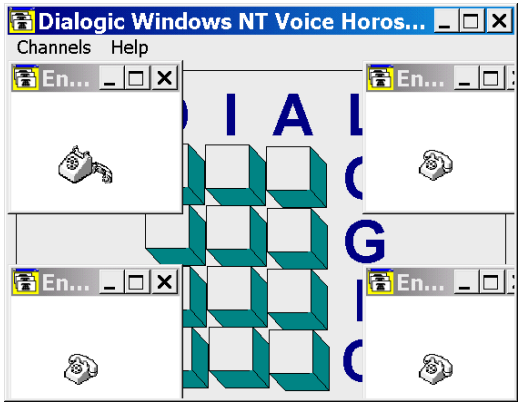


Figure 6-48 Call received window

The user is required to enter in data via DTMF. If successful, the display will change to reflect the input, as illustrated in Figure 6-49.

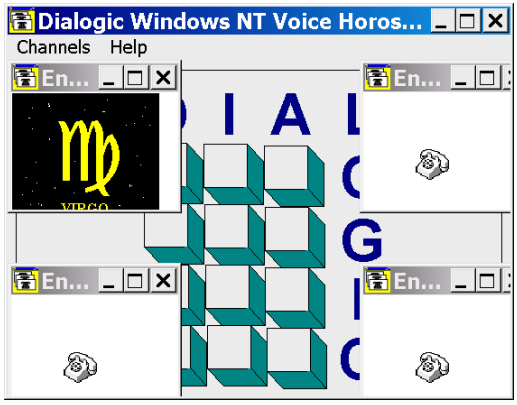


Figure 6-49 Data entered window

6.3.8 Voice Server prerequisite software

There are several requirements that need to be met before the actual Voice Server is installed. Ensure that the user ID logged on has administrator rights. Also, read the section on language support before proceeding.

6.3.9 Voice Server installation

Our choice of the stand-alone configuration has both server and client running on the same machine. In our case we had an executable file. You should have a CD labeled WebSphere Voice Server for Windows 2000 - Use with the Intel Dialogic telephony platform.

Step 1: Start install

Either run the executable file or insert the CD. You will be presented with a window like the one shown in Figure 6-50. Click **Next**.

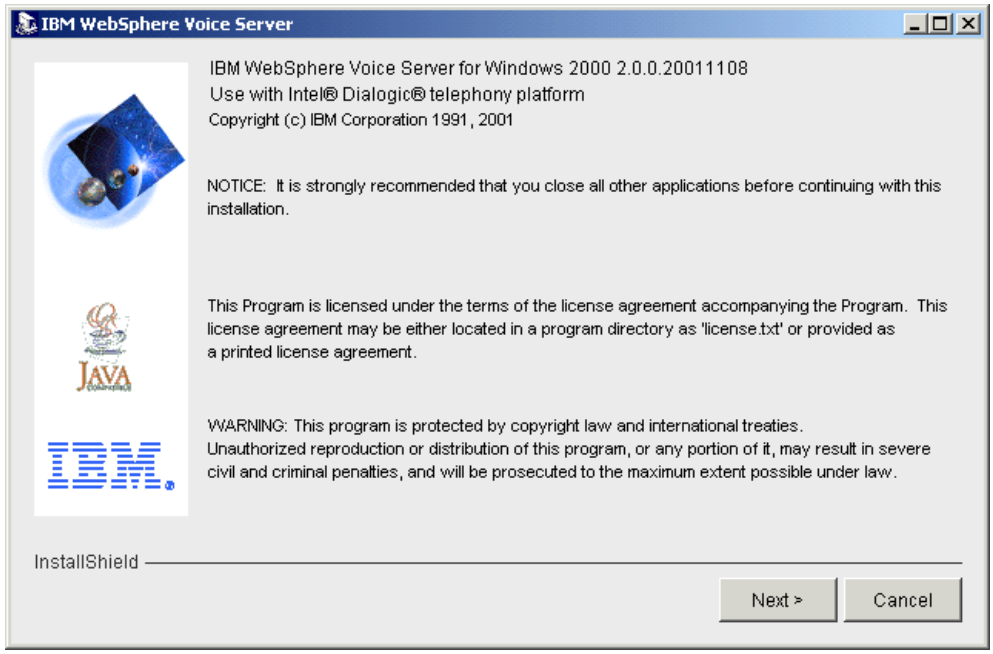


Figure 6-50 Voice Server installation window

Step 2: Setup options

Voice Server has three different configurations. Figure 6-51 on page 265 displays the three options, as follows:

1. If this is just a voice browser client, then select the **Intel Dialogic Interface** option.
2. If this is just the server with the Dialogic cards, then select the **Speech Interface** option.
3. And if both server and browser clients need to run on one machine, select **Both**.

Since we have chosen a stand-alone configuration, both client and server are selected, as shown in Figure 6-51. This means both the server and browser client software will be installed on one machine with the Dialogic card. When you have chosen, click **Next**.

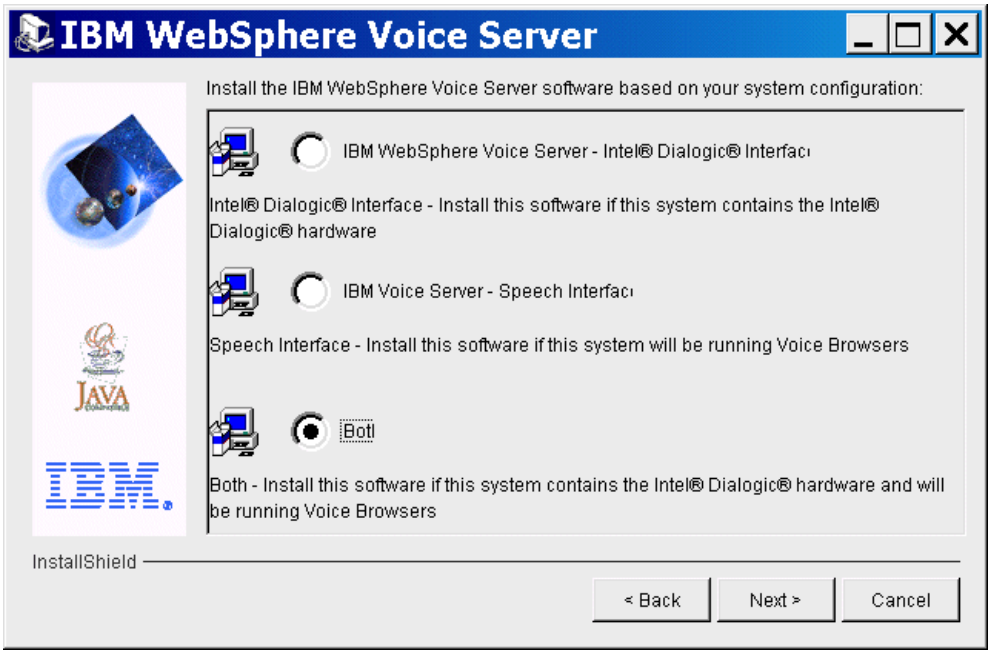


Figure 6-51 Voice Server options

Step 3: Prerequisite check

Voice Server will check for the necessary prerequisites before it will continue. If one does not pass, the installation will fail. Correct the problem and try again. If all pass, click **Next**, as shown in Figure 6-52 on page 266.

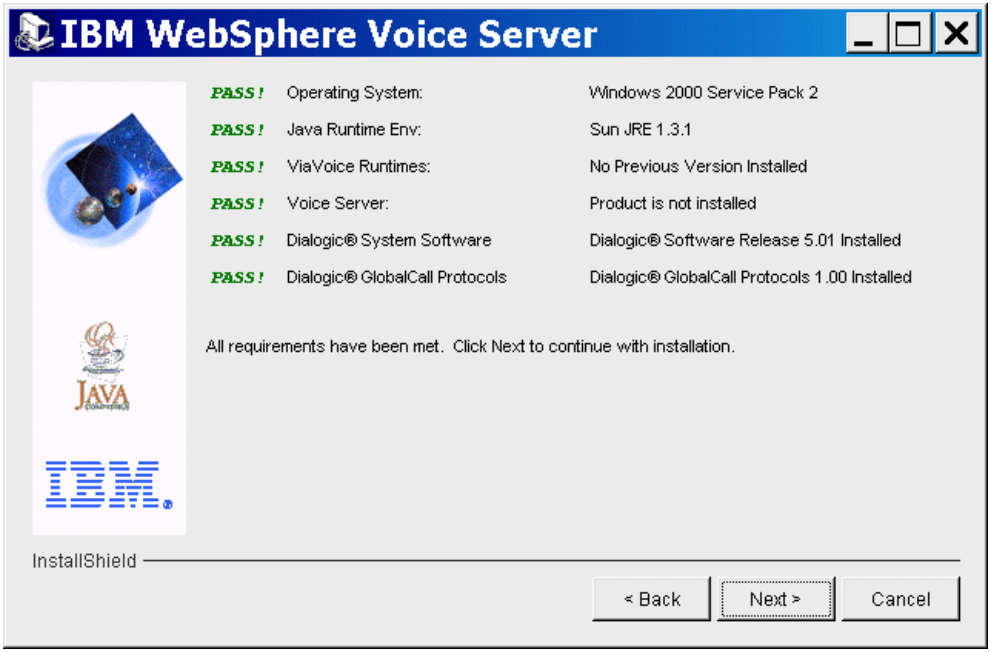


Figure 6-52 Prerequisite window

Step 4: License agreement

The license agreement appears. Read and accept it as in Figure 6-53 on page 267. Click **Next**.

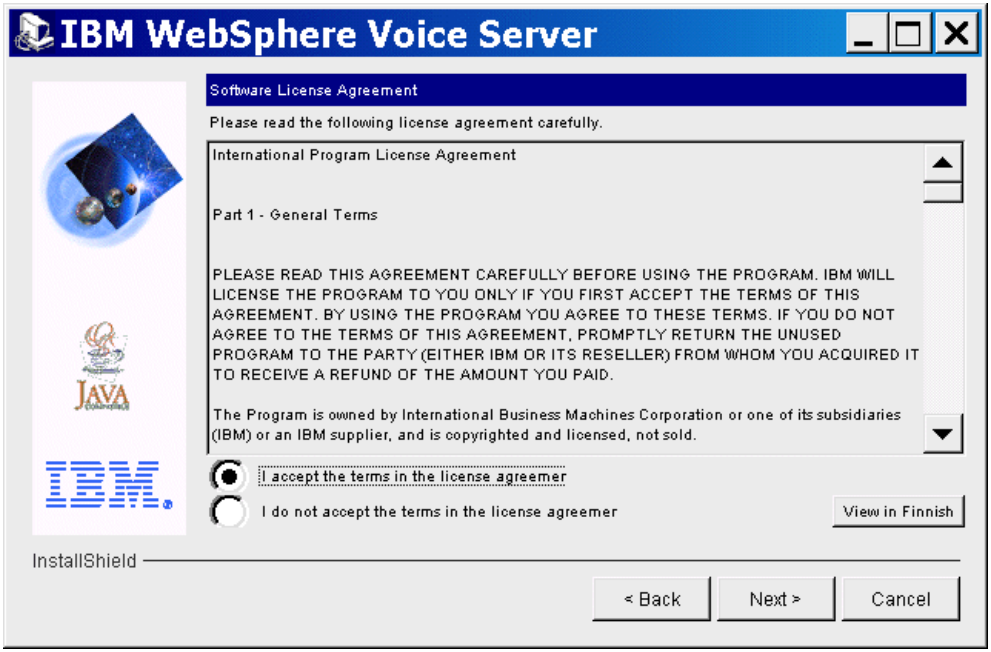


Figure 6-53 License agreement

Step 5: README file

The README file is displayed, as illustrated in Figure 6-54 on page 268. Read for any last-minute installation changes. Click **Next**.

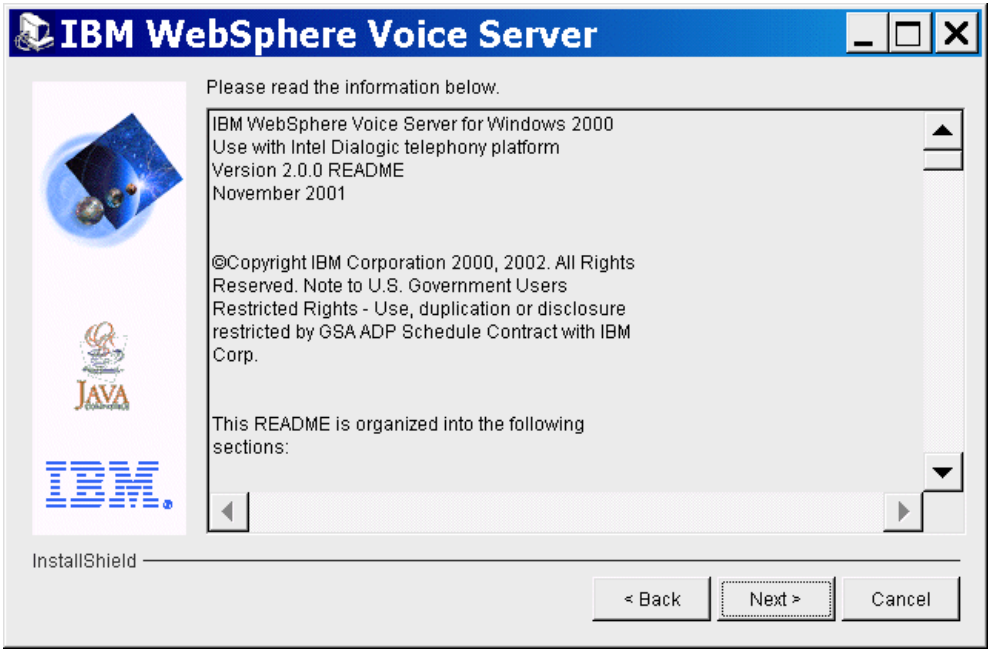


Figure 6-54 README window

Step 6: Creation of the destination directory

Voice Server needs to be installed into a directory. A default is provided. We took this option. Click **Next** after setting the destination directory, as shown in Figure 6-55 on page 269.

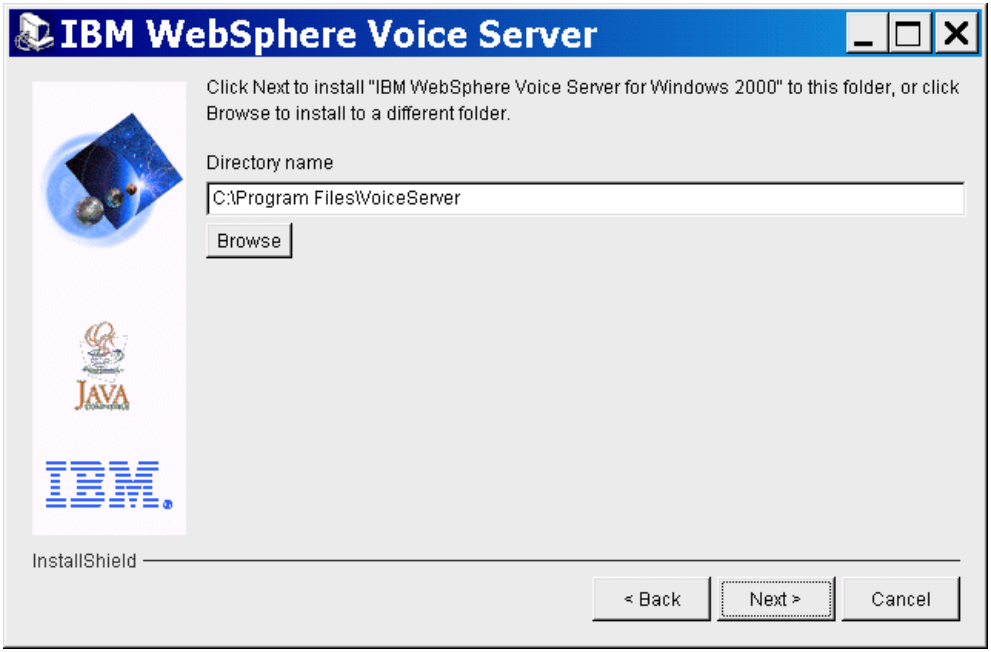


Figure 6-55 Destination directory window

A warning appears advising you that the default directory does not exist. Click **Yes** to create it, as in Figure 6-56.

Figure 6-56 Directory creation

Step 7: Language support

This window displays all the support languages for the Dialogic environment. Within this list you will see three languages highlighted: US English, UK English, and Simplified Chinese. These are the languages we installed during the language support components step. Click **Next**, as in Figure 6-57 on page 270.

Note: Once Voice Server installed, additional languages cannot be installed afterwards, that is, on top of the server.

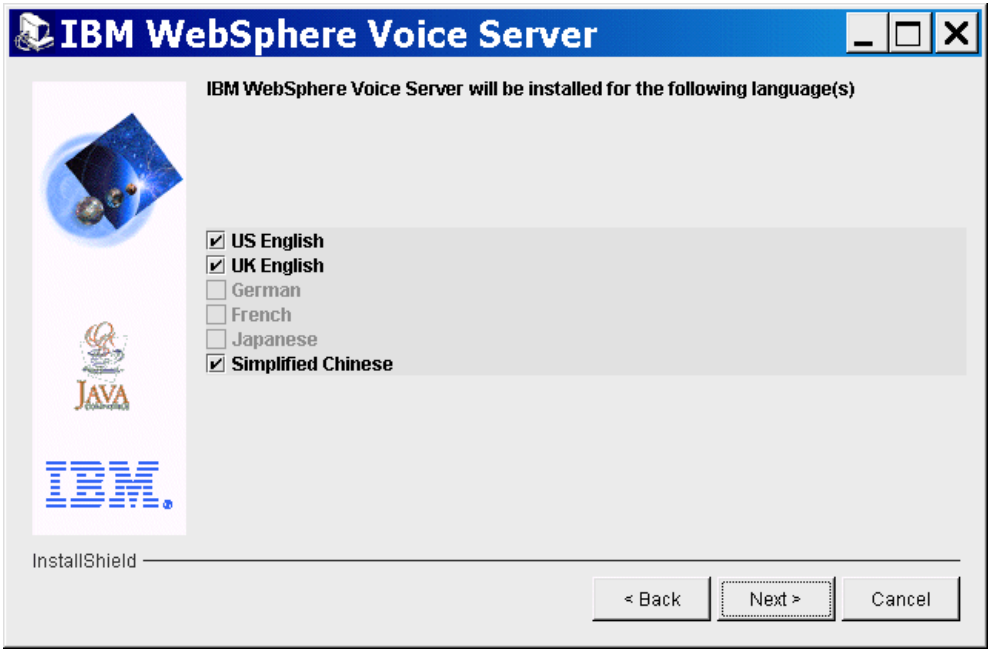


Figure 6-57 Language window

Step 8: Voice Server installing

For each language, Voice Server installs that version of itself. In our scenario, US English, UK English, and Simplified Chinese versions of Voice Server were installed. Figure 6-58 on page 271 shows US English, and Figure 6-59 on page 272 shows UK English.

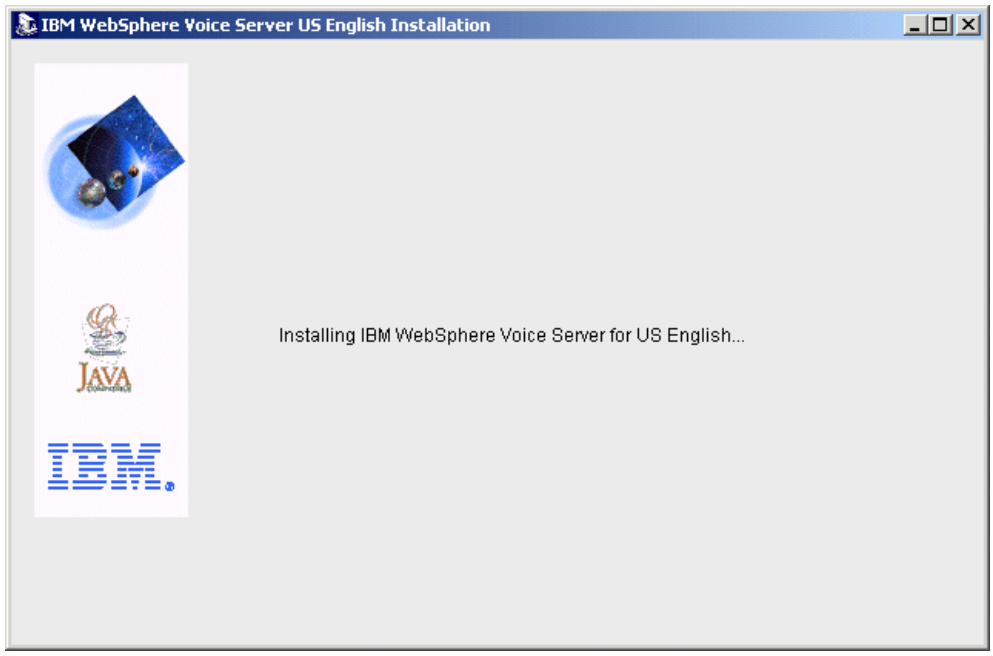


Figure 6-58 US English Voice Server

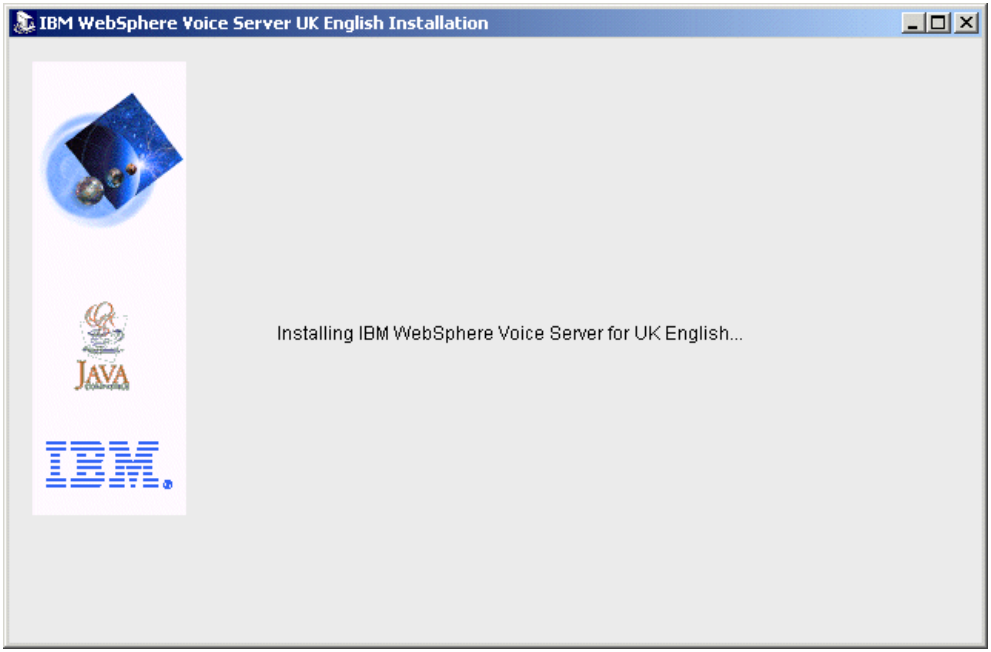


Figure 6-59 UK English Voice Server

Step 9: Completion

Figure 6-60 on page 273 shows the completion of the Voice Server installation. Click **Finish** to wrap it up. At this point you should reboot your machine before continuing.

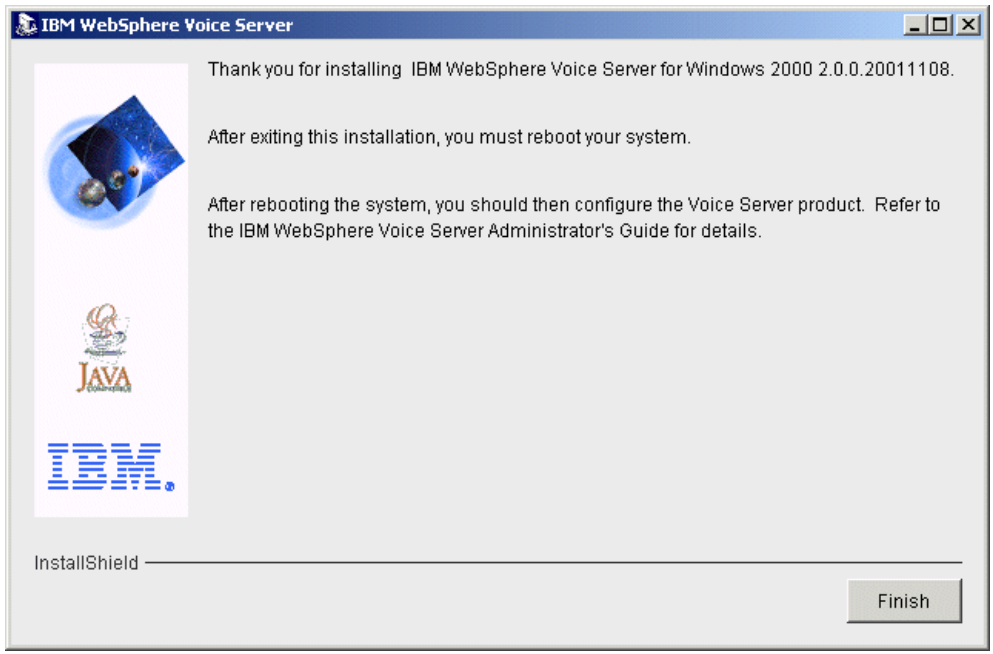


Figure 6-60 Voice Server completion window

6.4 Configuring Voice Server

Based on your deployment environment, you may need to modify the Voice Server configuration. There are two configuration files that may need to be modified. They are the sysmgmt.properties and VVTDefaults files. These will have to be modified if:

- Your environment does not match the preconfigured or default values assigned in the sysmgmt.properties configuration file or the VVTDefaults file.
- You want to modify or scale your telephony environment by adding additional VoiceXML browsers.
- You want to run other or additional VoiceXML applications.

The following steps are required to reconfigure a Voice Server:

1. Modify the configuration and VVTDefaults files.
2. Validate the configuration file.
3. Force a system update.
4. Check the VoiceXML Browser status.
5. Run VoiceXML applications.

6.4.1 VVTDefaults configuration file

The Voice Server's Dialogic connection environment is configured by means of parameters specified in the file called VVTDefaults. In our scenario, this is located in c:\Program Files\Voice Server\cfg. The four default configuration files provided are for each of the supported Dialogic cards. You must create the VVTDefaults file from one of the following:

- VVTDefaults.d120jct
- VVTDefaults.d240jct
- VVTDefaults.d480jct
- VVTDefaults.d600jct

Locate the file whose name matches the model of your installed Dialogic board. Copy it and rename the copy to VVTDefaults. Our tested system used the D/120JCT board, so we made a copy of the file VVTDefaults.d120jct and renamed the copy VVTDefaults.

VVTDefaults is a flat text file. Each line contains an entry, which must be placed at the beginning of the line. Comments will start with a pound sign symbol (#) or two forward slashes (//).

An entry or specification in VVTDefaults is called a resource. Example 6-1 shows the default 120JCT file installed with Voice Server.

Example 6-1 Default 120JCT file

```
##########################################################################
## Licensed Materials - Property of IBM
##      5724-B59
## (C) Copyright IBM Corp. 2000, 2001 All Rights Reserved
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
##########################################################################


# the list of boards, one of
# d120sc-bargein -- analog, usable channels: 0,2,4,6,8,10
# d120jct -- analog, usable channels: 0..11
# d240sc-bargein -- T1 CAS (usable channels 0..11) or PRI (usable channels
0..10)
# d240jct -- T1 CAS, usable channels 0..23
# d240jct-pricsp -- T1 PRI, usable channels 0..10
```

```
# d480jct -- dual T1 CAS, usable channels 0..47
# d480jct-pricsp -- dual T1 PRI, only on span usable (channels 0..22)
# d600jct -- dual E1 CAS, usable channels 0..59
# d600jct-pricsp -- dual E1 PRI, only one span usable (channels 0..30)
*.*.DialogicApi.boards: d120jct

# 0 for boards *sc-bargein, d240jct-pricsp
*.*.DialogicApi.useCSP: 1

# jct or sc
*.*.TelDevice.boardType: jct

# Record Buffer Size
*.*.TelChannel.*.recordbuffersize: 1536

# t1, e1 or analog
*.*.TelDevice.model: analog

*.*.TelDevice.oem: Dialogic

# Setting this value to zero will disable the EC resource.
# The D/240JCT card (and all T1/E1 SC cards) need this value to be zero.
*.*.DialogicApi.useECResource: 1

# pri, cas or tr (tip-ring for analog cards)
*.*.TelDevice.protocolFamily: tr

# isdn for PRI, winkstart, loopstart or opx for CAS,
# loopstart for analog cards
*.*.TelChannel.protocol: loopstart

# the frequencies of cleardown and dial tones are defined as
#
#  f1 delta1 f2 delta2
```

```
#
#  or
#
#  f1 delta1 0 0
#
# for single frequency tones
#
# cleardown has to match the tone received on disconnect
*.*.TelChannel.*.cleardownFrequencies: 350 40 440 40

*.*.TelChannel.*.dialtoneFrequencies: 350 40 440 40        1
*.*.*.bargeIn: 1

*.*.TelChannel.*.streamAudio: 1
*.*.TelChannel.*.streamDigits: 1

*.*.*.useConsole: no

# This property will allow exceptions to be displayed on the desktop
*.*.*.postMortemDebug: yes

# This property turns on echo cancellation
*.*.telChannel.*.echoCancellation: 1            2

# This property sets the default log file directory
*.*.logdir: logs

# Uncomment the following statements to generate debug log files
#*.*.TelDevice.debugLevel: trace
#*.*.TelAudio.debugLevel: trace
#*.*.DialogicApi.debugLevel: trace
```

The key parameters (bolded in Example 6-1 on page 274) are:

1 `*.*.TelChannel.*.dialtoneFrequencies: 350 40 440 40`

2 `*.*.telChannel.*.echoCancellation: 1`

The default values of `350 40 440 40` represent the frequencies the telephone switch uses for dial tones. Consult your telephony specialist to determine the appropriate values. These values must also be entered into your VVTDefaults file for the *.*.TelChannel.*.dialtoneFrequencies resource.This file, VVTdefaults, is present once Voice Server is installed.

Note: Debug options are at the end of the file. When active, they generate additional logging information to assist with problem diagnosis.

6.4.2 System management file

You can control the behavior of Voice Server by means of system management parameters in the file sysmgmt.properties. When Voice Server is installed a sample sysmgmt.properties file is included, regardless of it being the Voice Server or client voice browser machine. The default file is configured to support a Dialogic telephony environment and one VoiceXML browser.

You will notice that there are two types of configuration parameters:

- General Voice Server
- VoiceXML browser specific

These are documented in the *WebSphere Voice Server Version 2.0 - Use with Intel Dialogic Telephony Platform Administrator's Guide*, G210-1263.

The configuration file documents all possible parameters in the comment sections of the file (lines whose first character is #). Only those parameters that must be explicitly assigned (that is, parameters that are required and have no default value) are required in this file. Review the parameters in more detail to understand what they do.

The file is located in the Voice Server subdirectory, as shown in Figure 6-61 on page 278.

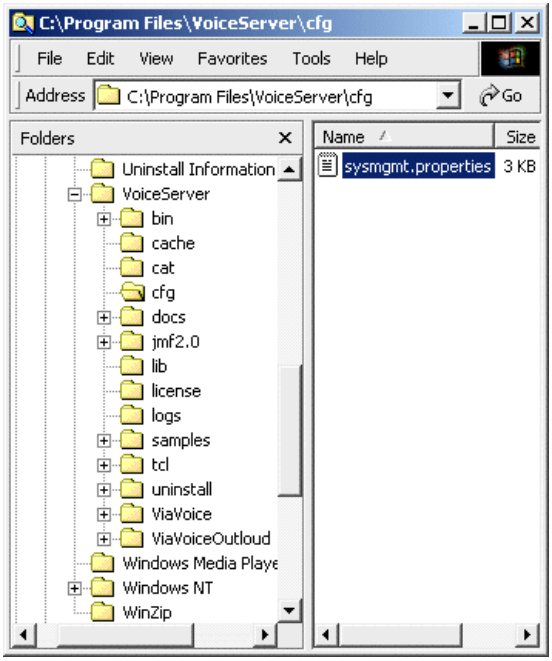


Figure 6-61 System management file location

Example 6-2 shows the default sysmgmt.properties file. It is set up to support a Dialogic card and one VoiceXML browser.

Example 6-2 System management file

```
################################################################################
## Licensed Materials - Property of IBM
##      5724-B59
## (C) Copyright IBM Corp. 2000, 2002 All Rights Reserved
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
################################################################################

# This file contains the IBM WebSphere Voice Server Version 2.0
# System Management configuration parameters.
# The file defines 2 types of configuration parameters:
# 1. General
# 2. VoiceXML Browser
```

```
#------------------------------------------------------------------------
# General Configuration Parameters
#
# ALARM_FILE_MAX_SIZE
# AUTO_START_ON_INIT
# BROWSER_CACHE_MAXSIZE
# BROWSER_CACHE_PATHNAME
# BROWSER_DIRECTORIES_KEPT
# HTTP PROXY
# LOG_FILE_MAX_SIZE
# NUMBER_INBOUND_BROWSERS
# SYS_MGMT_OUTPUT_PATHNAME
#------------------------------------------------------------------------
ALARM_FILE_MAX_SIZE=1
AUTO_START_ON_INIT=false
BROWSER_CACHE_MAXSIZE=1
BROWSER_CACHE_PATHNAME=cache
LOG_FILE_MAX_SIZE=1
NUMBER_INBOUND_BROWSERS=1
SYS_MGMT_OUTPUT_PATHNAME=logs
#------------------------------------------------------------------------
# VoiceXML Browser Configuration Parameters
#
# NOTE: The "n" in INBOUND_*n parameters identifies the VoiceXML browser
# instance. Valid values are 0 through (NUMBER_INBOUND_BROWSERS - 1) and
# correspond to the VoiceXML browser naming convention vvi0"n".
#
# INBOUND_BARGE_INn
# INBOUND_CALLED_NUMBERn
# INBOUND_DUPLEXn
# INBOUND LOCALEn (*)
# INBOUND_SITEn
# INBOUND_TIMEOUTn
# INBOUND_URLn
```

```
#
# (*) NOTE: Valid locales are:
#
# fr_FR for French
# de_DE for German
# ja_JP for Japanese
# zh_CN for Simplified Chinese
# en_GB for UK English
# en_US for US English
--------------------------------------------------------------------------
INBOUND_URL0=file:///<install-directory>/samples/<locale>/AudioSample/AudioSamp
le.VXML
```

6.4.3 Specific parameters

The following sections explain certain parameters.

AUTO_START_ON_INIT

The VoiceXML browsers will not start automatically when Voice Server is powered up because the preconfigured value of the AUTO_START_ON_INIT parameter is false. Edit the sysmgmt.properties file and set the AUTO_START_ON_INIT parameter to true to start the VoiceXML browsers automatically when the Voice Server is powered up.

CALLMANAGER_HOST

The CALLMANAGER_HOST parameter in the sysmgmt.properties points a voice browser client to the Voice Server. When used, ensure that it correctly identifies the host name or IP address of the Intel Dialogic platform server.

Note: Ensure that the parameter is *not* defined on any computer equipped with a Dialogic board. Failure to do so will disable the Dialogic Telephony connection from the Voice Server to the Dialogic cards.

NUMBER_INBOUND_BROWSERS

The parameter NUMBER_INBOUND_BROWSERS specifies the number of browsers you have running to handle inbound calls. For each inbound browser, there must be a corresponding INBOUND_URL. This tells the browser where the VoiceXML file is located.

Note: If the number specified in the NUMBER_INBOUND_BROWSERS does not match the total number of INBOUND_URLs, Voice Server will not start.

INBOUND_URL

When the voice browser is started, the application it runs is set by the INBOUND_URL parameter. The default setting points to the sample application called AudioSample.VXML. This is a simple VoiceXML application that prompts the caller to record some speech and then plays the recorded audio back to the caller. It is located in < install-directory>/samples/< locale> /AudioSample.VXML, where < locale> is one of the following five-character language and country codes:

- de_DE for German
- fr_FR for French
- ja_JP for Japanese
- zh_CN for Simplified Chinese
- en_GB for UK English
- en_US for US English

Note: The URL must not be on a network drive. Any loss of the network drive will cause the voice browser to fail.

If you configured Voice Server in a client/server configuration, you should modify the sample application AudioSample.VXML to indicate which client's browser is accepting a call. You can do this by adding, for example, a prompt to say "This is the client one". This scheme can be used to verify that each client is successfully communicating with the server. If you installed VoiceXML browsers on the server, you should confirm that those browsers also accept calls.

INBOUND naming convention

The INBOUND_*n parameters are configured using a naming convention that is applied to the parameter names for VoiceXML browsers. Table 6-3 illustrates the naming convention.

Table 6-3 Browser naming convention

VoiceXML Browser vvi00	**VoiceXML Browser vvi01**	**VoiceXML Browser vvi02**	**VoiceXML Browser vvi0n**
INBOUND_BARGE_IN *0*	INBOUND_BARGE_IN *1*	INBOUND_BARGE_IN *2*	INBOUND_BARGE_IN *n*
INBOUND_DUPLEX_IN*0*	INBOUND_DUPLEX_IN*1*	INBOUND_DUPLEX_IN*2*	INBOUND_DUPLEX_IN*n*
INBOUND_URL*0*	INBOUND_URL*1*	INBOUND_URL*2*	INBOUND_URL*n*

6.4.4 Validating the configuration file

A system management component is included with Voice Server. This provides basic system management and system administration functions. It is responsible for starting, stopping, and monitoring VoiceXML browsers. It also provides logging and error data in the SM log and alarm files. There is one system management component on every computer on which Voice Server components are installed.

To validate the sysmgmt.properties configuration file before you start up the VoiceXML browsers, issue the following command from a Windows command prompt:

```
SMConfig
```

Each time SMConfig detects an invalid parameter in the configuration file, it will display an error message on the console and stop running. Correct any errors and re-run SMConfig until a No Errors status message is returned to the console, indicating that the configuration file passed all syntax validation.

Note: Only after the system reboots and the Voice Server service is running in Windows can you use the system management component to start the VoiceXML browsers.

6.4.5 Using Voice Server commands

Once the system management component is started and the VoiceXML browsers are operational, you can monitor their operation by means of system management commands.

Note: You must log on with an Administrator ID at each client computer in order to use these commands for VoiceXML browsers running on that computer.

The system management commands are listed in Table 6-4.

Table 6-4 System management commands

Command	Description
vvsm start	Starts all configured VoiceXML browsers. This forces the system to read the sysmgmt.properties configuration file; it is ignored if the VoiceXML browsers are already running.
vvsm getstatus	Returns the status of all VoiceXML browsers running on this computer.
vvsm locale	Returns a list of installed language locales.

Command	Description
`vvsm help`	Displays a list of all available commands with a brief description of each.
`vvsm stop now`	Requests an immediate stop to all VoiceXML browsers. This request will not wait until the current calls end before shutting down the VoiceXML browsers. It will terminate all ongoing calls immediately.
`vvsm stop`	Requests all VoiceXML browsers to gracefully shut down. This request is asynchronous (that is, the command ends before the VoiceXML browsers have stopped). Browsers that are in the “received call” state will not terminate until the call dialog ends, or a 10-minute timeout is reached.

6.5 The final test

Once the parameters have been set, checked and verified, reboot the machine. The machine will start up with Dialogic functionality and Voice Server running. Use the system management commands to verify that Voice Server is waiting for the call status, as shown in Figure 6-62 on page 284.

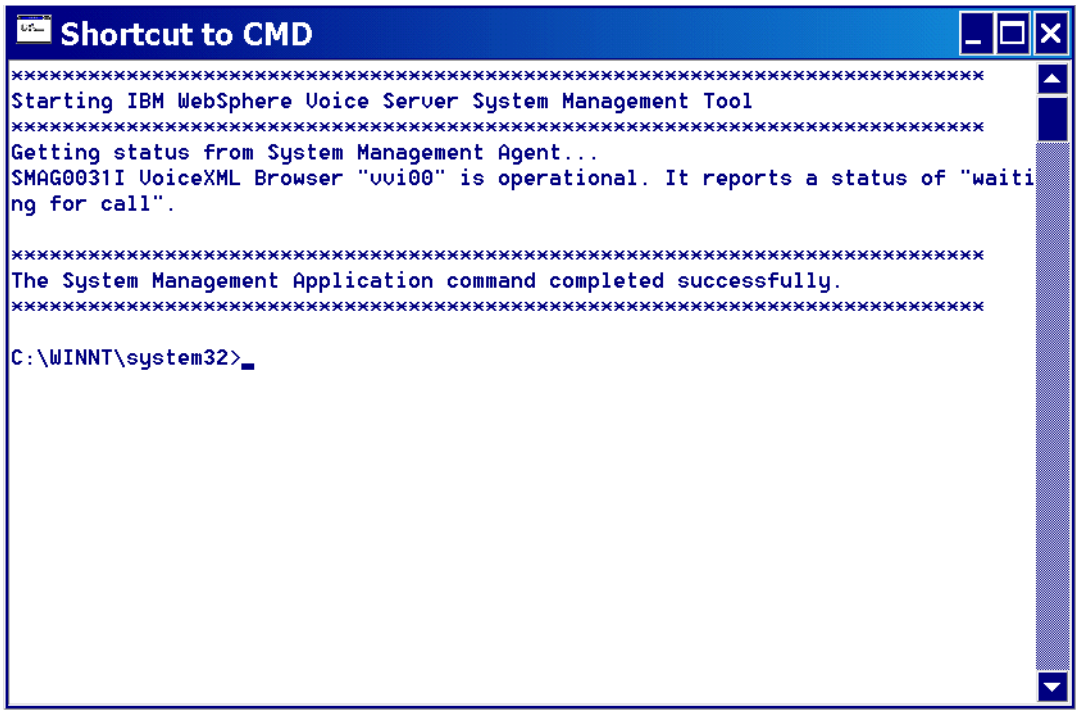


Figure 6-62 System management window

Place a call to Voice Server. You should hear your application running!

6.6 Distributed installation

Both the mixed and full distribution implementations of Voice Server have the benefit of load balancing. In each solution, a Voice Server telephony machine would accept inbound calls and then distribute them to Voice clients.

In our testing, we had several client machines connect to the Voice Server telephony machine. A client is determined at installation time by selecting the **Voice Server Speech Interface** option as in Figure 6-4 on page 228. Once installed, the client is then configured using the system management file to point to the Voice Server machine. This is done by setting the CALLMANAGER_HOST parameter using either the name or IP address of the telephony server (the computer where the Dialogic board is installed).

Note: This parameter is set only on computers configured as clients, not on the telephony server. If it is used on the telephony server, it will fail to connect with the Dialogic cards. The parameter sets the machine to a client regardless of having Dialogic cards or not, and as such it will now try to connect to a remote telephony server.

The client machine passes its own system parameters to the Voice Server machine when it connects. Voice Server uses these parameters to manage inbound calls. As more concurrent inbound calls are received, Voice Server would delegate the workload to the clients. We were able to run French and German on separate client machines even though only one Voice Server telephony machine was used.

Our test environment did not allow us to use DNIS or ANI. However, in theory Voice Server could be configured so that specific phone numbers could be directed to a particular client. What is the advantage here? Imagine a customer environment where they have several phone numbers for service. In an insurance environment, for example, they may have separate phone numbers for insurance claims, renewals, and new business.

Each would have its own dedicated phone number. A traditional IVR solution may be built to offer a front menu to the client where they select from the choices of claims, new business or renewals. With the use of DNIS, the Voice Server could distinguish what number was dialed by the customer and direct them to the appropriate application. This removes one menu layer and improves customer service.

Also, this could help separate client machines by way of departments and their specific applications. For example, claims could be handled by three clients and new business could be handled by four clients.

Additionally the ANI could be used to tailor the customer experience. If the number they are calling from is registered, the voice application could personalize the call by using the person's stored details. A simple example would be the greeting, "Welcome Tom".

6.7 Dialogic digital cards

So far we have described the step-by-step procedure to install the D/120JCT analog card. Voice Server also supports digital cards. Each card has its own specific hardware settings. It's *very* important that these parameters be set correctly. If not, the digital card will fail to work with Voice Server.

Since Voice Server's release, another analog card has been released by Dialogic. It is called the D/41JCT-LS. Although not officially supported by Voice Server, we successfully configured it. The card supports four analog lines. The benefit we see with this card is that it can provide a lower-cost telephony environment for testing systems. Alternatively it maybe used in smaller Voice Server deployment.

We were able to test two different T1 digital cards: the D/240JCT-T1 and the D/480JCT-2T1. These were configured using a CAS deployment on an Avaya Infinity PBX. The software installation procedure is the same as described earlier in this chapter. However, there are specific settings needed for each card. We have documented these specific settings for each card.

6.7.1 Dialogic D/41JCT-LS card

In this section, we discuss the D/41JCT-LS Intel Dialogic card.

Software settings

Install the Dialogic software and Service Pack 1, as described in 6.3.1, "Installing Dialogic System Release 5.1.1" on page 228.

Hardware configurations

The digital card settings are configured with DCM. The DCM window in Figure 6-63 on page 287 shows an installed D/41JCT Dialogic card.

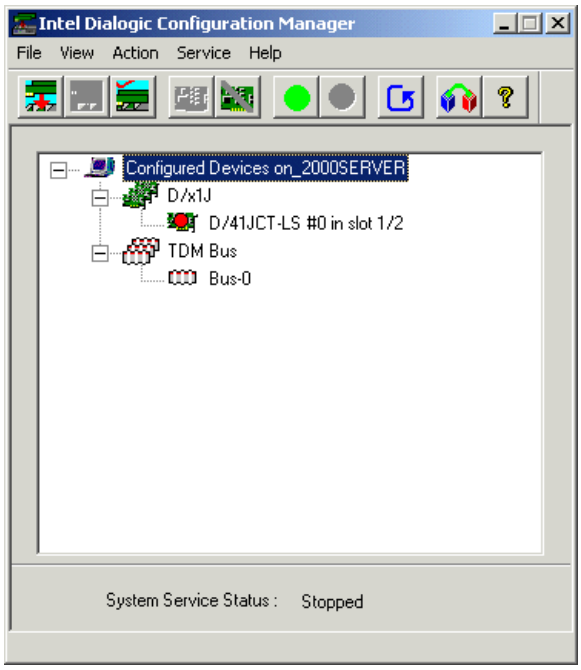


Figure 6-63 DCM window for D/41JCT-LS

Right-click the installed card (in this case it is a D/41JCT-LS1) and select **Configure Device.** The Properties window appears. Select the **Misc** tab, as illustrated in Figure 6-64 on page 288.

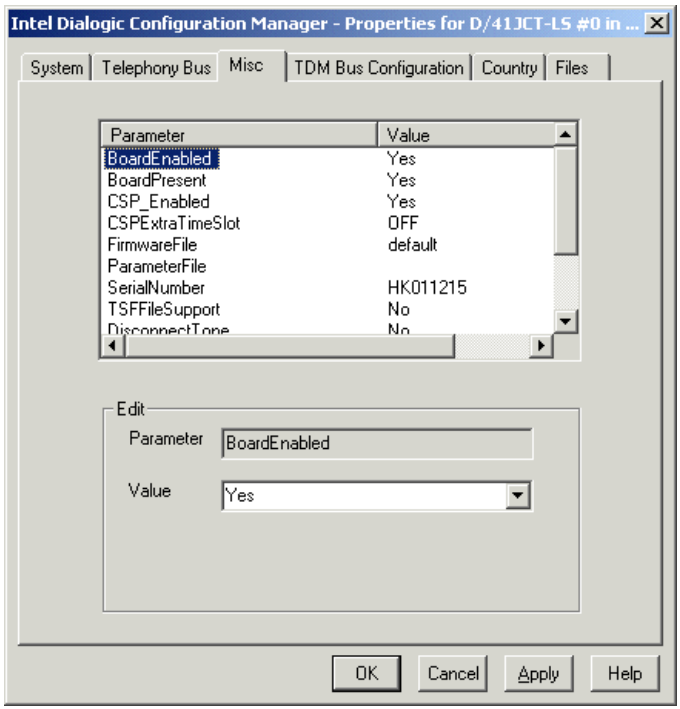


Figure 6-64 Default MISC tab for D/41JCT-LS

Table 6-5 displays the list of options specific for our D/240JCT-T1 card in a CAS deployment. You need to set these parameters correctly, or Voice Server will fail to answer inbound calls.

Table 6-5 DCM settings for D/41JCT-LS

At this tab	Ensure that this parameter	Has this value
Misc	FirmwareFile	d41JCSP.FWL
	CSPExtraTimeSlot	ON
Country	Country	*your country*

A completed Misc tab is displayed in Figure 6-65 on page 289. Note the D/41JCT-LS card options do not include the EC_Resource parameter.

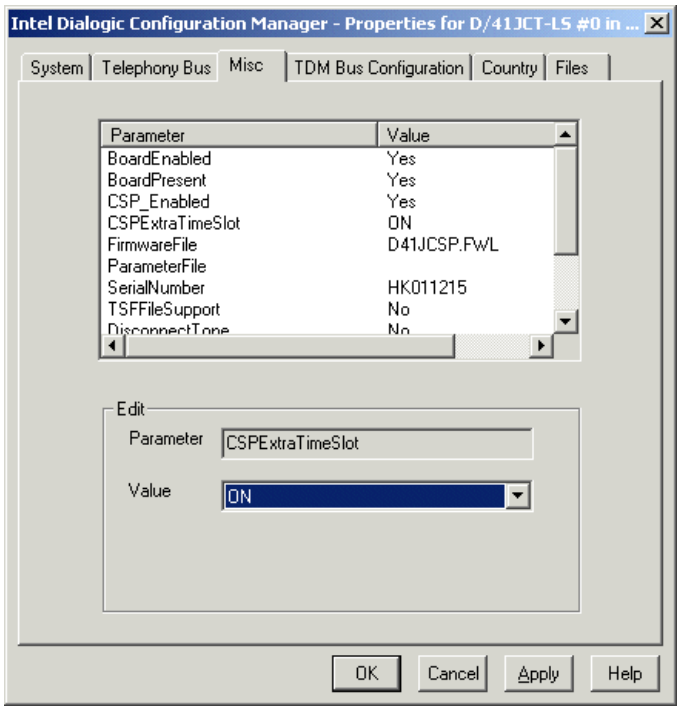


Figure 6-65 Completed Misc tab for D/41JCT-LS

Next the bus type needs to be changed. Right-click **Bus-0** on the TDM Bus Configuration tab and select **Configure Device**. The Bus configuration window will appear. One parameter needs to be changed here, as shown in Table 6-6.

Table 6-6 TDM bus parameter

On this tab	Ensure that this parameter	Has this value
TDM Bus Configuration	TDM Bus Type (User defined)	SCBus

A completed TDM Bus configuration tab is shown in Figure 6-66 on page 290. Click **Next** to return to the DCM main menu.

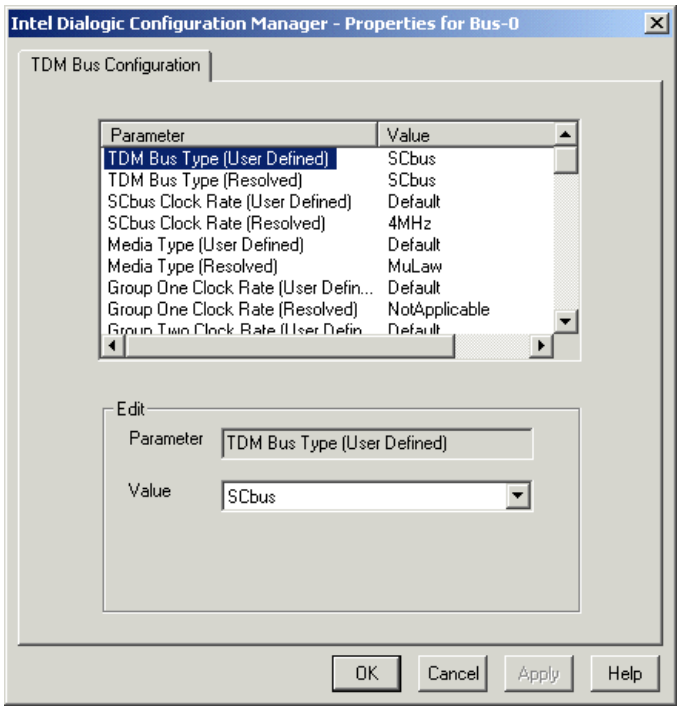


Figure 6-66 Completed TDM Bus tab for D/41JCT-LS

Start the card to test the configuration by clicking the green circle. Once completed the card will be active. The card has a green circle on it to indicate this, as shown in Figure 6-67 on page 291.

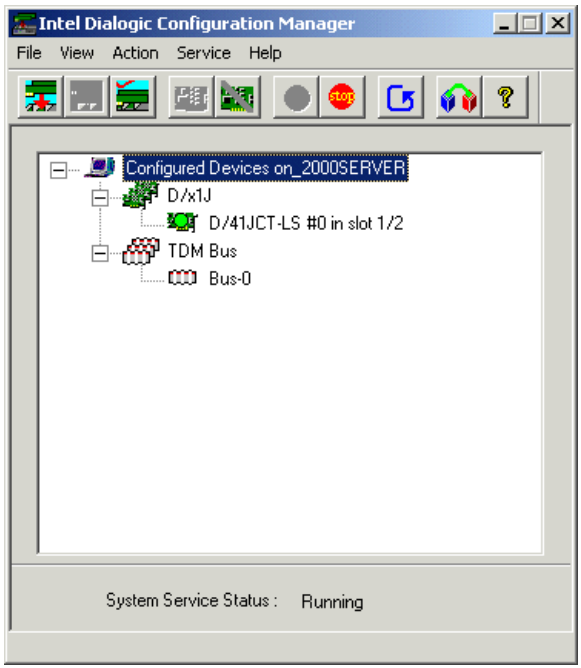


Figure 6-67 D/41JCT-LS running

D/41JCT-LS configuration parameters

Our testing with the D/41JCT was done using the same telephony environment as the D/120JCT card. That is, we had two analog lines connected to a PBX. There are two configurations files for this environment: d120jct vvtdefaults and d41jct vvtdefaults.

VVTDefaults

As with the D/120JCT, Voice Server connection to the Dialogic card is by means of parameters specified in the file called VVTDefaults. In our scenario, this file is located in c:\Program Files\Voice Server\cfg. There are four default configuration files provided. However, since the D/41JCT card was not available at the time Voice Server was released, no default VVTDefault file exists for it. We built a customized version using the Default VVTDefaults.d120jct file.

Copy and paste VVTDefaults.d120jct and rename it to VVTDefaults.

VVTDefaults is a flat text file. Each line contains an entry, which must be placed at the beginning of the line. Comments will start with a pound sign symbol (#) or two forward slashes (//).

An entry or specification in VVTDefaults is called a resource. Example 6-1 on page 274 shows a configured D/120JCT. This file can be used as the basis for the D/41JCT card with minimum parameter changes. Table 6-7 lists the parameters that need to be altered for the D/41JCT card.

Table 6-7 D/41JCT VVTDefault settings

Parameters	d120jct vvtdefaults	d41jct vvtdefaults
.,DialogicApi.boards: xxxxxx	*.*.DialogicApi.boards: d120jct	*.*.DialogicApi.boards: d41jct
..TelChannel.*.dialtoneFrequencies: xxx xxx xxx xxx	Set for your telephone environment	Set for your telephone environment
..telChannel.*.echoCancellation: x	*.*.telChannel.*.echoCancellation: 1	*.*.telChannel.*.echoCancellation: 1

6.7.2 Dialogic D/240JCT-T1 card

This section discusses the Intel Dialogic D/240JCT-T1 card.

Software settings

Install the Dialogic software and Service Pack 1, as described in 6.3.1, “Installing Dialogic System Release 5.1.1” on page 228. The process is covered in 6.3.3, “Installing GlobalCall Protocols 1.00” on page 240, and is the same process, except for the installed analog protocols setting, as shown in Figure 6-68 on page 293.

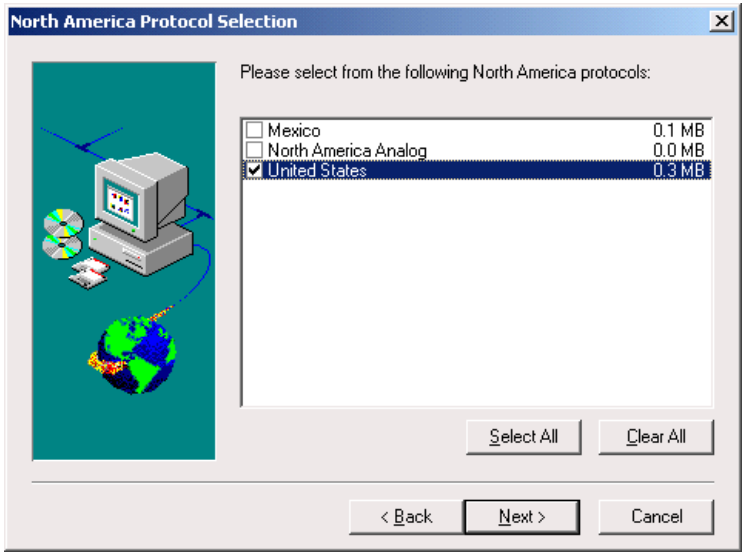


Figure 6-68 Global protocols for digital card

Hardware configurations

The digital card settings are configured with DCM. The DCM window in Figure 6-69 on page 294 shows an installed D/240JCT Dialogic card.

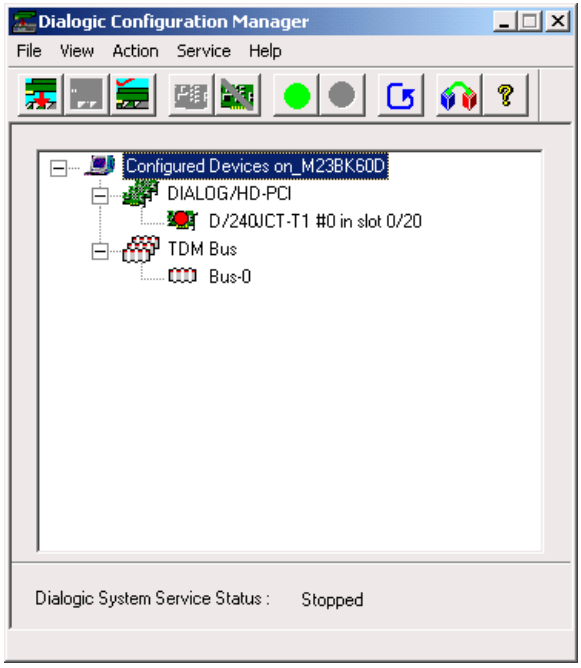


Figure 6-69 DCM window for D/240JCT

Right-click the installed card (in this case it is a D/240JCT-T1) and select **Configure Device.** The Properties window appears. Select the **Misc** tab, as illustrated in Figure 6-70 on page 295.

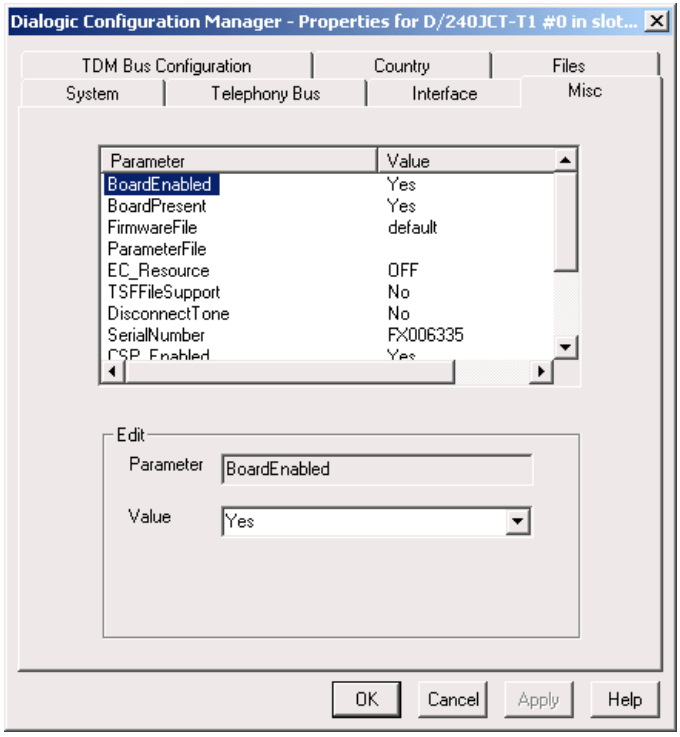


Figure 6-70 Default Misc settings

Table 6-8 displays the list of options specific for our D/240JCT-T1 card in a CAS deployment. You need to set these parameters correctly, or Voice Server will fail to answer inbound calls.

Table 6-8 DCM settings for D/240JCT card

At this tab	Ensure that this parameter	Has this value
Misc	EC_Resource	OFF
	FirmwareFile	spcsp.fwl
	ParameterFile	spandti.prm
Country	Country	*your country*
Interface	ISDNProtocol	None

For the D/240JCT card a new parameter, called ParameterFile, is required. There is no default value, and there are no listed alternatives. The Dialogic software provides a parameter file called SPANDTI.PRM. This file name must be entered into the ParameterFile value. The actual file is located in the Dialogic

CFG subdirectory. In our case it is located in the default directory, as shown in Figure 6-71.

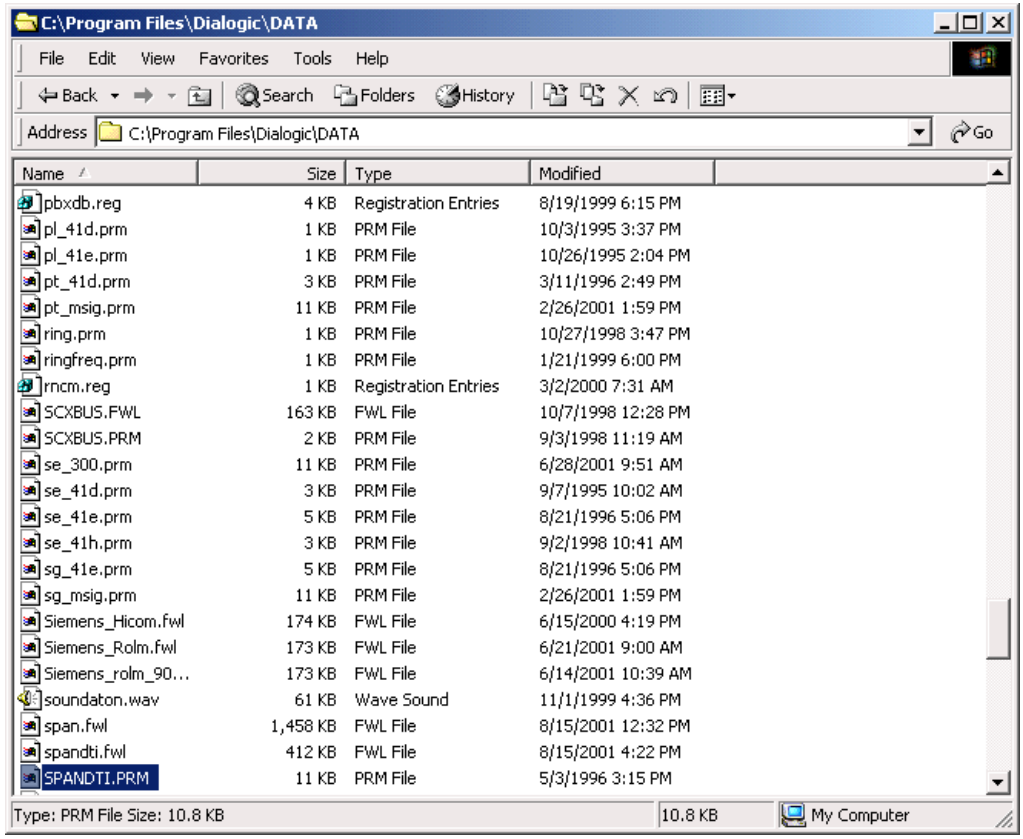


Figure 6-71 Parameter file location

When completed, the Misc tab should look like Figure 6-72 on page 297. When finished, click **Apply**, then **OK**.

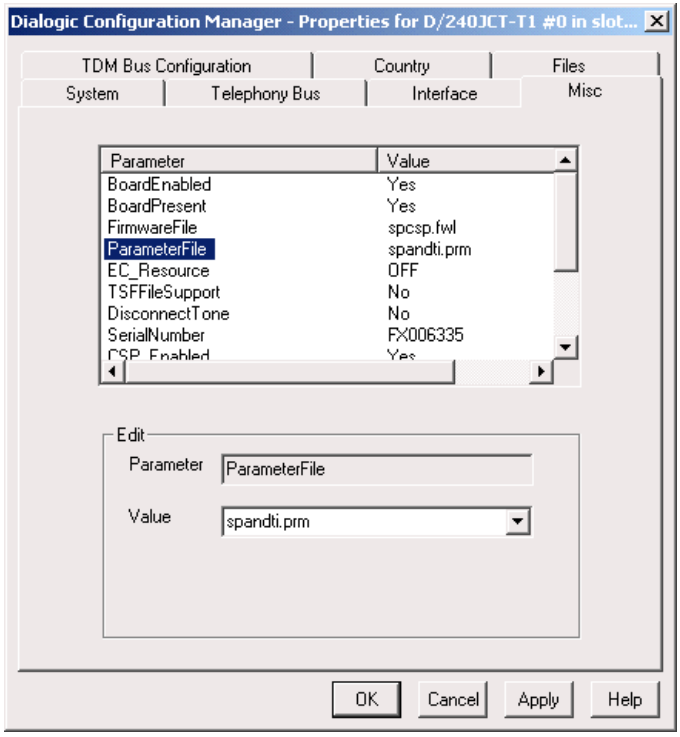


Figure 6-72 Completed Misc tab

Next the bus type needs to be changed. Right-click **Bus-0** on the TDM Bus Configuration tab and select **Configure Device**. The Bus configuration window will appear. One parameter needs to be changed here, as shown in Table 6-9.

Table 6-9 TDM Bus parameter

On this tab	Ensure that this parameter	Has this value
TDM Bus Configuration	TDM Bus Type (User defined)	SCBus

Click **OK** when done, as shown in Figure 6-73 on page 298.

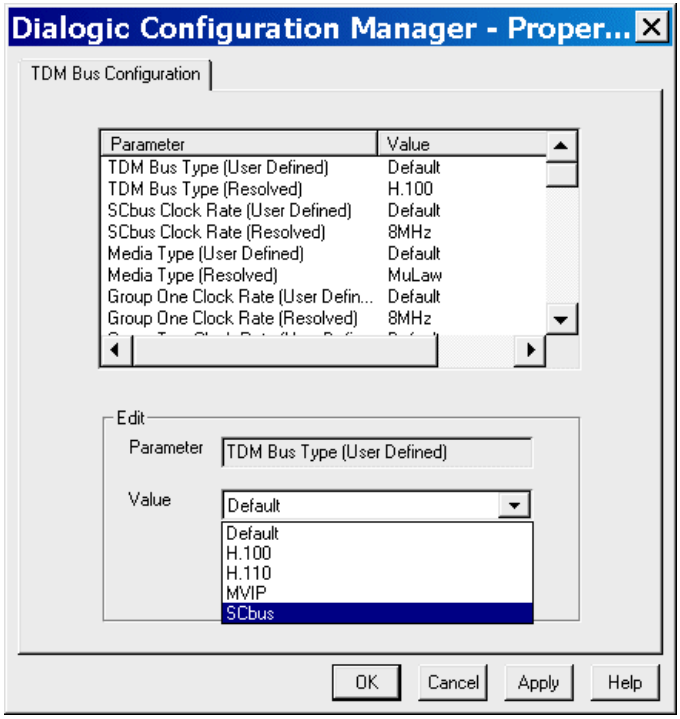


Figure 6-73 TDM Bus configuration window

Start the card to test the configuration by clicking the green circle, as in Figure 6-74 on page 299. Once completed, the card will be active. The card has a green circle on it to indicate this.

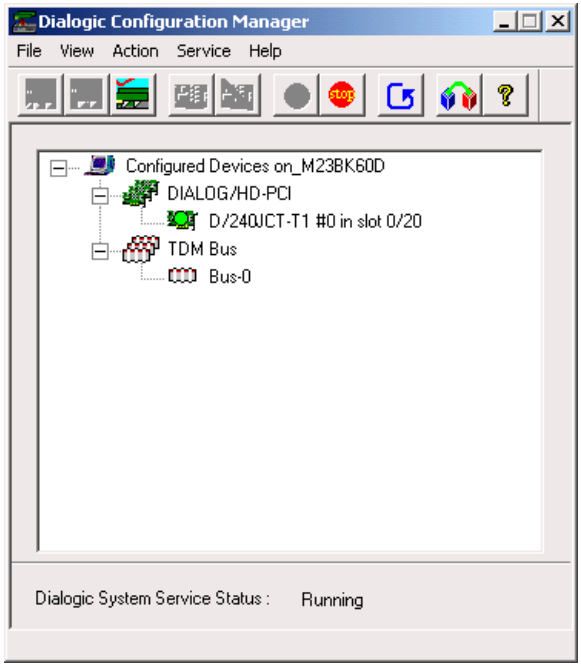


Figure 6-74 Started DCM window

D/240JCT-T1 configuration parameters

Our testing with the D/240JCT was done in a CAS environment. We had to configure three files for this environment. They were:

- VVTDefaults
- us_mf_loop_io.cdp
- icapi.cfg

VVTDefaults

The VVTDefaults configuration file needed to be created. The default file is located in c:\Program Files\Voice Server\cfg. Locate the file whose name matches the model of your installed Dialogic board. Our tested system used the D/240JCT card, so we made a copy of the file VVTDefaults.d240jct and renamed the copy VVTDefaults.

The VVTDefaults required certain parameters to be altered. These are listed in Table 6-10 on page 300.

Table 6-10 VVTDefaults D/240JCT parameters

Parameter	From:	To:
..DialogicApi.boards:	d240jct-pricsp	d240jct
..DialogicApi.useCSP:	0	1
..TelDevice.protocolFamily:	pri	cas
..TelChannel.protocol:	isdn	loopstart
..DialogicApi.resetDChannel:	1	comment out this line

Once completed, you will have a working VVTDefaults file. Example 6-3 shows the working file from our test environment.

Example 6-3 VVTdefaults for D/240JCT

```
##########################################################################
## Licensed Materials - Property of IBM
##      5724-B59
## (C) Copyright IBM Corp. 2000, 2001 All Rights Reserved
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
##########################################################################

# the list of boards, one of
# d120sc-bargein -- analog, usable channels: 0,2,4,6,8,10
# d120jct -- analog, usable channels: 0..11
# d240sc-bargein -- T1 CAS (usable channels 0..11) or PRI (usable channels
0..10)
# d240jct -- T1 CAS, usable channels 0..23
# d240jct-pricsp -- T1 PRI, usable channels 0..10
# d480jct -- dual T1 CAS, usable channels 0..47
# d480jct-pricsp -- dual T1 PRI, only on span usable (channels 0..22)
# d600jct -- dual E1 CAS, usable channels 0..59
# d600jct-pricsp -- dual E1 PRI, only one span usable (channels 0..30)
*.*.DialogicApi.boards: d240jct

# 0 for boards *sc-bargein, d240jct-pricsp
```

```
*.*.DialogicApi.useCSP: 1

# jct or sc
*.*.TelDevice.boardType: d240jct

# t1, e1 or analog
*.*.TelDevice.model: t1

*.*.TelDevice.oem: Dialogic
# pri, cas or tr (tip-ring for analog cards)
*.*.TelDevice.protocolFamily: cas

# isdn for PRI, winkstart, loopstart or opx for CAS,
# loopstart for analog cards
*.*.TelChannel.protocol: loopstart

# Certain ISDN configurations require this value to be set to 0.
#*.*.DialogicApi.resetDChannel: 1

# Setting this value to zero will disable the EC resource.
# The D/240JCT card (and all T1/E1 SC cards) need this value to be zero.
*.*.DialogicApi.useECResource: 0

# the frequencies of cleardown and dial tones are defined as
#  f1 delta1 f2 delta2
# or
#  f1 delta1 0 0
# for single frequency tones
# cleardown has to match the tone received on disconnect
*.*.TelChannel.*.cleardownFrequencies: 350 40 440 40
*.*.TelChannel.*.dialtoneFrequencies: 350 40 440 40

*.*.TelChannel.*.recordbuffersize: 1536
*.*.*.bargeIn: 1
```

```
*.*.*.useConsole: no
*.*.*.postMortemDebug: yes

# This property turns on echo cancellation
# This property does not apply to the 240 card
#*.*.telChannel.*.echoCancellation: 0

# This property sets the default log file directory
*.*.logdir: logs

# Uncomment the following statements to generate debug log files
#*.*.TelDevice.debugLevel: trace
#*.*.TelAudio.debugLevel: trace
#*.*.DialogicApi.debugLevel: trace
```

US_MF_LOOP_IO.CDP file

The us_mf_loop_io.cdp file needs to be created. The file shown in Example 6-4 should be copied and saved into a file called us_mf_loop_io.cdp. This file needs to be saved into the \dialogic\cfg subdirectory.

Example 6-4 Working us_mf_loop_io.cdp file

```
CENTRAL OFFICE TONE INFORMATION

T1 group A, B, and group D inbound protocol

@0 : "US_MF_I.LOD"

****************************************

*****************************
*    TNG # 101  DIAL         *
*****************************
@101 DIAL     : 350        -12440-12
%01 cadence   : 700
```

```
*****************************
*    TNG # 103  BUSY         *
*****************************
@103 BUSY     : 480        -12620-12
%01 cadence   : 50         50

*****************************
*    TNG # 104  SBUSY         *
*****************************
@104 BUSY     : 480        -12620-12
%01 cadence   : 25         25

*****************************
*    TNG # 105  RINGBACK     *
*****************************
@105 RINGBACK : 440        -12480-12
%01 cadence   : 100        300

$4 wink generation length (10ms) : 0

$6 inter digit timeout (s) : 0

$7 maximum number of DDI digits (excluded KP and ST) (inbound) : 0

$8 maximum number of ANI digits (excluded KP and ST) (inbound) : 0

$9 Number of ring before speech (inbound) : 0

$10 wait for idle (s) : 5

$16 collect ANI after DDI (see $47 for other options) (1=YES, 0=NO) : 0

$17 signalling transition confirmation time (10ms) : 25
```

```
$23 detect B bit signaling transitions. set to 1 if the     : 1
$26 wait for silence 0 otherwise set to 1 to monitor A/B bit for  : 1
    disconnct
$45 minimum wink duration for wink detection (10ms) : 0

$46 maximum wink duration for wink detection (10ms) : 0

$47 T1 MF protocol option mask : 64
   bit 0: 0 for wink start, 1 for special protocols
   bit 1: 0 for DTMF digit, 1 for MF digit
   bit 2: 0 for group feature B, 1 for group feature D
   bit 3: selects DNIS wink for group D protocol 0=no wink, 1=wink
   bit 4: selects ANIS wink for group D protocol 0=no wink, 1=wink
   bit 5: selects dial tone after seizure: 0=no dial tone, 1=dial tone
   bit 6: disable seizure ack wink, 0=wink, 1=no wink

$50 bits set and cleared for NT BLOCKING   :   "-1--"
                                               ABCD ('-'=don't care)

$51 bits set and cleared for NT IDLE       :   "01--"
                                               ABCD ('-'=don't care)

52 bits set and cleared for NT SEIZURE    :   "00--"
                                               ABCD ('-'=don't care)

$53 bits set and cleared for NT DISCONNECT :   "11--"
                                               ABCD ('-'=don't care)

@50 bits set and cleared for DTI BLOCKING   :  "1100"
                                               ABCD ('-'=don't modify)

@51 bits set and cleared for DTI IDLE       :  "01--"
                                               ABCD ('-'=don't modify)
```

```
@52 bits set and cleared for DTI SEIZURE    :  "11--"
                                               ABCD ('-'=don't modify)

@53 bits set and cleared for DTI DISCONNECT :  "01--"
                                               ABCD ('-'=don't modify)

@54 bits set and cleared for DTI ACKNOWLEDGE:  "00--"
                                               ABCD ('-'=don't modify)

@55 bits set and cleared for DTI ANSWER     :  "11--"
                                               ABCD ('-'=don't modify)

@56 bits set and cleared for DTI START      :  "1100"
                                               ABCD ('-'=don't modify)
```

ICAPI.CFG file

The Dialogic file called icapi.cfg, located in the dialogic/cfg subdirectory, required a parameter to be changed:.

```
$14 disable DTI wait call function (1=Yes, 0=No) : 1
```

When created and configured correctly, these files will enable Voice Server to connect to the telephony system and answer inbound calls.

Important: Care must be taken to ensure all the values are set correctly. If not, Voice Server will never connect to the telephony hardware platform. Ensure that the inbound calls can be answered by Dialogic software before attempting to configure Voice Server. See the Dialogic documentation if inbound calls are not answered at this level.

6.7.3 Dialogic D/480JCT-2T1 card

In this section, we discuss the Intel Dialogic D/480JCT-2T1 card.

Software settings

Install the Dialogic software and Service Pack 1, as described in 6.3.1, “Installing Dialogic System Release 5.1.1” on page 228. The process is covered in 6.3.3, “Installing GlobalCall Protocols 1.00” on page 240, and is the same process, except for the installed analog protocols setting, as shown in Figure 6-75 on page 306.

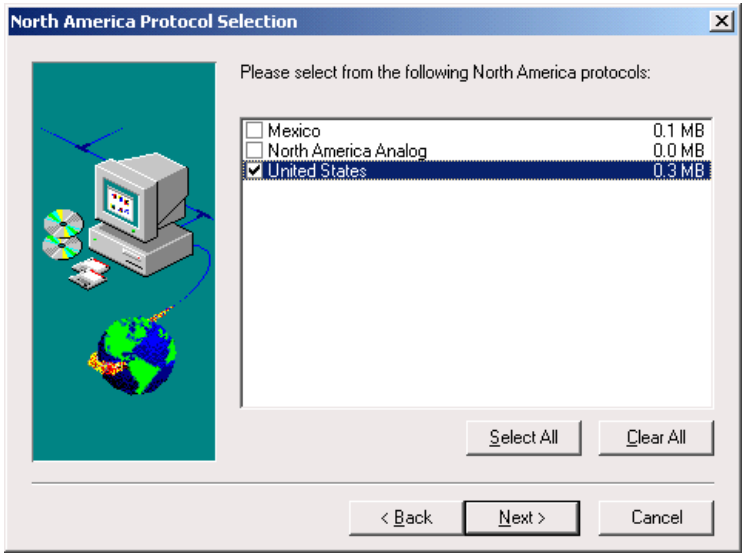


Figure 6-75 Global protocols for digital card

Hardware configurations

The digital card settings are configured with DCM. The DCM window in Figure 6-76 on page 307 shows an installed D/480JCT Dialogic card.

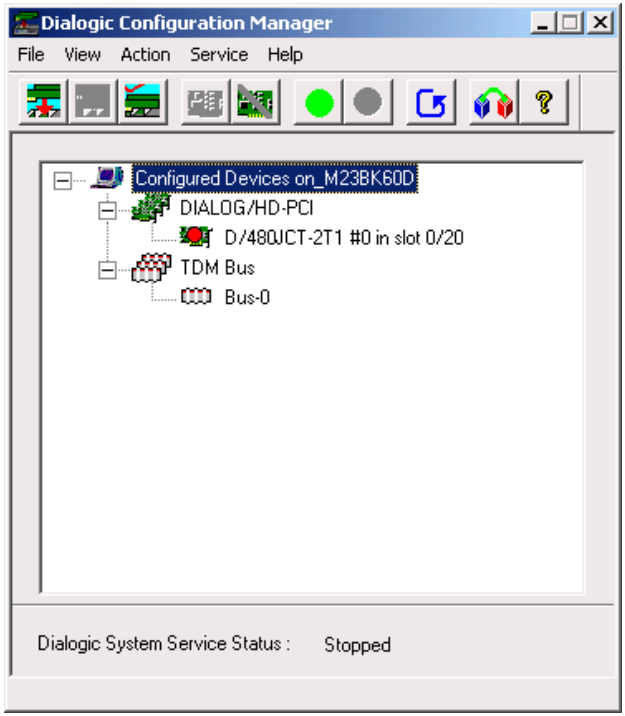


Figure 6-76 DCM window for D/480JCT

Right-click the installed card (in this case it is a D/480JCT-2T1) and select **Configure Device.** The Properties window appears. Select the **Misc** tab, as illustrated in Figure 6-77 on page 308.

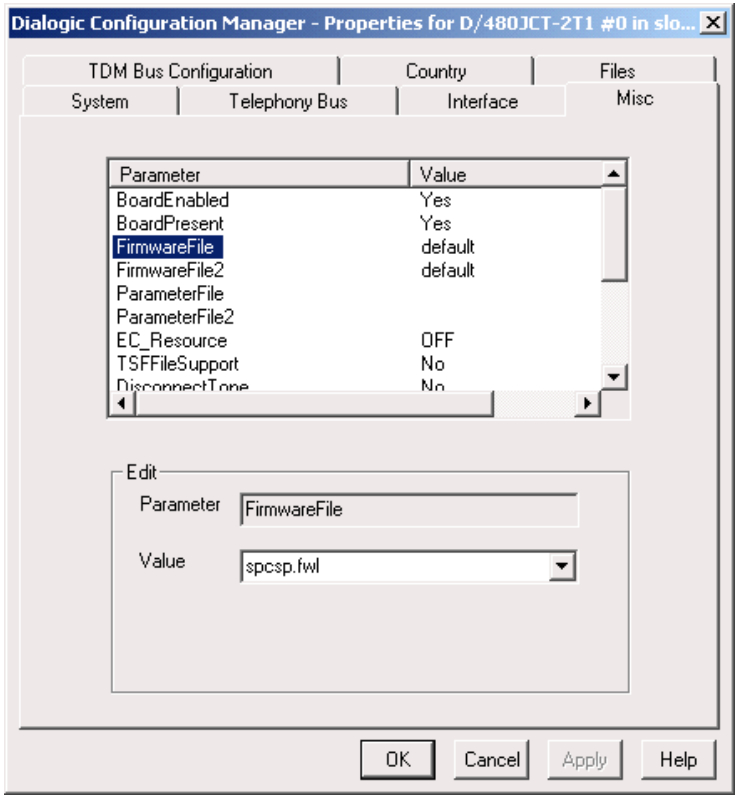


Figure 6-77 Default Misc settings

Table 6-8 on page 295 displays the list of options specific for our D/480JCT-2T1 card in a CAS deployment. You need to set these parameters correctly, or Voice Server will fail to answer inbound calls.

Table 6-11 DCM settings for D/480 card

At this tab	Ensure that this parameter	Has this value
Misc	EC_Resource	OFF
	FirmwareFile	spcsp.fwl
	FirmwareFile2	spcsp.fwl
	ParameterFile	spandti.prm
	ParameterFile2	spandti.prm
Country	Country	*your country*

At this tab	Ensure that this parameter	Has this value
Interface	ISDNProtocol	None
	ISDNProtocol2	None

Note: There are three parameters that appear twice. In each instance, the parameter must contain a value.

For the D/480JCT card a new parameter, called ParameterFile, is required. There is no default value, and there are no listed alternatives. The Dialogic software provides a parameter file called SPANDTI.PRM. This file name must be entered into the ParameterFile and ParameterFile2 value. The actual file is located in the Dialogic CFG subdirectory. In our case it is located in the default directory, as shown in Figure 6-78.

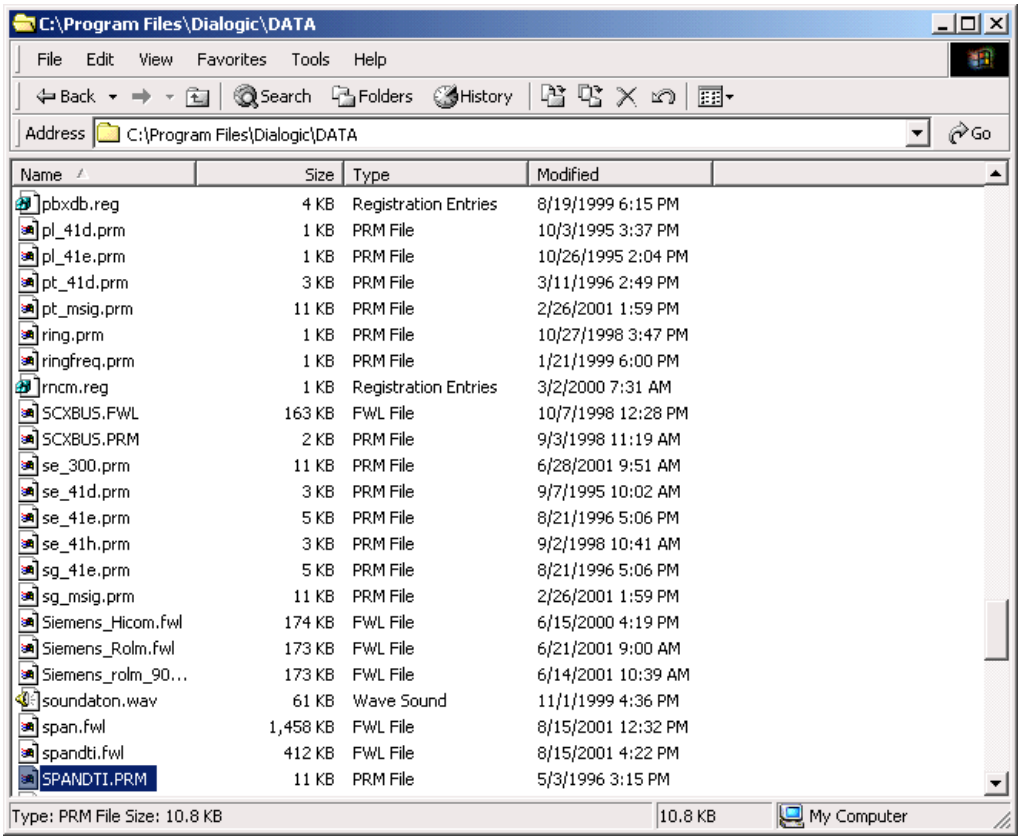


Figure 6-78 Parameter file location

When completed, the Misc tab should look like Figure 6-79. When finished, click **Apply**, then **OK**.

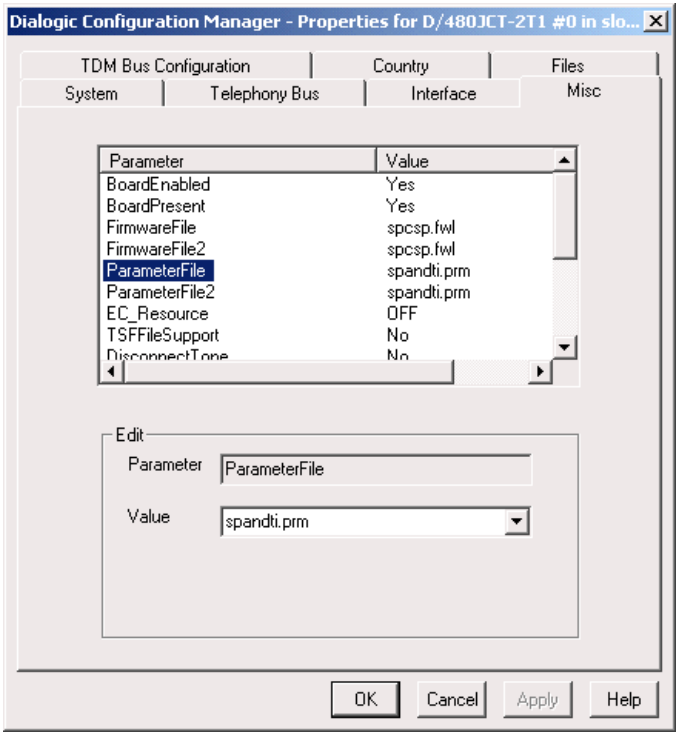


Figure 6-79 Completed Misc tab

Next the bus type needs to be changed. Right-click **Bus-0** on the TDM Bus Configuration tab and select **Configure Device**. The Bus configuration window will appear. One parameter needs to be changed here, as shown in Table 6-12.

Table 6-12 TDM Bus parameter

At this tab	Ensure that this parameter	Has this value
TDM Bus Configuration	TDM Bus Type (User defined)	SCBus

Click **OK** when done, as shown in Figure 6-80 on page 311.

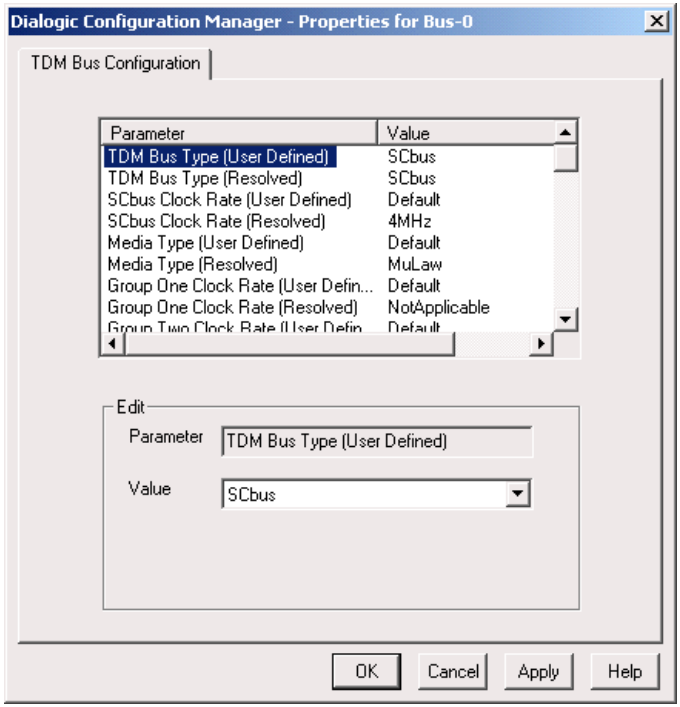


Figure 6-80 TDM Bus configuration window

Start the card to test the configuration by clicking the green circle, as in Figure 6-81 on page 312. Once completed the card will be active. The card has a green circle on it to indicate this.

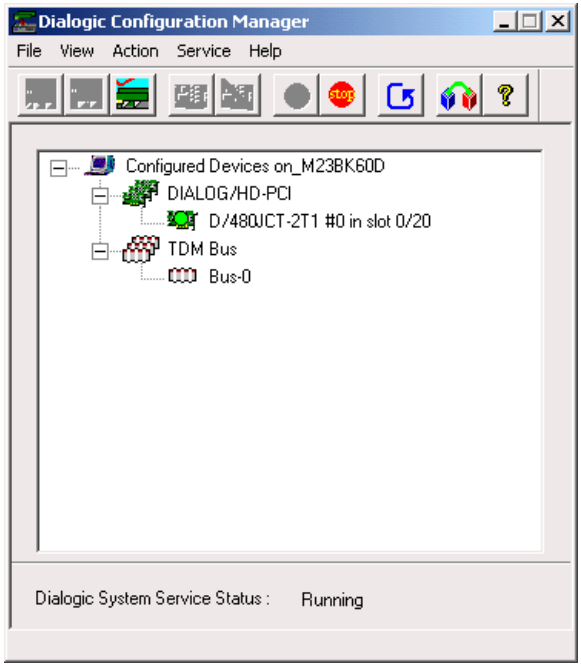


Figure 6-81 Started DCM window

D/480JCT-2T1 configuration parameters

Our testing with the D/480JCT was done in a CAS environment. We had to configure three files for this environment. They were:

- VVTDefaults
- us_mf_loo_io.cdp
- icapi.cfg

VVTDefaults

The VVTDefaults configuration file needed to be created. The default file is located in c:\Program Files\Voice Server\cfg. Locate the file whose name matches the model of your installed Dialogic board. Our tested system used the D/480JCT card, so we made a copy of the file VVTDefaults.d480jct and renamed the copy VVTDefaults.

The VVTDefaults required certain parameters to be altered. These are listed in Table 6-13 on page 313.

Table 6-13 VVTDefaults D/480JCT parameters

Parameter	From:	To:
..DialogicApi.boards:	d480jct-pricsp	d480jct
..DialogicApi.useCSP:	0	1
..TelDevice.protocolFamily:	pri	cas
..TelChannel.protocol:	isdn	loopstart
..DialogicApi.resetDChannel:	1	comment out this line

Once completed, you will have a working VVTDefaults file. Example 6-5 shows the working file from our test environment.

Example 6-5 VVTdefaults for D/480JCT

```
##########################################################################
## Licensed Materials - Property of IBM
##      5724-B59
## (C) Copyright IBM Corp. 2000, 2001 All Rights Reserved
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
##########################################################################

# the list of boards, one of
# d120sc-bargein -- analog, usable channels: 0,2,4,6,8,10
# d120jct -- analog, usable channels: 0..11
# d240sc-bargein -- T1 CAS (usable channels 0..11) or PRI (usable channels
0..10)
# d240jct -- T1 CAS, usable channels 0..23
# d240jct-pricsp -- T1 PRI, usable channels 0..10
# d480jct -- dual T1 CAS, usable channels 0..47
# d480jct-pricsp -- dual T1 PRI, only on span usable (channels 0..22)
# d600jct -- dual E1 CAS, usable channels 0..59
# d600jct-pricsp -- dual E1 PRI, only one span usable (channels 0..30)
*.*.DialogicApi.boards: d480jct

# 0 for boards *sc-bargein, d240jct-pricsp
```

```
*.*.DialogicApi.useCSP: 1

# jct or sc
*.*.TelDevice.boardType: d480jct

# t1, e1 or analog
*.*.TelDevice.model: t1

*.*.TelDevice.oem: Dialogic
# pri, cas or tr (tip-ring for analog cards)
*.*.TelDevice.protocolFamily: cas

# isdn for PRI, winkstart, loopstart or opx for CAS,
# loopstart for analog cards
*.*.TelChannel.protocol: loopstart

# Certain ISDN configurations require this value to be set to 0.
#*.*.DialogicApi.resetDChannel: 1

# Setting this value to zero will disable the EC resource.
# The D/240JCT card (and all T1/E1 SC cards) need this value to be zero.
*.*.DialogicApi.useECResource: 1

# the frequencies of cleardown and dial tones are defined as
#  f1 delta1 f2 delta2
# or
#  f1 delta1 0 0
# for single frequency tones
# cleardown has to match the tone received on disconnect
*.*.TelChannel.*.cleardownFrequencies: 350 40 440 40
*.*.TelChannel.*.dialtoneFrequencies: 350 40 440 40

*.*.TelChannel.*.recordbuffersize: 1536
*.*.*.bargeIn: 1
```

```
*.*.*.useConsole: no
*.*.*.postMortemDebug: yes

# This property turns on echo cancellation
*.*.telChannel.*.echoCancellation: 1

# This property sets the default log file directory
*.*.logdir: logs



# Uncomment the following statements to generate debug log files
#*.*.TelDevice.debugLevel: trace
#*.*.TelAudio.debugLevel: trace
#*.*.DialogicApi.debugLevel: trace
```

US_MF_LOOP_IO.CDP file

The us_mf_loop_io.cdp file needs to be created. The file in Example 6-6 should be copied and saved into a file called us_mf_loop_io.cdp. This file needs to be saved into the \dialogic\cfg subdirectory.

Example 6-6 Working us_mf_loop_io.cdp file

```
CENTRAL OFFICE TONE INFORMATION
T1 group A, B, and group D inbound protocol

@0 : "US_MF_I.LOD"
****************************************

*****************************
*    TNG # 101  DIAL        *
*****************************
@101 DIAL     : 350        -12440-12
%01 cadence   : 700
*****************************
*    TNG # 103  BUSY        *
*****************************
```

```
@103 BUSY     : 480        -12620-12
%01 cadence   : 50         50

*****************************
*    TNG # 104  SBUSY         *
*****************************
@104 BUSY     : 480        -12620-12
%01 cadence   : 25         25

*****************************
*    TNG # 105  RINGBACK     *
*****************************
@105 RINGBACK : 440        -12480-12
%01 cadence   : 100        300


$4 wink generation length (10ms) : 0

$6 inter digit timeout (s) : 0

$7 maximum number of DDI digits (excluded KP and ST) (inbound) : 0

$8 maximum number of ANI digits (excluded KP and ST) (inbound) : 0

$9 Number of ring before speech (inbound) : 0

$10 wait for idle (s) : 5

$16 collect ANI after DDI (see $47 for other options) (1=YES, 0=NO) : 0

$17 signalling transition confirmation time (10ms) : 25

$23 detect B bit signaling transitions. set to 1 if the     : 1
$26 wait for silence 0 otherwise set to 1 to monitor A/B bit for  : 1
```

```
    disconnct
$45 minimum wink duration for wink detection (10ms) : 0

$46 maximum wink duration for wink detection (10ms) : 0

$47 T1 MF protocol option mask : 64
   bit 0: 0 for wink start, 1 for special protocols
   bit 1: 0 for DTMF digit, 1 for MF digit
   bit 2: 0 for group feature B, 1 for group feature D
   bit 3: selects DNIS wink for group D protocol 0=no wink, 1=wink
   bit 4: selects ANIS wink for group D protocol 0=no wink, 1=wink
   bit 5: selects dial tone after seizure: 0=no dial tone, 1=dial tone
   bit 6: disable seizure ack wink, 0=wink, 1=no wink

$50 bits set and cleared for NT BLOCKING   :   "-1--"
                                               ABCD ('-'=don't care)

$51 bits set and cleared for NT IDLE       :   "01--"
                                               ABCD ('-'=don't care)

52 bits set and cleared for NT SEIZURE    :   "00--"
                                               ABCD ('-'=don't care)

$53 bits set and cleared for NT DISCONNECT :   "11--"
                                               ABCD ('-'=don't care)

@50 bits set and cleared for DTI BLOCKING   :  "1100"
                                               ABCD ('-'=don't modify)

@51 bits set and cleared for DTI IDLE       :  "01--"
                                               ABCD ('-'=don't modify)

@52 bits set and cleared for DTI SEIZURE    :  "11--"
                                               ABCD ('-'=don't modify)
```

```
@53 bits set and cleared for DTI DISCONNECT :  "01--"
                                                ABCD ('-'=don't modify)

@54 bits set and cleared for DTI ACKNOWLEDGE:  "00--"
                                                ABCD ('-'=don't modify)

@55 bits set and cleared for DTI ANSWER     :  "11--"
                                                ABCD ('-'=don't modify)

@56 bits set and cleared for DTI START      :  "1100"
                                                ABCD ('-'=don't modify)
```

ICAPI.CFG file

The Dialogic file called icapi.cfg, located in the dialogic/cfg subdirectory, required a parameter to be changed:

```
$14 disable DTI wait call function (1=Yes, 0=No) : 1
```

When created and configured correctly, these files will enable Voice Server to connect to the telephony system and answer inbound calls.

Important: Care must be taken to ensure all the values are set correctly. If not, Voice Server will never connect to the telephony hardware platform. Ensure that the inbound calls can be answered by Dialogic software before attempting to configure Voice Server. See the Dialogic documentation if inbound calls are not answered at this level.

6.8 Voice Server considerations

During our testing of Voice Server, we were able to configure different environments and implementations. Some of the more important findings are included in this section to help you with your deployment.

Voice browser auto start

The default setting for AUTO_START_ON_INIT is false. This means every time Voice Server is started, you manually have to start the voice browsers. If you change this parameter to true, the VoiceXML browsers will start automatically on reboot. No manual intervention is required.

Call transfer

WebSphere Voice Server 2.0 for the Dialogic telephony environment does not support call transfers. Within the VoiceXML standard are call transfer functions. A VoiceXML application that has these parameters will appear to be coded correctly. However, when a voice browser attempts to execute them, a systems failure error will occur. The error will be recorded in the VXML.log, as shown in Figure 6-82.

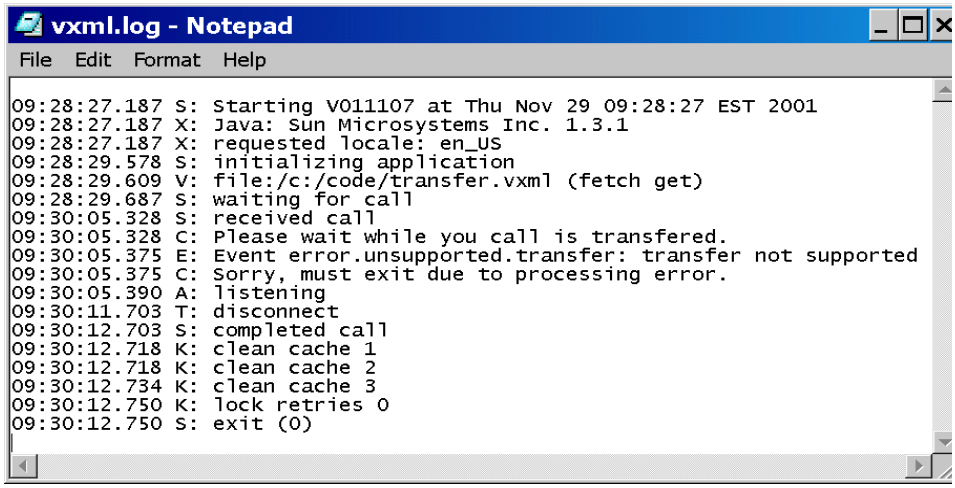


Figure 6-82 VXML log

Dialogic card D41ESC

Earlier in this chapter we listed the support Dialogic PCI telephony cards. Dialogic also markets a variety of others. The D41ESC is an ISA card. It does not support barge-in and continuous speech processing (that is, full-duplex), which barge-in requires. Voice Server does not support this card.

Chaining multiple Dialogic cards

At the time of writing, the concept of using multiple cards in a machine is supported in theory. When using multiple cards the SCBus cable must be used to connect the cards together. Failure to do this will stop DCM from starting the cards. However, after some testing there appears to be a limitation with Voice Server. This issue has been raised with the developers of Voice Server. Currently this feature is not functioning.

Distributed system configuration

When multiple voice browsers clients are used, care needs to be taken in regards to security. When each client is started, it notifies the Voice Server and also passes its own system management properties to the server. The server will accept these parameters in its call management process. No authentication process occurs. The potential is that another computer on the same network, which knows what the Voice Server IP address is, could connect to the server as a voice browser client. This client could then run its own unauthorized VoiceXML application.

Debugging functions

Voice Server provides several debugging options that generate detailed logs. These are useful when trying to diagnose Voice Server problems. The most commonly used debugging options are set in the VVTDefaults file:

- #*.*.TelDevice.debugLevel: trace
- #*.*.TelAudio.debugLevel: trace
- #*.*.DialogApi.debugLevel: trace

System resource usage is high when these are active. Care should be taken to have them disabled when the system is running a production environment.

7

WebSphere Voice Server for Windows 2000 with Software Developers Kit V3.1

This chapter describes the installation and use of the WebSphere Voice Server for Windows 2000 with SDK 3.1. The SDK uses VoiceXML technology to facilitate developers to create voice-enabled applications and test them on a desktop workstation.

7.1 The Software Developers Kit (SDK)

The WebSphere Voice Server for Windows 2000 with SDK 3.1 includes a VoiceXML speech browser, speech recognizer, speech synthesizer, sample applications, and other tools for developing and testing VoiceXML applications.

Features of the WebSphere Voice Server for Windows 2000 with SDK 3.1 include:

- Support for VoiceXML 1.0 plus IBM extensions, as documented in the *IBM WebSphere Voice Server Software Developers Kit (SDK) Programmer's Guide*, Version 3.1.
- Speech or simulated DTMF (telephone key press) input.
- Synthesized speech (text-to-speech) or prerecorded audio file output.
- Six audio types for prerecorded audio files:
  - 8kHz 8-bit mu-law .au file
  - 8kHz 8-bit mu-law .wav file
  - 8kHz 8-bit a-law .au file
  - 8kHz 8-bit a-law .wav file
  - 8kHz 16bit linear .au file
  - 8kHz 16bit linear .wav file
- Support for the following languages:
  - Brazilian Portuguese
  - Canadian French
  - French
  - German
  - Italian
  - Japanese
  - Simplified Chinese
  - Spanish
  - UK English
  - US English
- "Barge-in" -- application users can interrupt audio prompts by speaking.
- Grammar-based speech recognition, including support for dynamically generated grammars.
- Support for filling in forms.
- Support for "mixed initiative dialogs" -- application users can include multiple requests in a single utterance versus directed dialog where a user's response is limited to only one request at a time.

7.1.1 Updates from SDK 2.1 to SDK 3.1

- Added support for additional languages:
  - Brazilian Portuguese
  - Canadian French
- Additional VoiceXML attributes and elements:
  - For Japanese and Simplified Chinese, the notation "Spelling~Soundslike" enables grammar developers to specify a word's “sounds-like” spelling for use by the text-to-speech engine.
  - The VoiceXML element <object> is supported with some limitations.
  - The Augmented Backus Naur Form (ABNF) and Extensible Markup Language (XML) form of the W3C Speech Recognition Grammar can be used for grammar development. Some limitations apply.
  - The name="confidencelevel" attribute of the <property> element and the name$.confidence shadow variable of the <field> element are supported as described in the VoiceXML 1.0 Specification.
  - N-best results representing the outcome of a speech-recognition event are implemented. The following application variables attribute are supported:
    - application.lastresults$[i].confidence
    - application.lastresults$[i].utterance
    - application.lastresults$[i].inputmode
    - <property> element's name="maxnbest"
- WebSphere Voice Server with WebSphere Voice Response for Windows is a supported connection environment. (WebSphere Voice Response was formerly known as IBM DirectTalk.)
- VoiceXML applications that involve speech in Brazilian Portuguese and Canadian French are supported.
- The WebSphere Voice Server for Windows 2000 with SDK 3.1 supports the use of the separately available WebSphere Voice Server Version 3.1 Concatenative Text-to-Speech Development Edition.

7.1.2 The WebSphere Voice Server for Windows 2000 with SDK 3.1 components

The following are the SDK components of WebSphere Voice Server:

- Speech recognition engine

  The IBM speech recognition engine converts spoken input to text by first parsing the input audio stream and then converting that information to text output. The high-level process looks like this:

  a. The application developer creates a series of speech recognition grammars defining the words and phrases that can be spoken by the user, and specifies where each grammar should be active within the application.

  b. When the application runs, the speech recognition engine processes the incoming audio signal and compares the sound patterns to the patterns of basic spoken sounds, trying to determine the most probable combination that represents the audio input.

  c. Finally, the speech recognition engine compares the sounds to the list of words and phrases in the active grammar(s). Only words and phrases in the active grammars are considered as possible speech recognition candidates. Any word for which the speech recognizer does not have a pronunciation is given one and is flagged as an unknown word.

- Text-to-speech (speech synthesis) engine

  To generate synthesized speech, the computer must first parse the input text to determine its structure and then convert that text to spoken output. In the WebSphere Voice Server SDK, this is done by the text-to-speech (TTS) engine.

- VoiceXML browser

  One of the primary functions of the VoiceXML browser is to fetch documents to process. The request to fetch a document can be generated either by the interpretation of a VoiceXML document or in response to an external event.

  The VoiceXML browser manages the dialog between the application and the user by playing audio prompts, accepting user inputs, and acting on those inputs. The action might involve jumping to a new dialog, fetching a new document, or submitting user input to the Web server for processing. The VoiceXML browser is a Java application. The Java console provides trace information on the prompts played, resource files fetched, and user input recognized. Other than this and the DTMF Simulator GUI, there is no visual interface.

7.2 WebSphere Voice Server for Windows 2000 with SDK 3.1 prerequisites

The following are the SDK's hardware and software requirements:

- Hardware requirements
  - Intel Pentium 366 MHz processor
  - CD-ROM drive, if installing the Voice Server SDK from a CD
  - 128 MB RAM
  - 290 MB disk space (minimum) for each language to be installed, which includes:
    - 30 MB for installing the Sun Java Runtime Environment (Sun JRE) 1.3.1
    - 80 MB in the Windows system directory
    - 130 MB disk space in the destination installation directory for installing each language selected
    - 28 MB in the installation destination directory for caching, logging, and tracing

Note: If you are downloading the SDK from a Web site, the installation packages (a total of nine) require an additional 58 MB to 171 MB of free space during download, plus 62 MB to 172 MB of free space for extracting the files that are created when executing the download packages. The range varies depending on the language packages downloaded:

- 50 MB for the main installation package (required)
- Approximately 50 MB for each language file

After you have installed the SDK product, you can remove the downloaded packages and the extracted installation program files. You must download at least one language package along with the main installation package.

  - If you plan to install IBM WebSphere Voice Server Concatenative Text-to-Speech, the following additional resources are required:
    - 150 MB RAM for each CTTS language (excluding Simplified Chinese, which requires an additional 300 MB RAM)
    - 640 MB for each CTTS language (excluding Simplified Chinese, which requires an additional 600 MB of disk space), of which 300 MB are for temporary storage

- Addtional requirements exist for CTTS voices. The shared memory listed in Table 7-1 is required.

Table 7-1 Shared memory requirements

Language Supported	Shared Memory Requirements
Brazilian Portuguese	► Voice 1 (Male): 115 MB ► Voice 2 (Female): 140 MB
Canadian French	► Voice 1 (Male): 140 MB ► Voice 2 (Female): 145 MB
French	► Voice 1 (Male): 95 MB ► Voice 2 (Female) 135 MB
German	► Voice 1 (Male): 110 MB ► Voice 2 (Female): 125 MB
Italian	► Voice 1 (Male): 105 MB ► Voice 2 (Female): 90 MB
Japanese	► Voice 1 (Male): 70 MB ► Voice 2 (Female): 75 MB
Simplified Chinese	► Voice 1 (Male): 225 MB ► Voice 2 (Female): 240 MB
Spanish	► Voice 1 (Male): 95 MB ► Voice 2 (Female): 115 MB
U.K. English	► Voice 1 (Male): 70 MB ► Voice 2 (Female): 145 MB
U.S. English	► Voice 1 (Male): 130 MB ► Voice 2 (Female): 160 MB ► Voice 4 (Male 2): 145 MB

  - A display adapter setting of greater than 256 colors.
  - A Microsoft Windows 2000 compatible 16-bit full-duplex sound card (with a microphone input jack) with good recording quality.
    - For a list of the supported sound cards and microphones, visit the Web site: http://www.ibm.com/software/speech/support.
  - A good quality microphone.

- Software requirements
  - Microsoft Windows 2000 Professional or Server with Service Pack 2 or higher

- Sun Java Runtime Environment (Sun JRE) 1.3.1 (included in this package, but must be installed prior to the IBM WebSphere Voice Server SDK software)
- Networking (for example, TCP/IP) must be enabled
- Adobe Acrobat Reader, Version 5.0 (included in this package) or later
- HTTP 1.1 Client, if desired

Restriction:

1. The SDK cannot be installed on a machine running one of the WebSphere Voice Server deployment platforms.
2. You must install the SDK on a local hard drive. You cannot install the SDK on a local floppy, zip, CD-ROM, or remote drive.
3. If you have an earlier release of the SDK, you must uninstall it before installing Release 3.1.

7.3 WebSphere Voice Server SDK 3.1 installation

The SDK installation consists of the following phases:

1. Unpacking the SDK base setup
2. Unpacking the SDK language setup
3. Running the SDK setup
4. Audio setup for the SDK

7.3.1 Uninstalling the SDK

If you want to install another IBM WebSphere Voice Server product or an IBM ViaVoice product on your system, you must first uninstall the IBM WebSphere Voice Server SDK. The uninstallation will remove all installed language versions.

To uninstall the SDK, start the Add/Remove Programs utility by selecting **Start > Settings > Control Panel > Add/Remove Programs**.

After uninstalling the SDK, uninstall the JRE.

Note: Note: If the uninstallation stops abnormally, you should run the utility described in 7.3.2, “TTSCLEAN utility” on page 328.

7.3.2 TTSCLEAN utility

If the SDK installation stops or the uninstallation is unsuccessful or incomplete, you should run the TTSCLEAN utility. This utility removes SDK registry entries that are left on your system.

TTSCLEAN is not installed on your hard disk; it is located only on the installation media in the following location:

...\install\Support\ttsclean.exe

Any installed files that are left on your system after running TTSCLEAN (for example, files in C:\Program Files\VoiceServerSDK) should be manually removed.

7.3.3 Implementing the Secure Sockets Layer (SSL) protocol

Secure Socket Layer (SSL) protocol provides authentication and data security. It encapsulates a TCP/IP socket and is used by TCP/IP applications that require secure communications. SSL is a low-level authentication and encryption service used by higher-level applications. SSL allows encrypted and secure exchange transmission between a VoiceXML browser instance and an HTTPS Web application server.

To implement secure communications within your deployment environment, your Web-based voice application must point to a secure Web application server (HTTPS protocol).

- In the WebSphere Voice Response connection environment, specify HTTPS as the transfer protocol in your URIs.

Example 7-1 Use HTTPS as the transfer protocol in default.cff for a managed applicatior

```
AppName=weather
Enabled=yes
Parameter=URI, https://my.secureserver/samples/weather.VXML
AppClass=com.ibm.speech.VXML.DTVoicelet
;
```

Example 7-2 Using HTTPS as the transfer protocol in an unmanaged application

```
VXML -debug https://my.secureserver/samples/weather.VXML
```

Example 7-3 Using HTTPS as the transfer protocol inside a document

```
<goto next="https://my.secureserver/next.VXML" />
```

- In the Cisco and Intel Dialogic connection environments, edit the INBOUND_URL nparameter in the sysmgmt.properties file so that the URL protocol defined for the VoiceXML application is HTTPS.

Example of configuring INBOUND_URL nparameters for two VoiceXML browsers:

Example 7-4 INBOUND_URL parameter for browser 1

```
INBOUND_URL0=https://my.secureserver/samples/weather.VXML
```

Example 7-5 INBOUND_URL nparamter for browser 2

```
INBOUND_URL1=https://my.secureserver/queries/balances.VXML
```

- For the SDK, use the protocol HTTPS for the URL of the target application. See 7.3.11, “Testing the application in audio mode” on page 359 for more information.

The encryption algorithm used for the secure transmission will automatically be negotiated between the VoiceXML browser and the Web application server hosting the Web-based voice application. An optimal encryption algorithm will be chosen.

Supported digital certificates

Server authentication is the only method supported between a Web application server hosting a Web-based voice application (using the HTTPS protocol) and the VoiceXML browser. This is because WebSphere Voice Server does not provide a tool to manage the X.509 digital certificates for SSL. For server authentication, the Web application server must have one of the digital certificates based on the X.509 standard. The trusted third parties, or signer certificates, verify the identification of a certificate holder. The certificate holders that are installed with WebSphere Voice Server include:

- Thawte Server CA
- VeriSign Class 1 Public Primary Certification Authority
- Thawte Personal Basic CA
- VeriSign Class 3 Public Primary Certification Authority
- VeriSign Test CA Root Certificates

- VeriSign Class 2 Public Primary Certification Authority
- RSA Secure Server Certification Authority
- Thawte Premium Server CA
- Thawte Personal Freemail CA
- Thawte Personal Premium CA

The digital certificate is used to authenticate the Web server to the VoiceXML browser. During the initial SSL handshake, the Web server supplies the VoiceXML browser with its X.509 certificate. If the VoiceXML browser validates the Web server's certificate, a secure encrypted communication channel is established between the Web server and the VoiceXML browser. You must make sure that the Web server hosting the VoiceXML browser application supports one of the digital certificates listed above.

7.3.4 Unpacking the SDK

To launch the WebSphere Voice Server SDK installation wizard base, run the executable, vssdkinstall_launcher.exe, to unpack the archive. The download executable will present a wizard interface to allow you to extract the main SDK setup in a selected directory as shown in Figure 7-1.

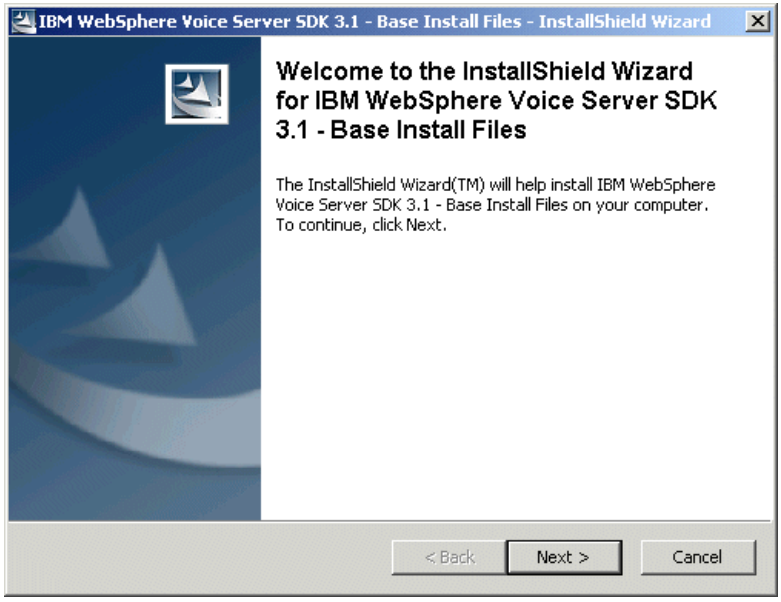


Figure 7-1 Initial SDK unpack wizard window

Perform the following steps:

1. Click **Next**. You will see a window similar to Figure 7-2 on page 331.

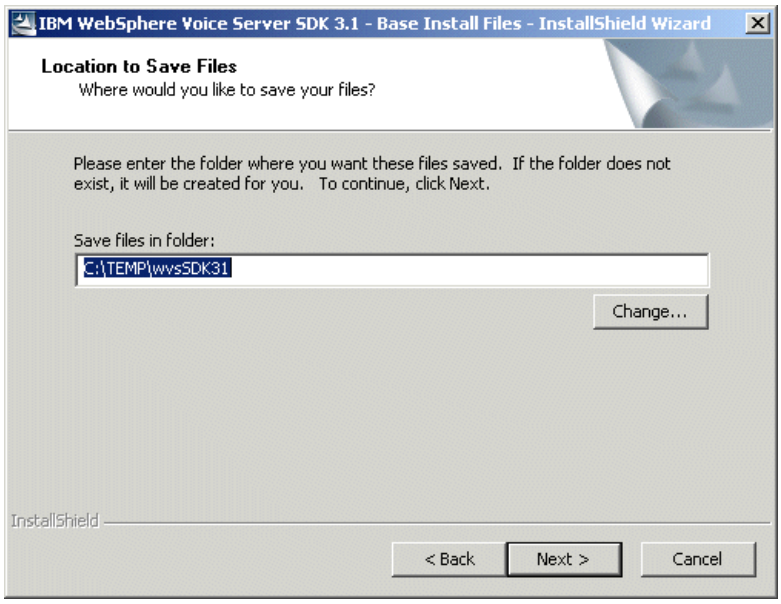


Figure 7-2 Selecting the unpack directory

2. Accept the default location to unpack the SDK installation. Click **Next**.

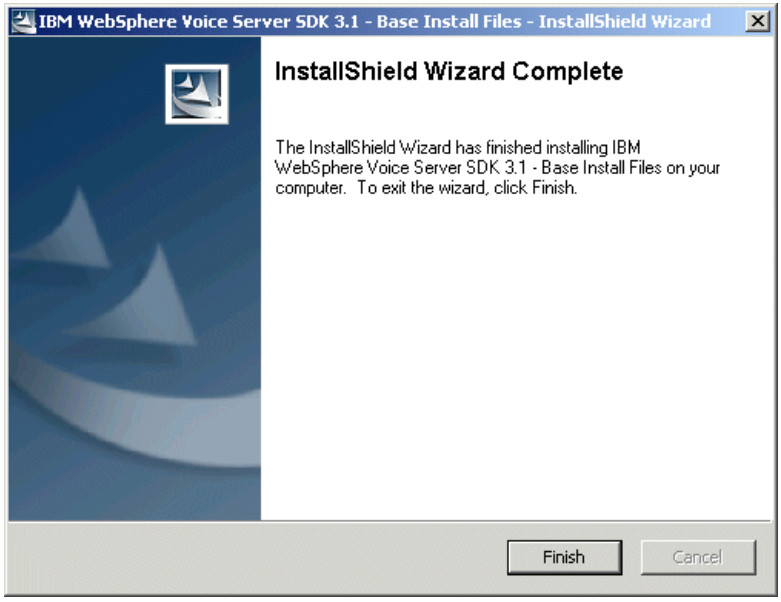


Figure 7-3 Unpacking the SDK complete

3. Click **Finish**.

7.3.5 Unpacking the SDK language

After you have selected the various SDK language executables, run each desired executable to unpack the installation archive. In our example we will run the vssdkinstall_en.exe, vssdkinstall_fr, and vssdkinstall_ptBR.exe files to install the US English, French, and Brazilian Portuguese languages. You must repeat this extraction per language set. The download executable will present a wizard interface to allow you to extract the SDK language setup in a selected directory as shown in Figure 7-4.

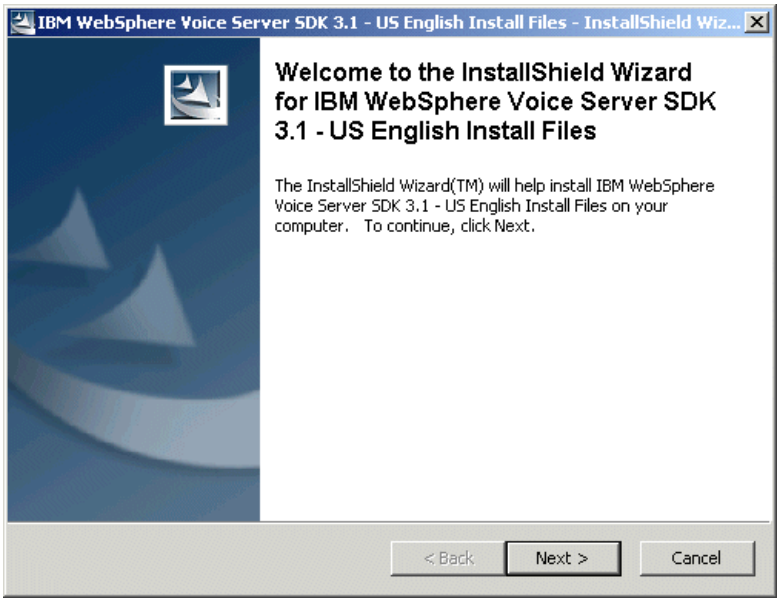


Figure 7-4 Initial SDK language unpack wizard window

Complete the following steps:

4. Click **Next**.

   Extract the language executables in the same temporary directory as you did the vssdk_launcher.exe. The default is the same directory, as shown in Figure 7-5 on page 333. Click **Finish** in Figure 7-6 on page 333 and proceed to either 7.3.7, “Unpacking the concatenative text-to-speech language” on page 339 if you would like to test your application with the CTTS engine or go directly to 7.3.6, “Running the SDK setup” on page 334 to install the SDK.

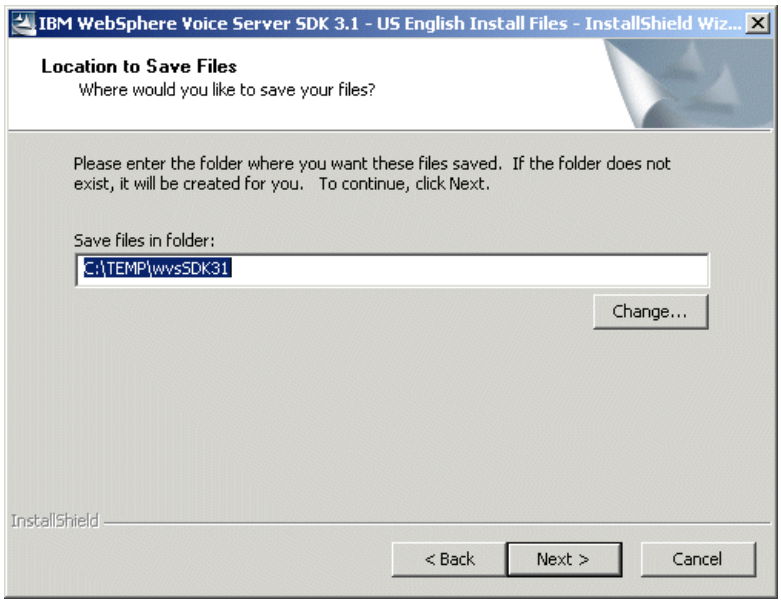


Figure 7-5 Selecting the unpack directory for the language setup

5. Accept the default location and click **Next**.

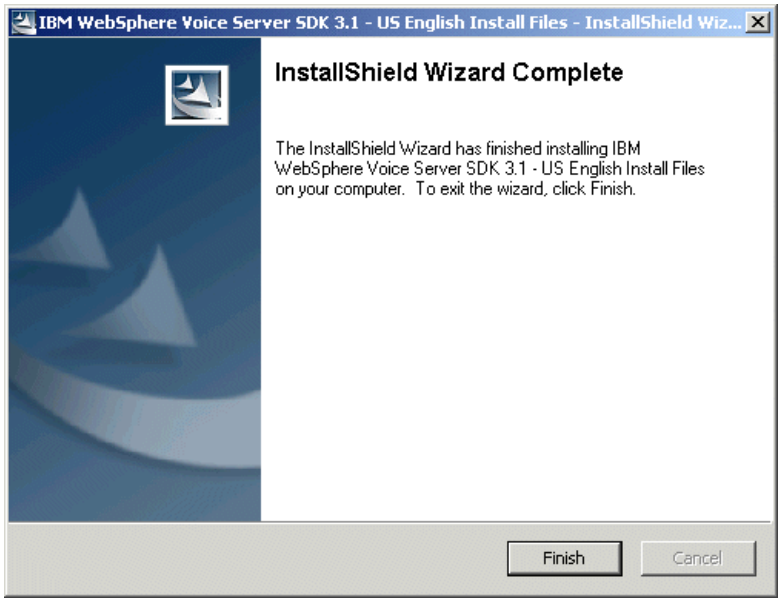


Figure 7-6 Unpacking the language SDK complete

Only after unpacking the main install file (Figure 7-3 on page 331) and at least one language file (Figure 7-6 on page 333), you can then invoke the SDK setup.exe to launch the installation process.

6. Click **Finish**.

7.3.6 Running the SDK setup

The SDK requires the Java Runtime Environment. It is recommended that you install the Java Runtime included with the SDK package if there is no Java Runtime currently installed. The SDK installation will check for the presence and version of the Java Runtime when the SDK setup is run.

1. Run the SDK setup.exe, which is located in the directory where you unpacked the installation package to begin the installation wizard as shown in Figure 7-7.

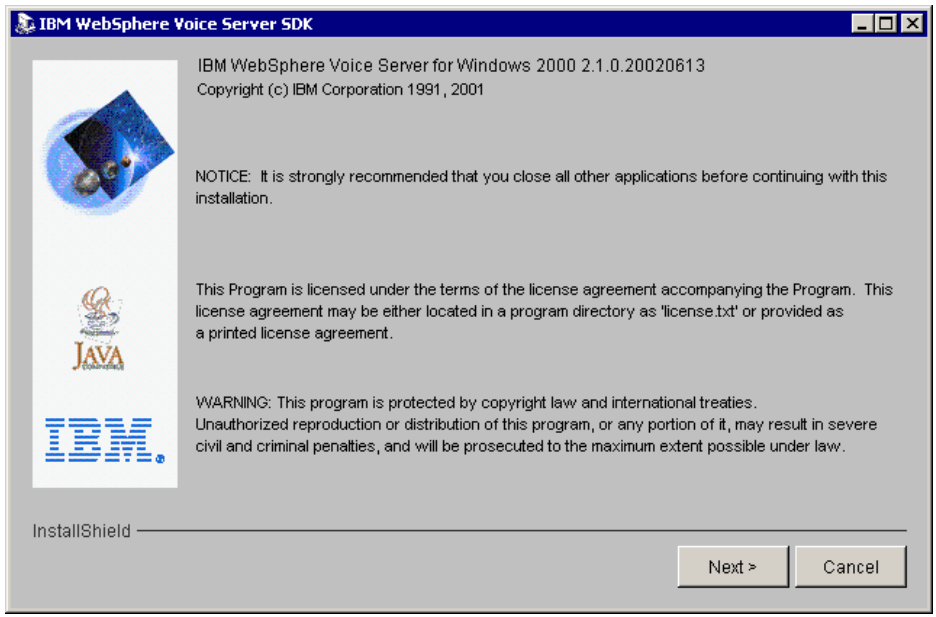


Figure 7-7 Initial SDK setup wizard window

2. Click **Next**. The setup program will evaluate your system for the SDK minimum requirements as shown in Figure 7-8 on page 335.

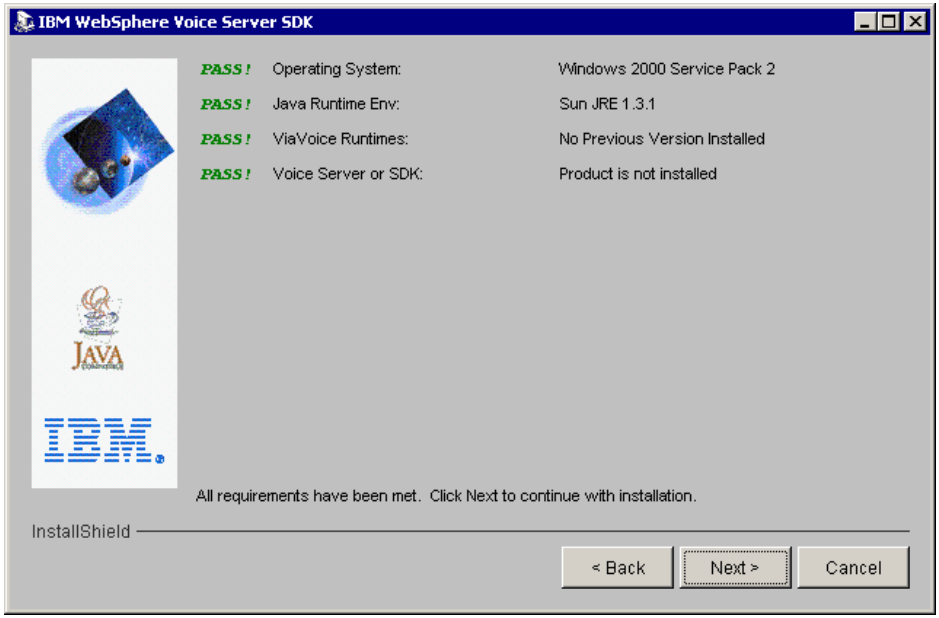


Figure 7-8 Checking the system requirements

3. Click **Next**.

   Once the system requirements have been verified, the SDK Software License Agreement will be displayed as shown in Figure 7-9 on page 336 for you to review and accept the terms. Click **View in Finnish** to see the license in other languages.

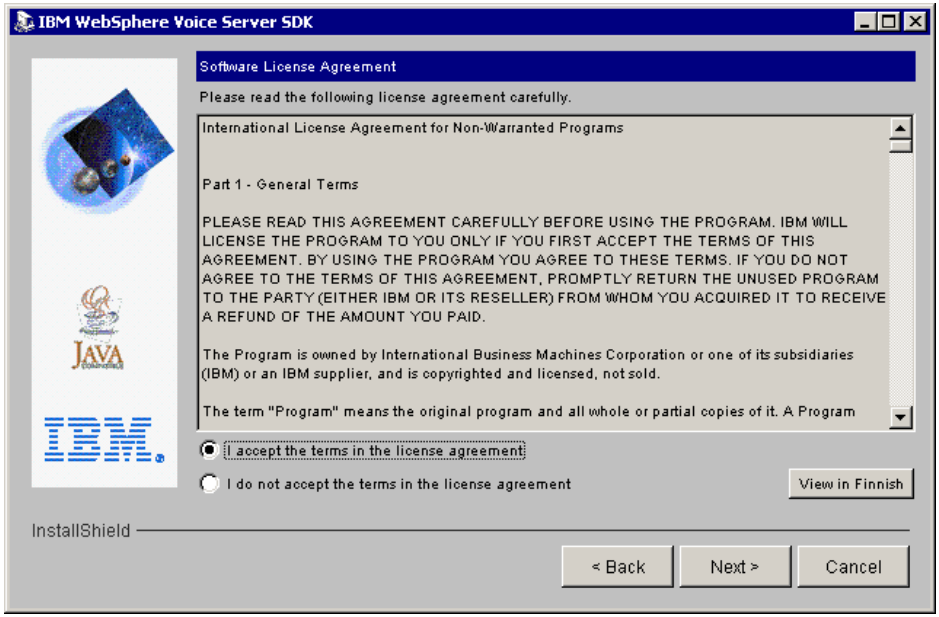


Figure 7-9 SDK software license agreement

4. Click **Next** to continue.

x

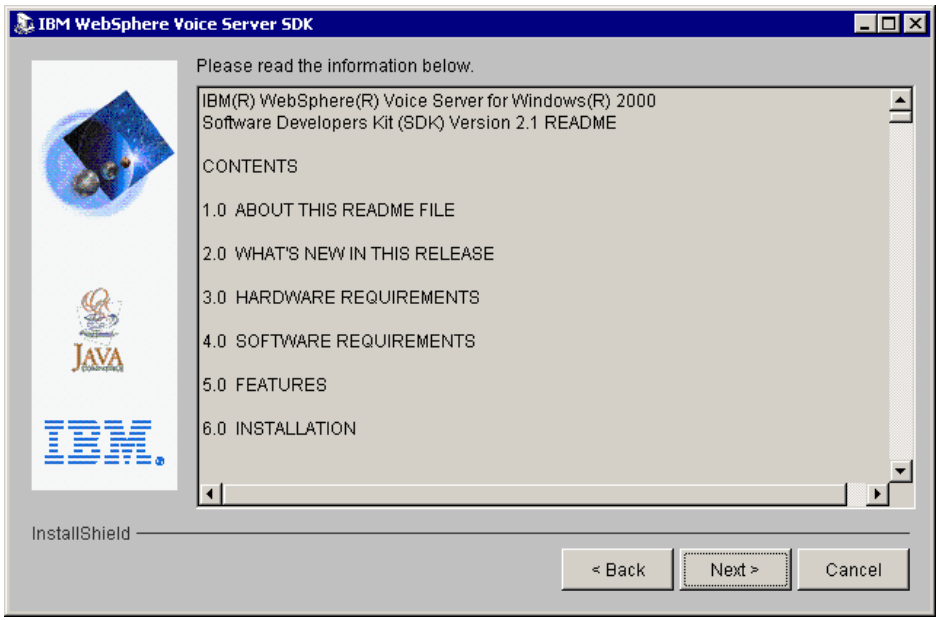


Figure 7-10 SDK README

5. Review the SDK README window. Click **Next**.

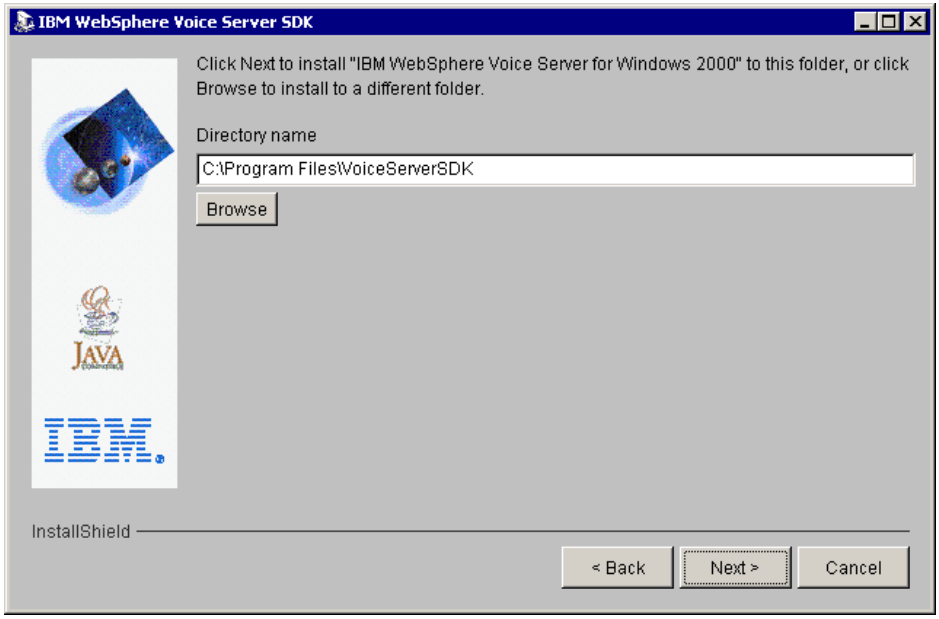


Figure 7-11 Select SDK directory

6. Select the directory to install the SDK. Click **Next**.

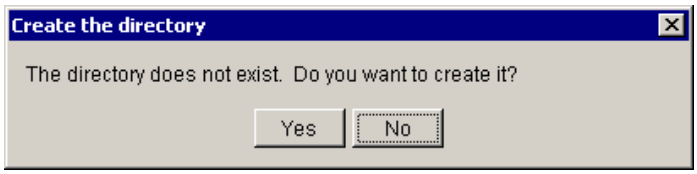


Figure 7-12 Create directory

7. You will be prompted by a question to create a new directory for this program. Click **Yes** as shown in Figure 7-12.
8. Select the languages to use with the SDK as shown in Figure 7-13.

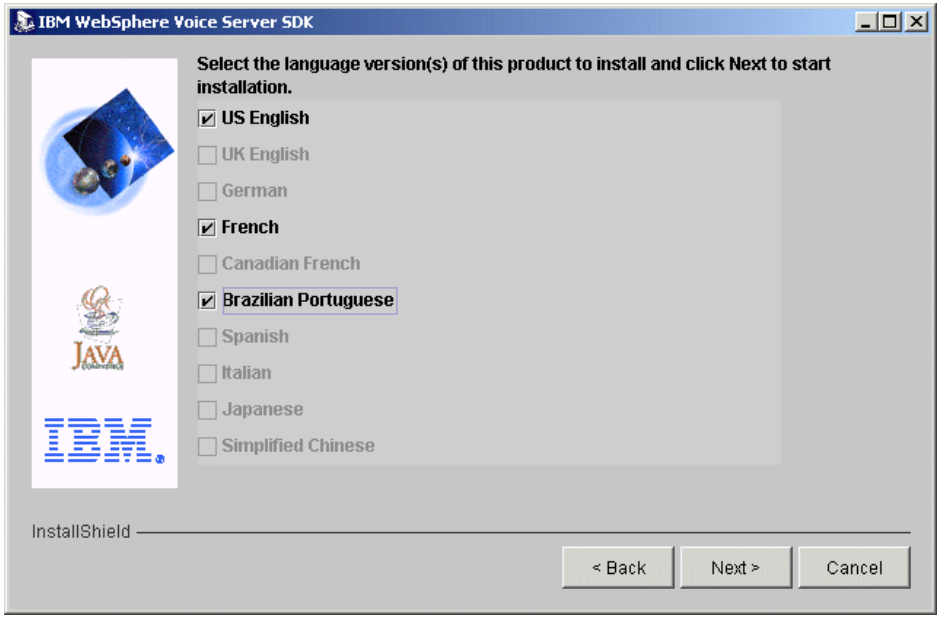


Figure 7-13 Select the SDK languages

9. Click **Next** to continue.

> **Restriction:** Not all languages may be supported by all runtime environments. For example, Voice Server for Japanese is only supported on the Japanese language version of Windows 2000. Please check the platform you are deploying to for the languages supported.

10. When the installation is complete, you will be notified to reboot your system, as shown in Figure 7-14.

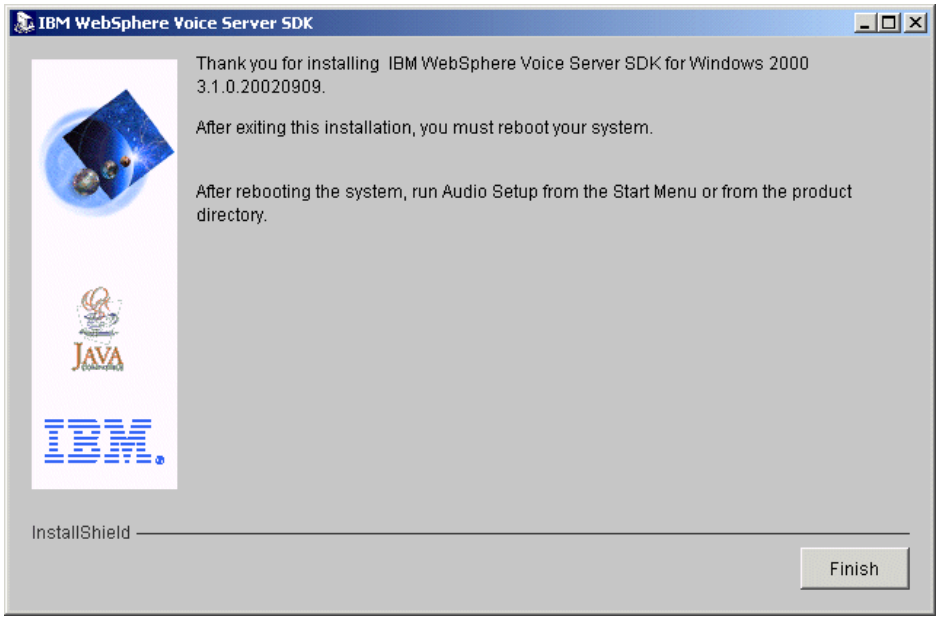


Figure 7-14 SDK installation complete

After rebooting, you have the option of either running the audio setup to complete the installation in 7.3.8, “Audio setup for the SDK” on page 347 or installing the CTTS engine, 7.3.7, “Unpacking the concatenative text-to-speech language” on page 339, prior to the audio setup.

7.3.7 Unpacking the concatenative text-to-speech language

To test your applications with the SDK and the CTTS engine, extract the CTTS Development Edition install files. The CTTS Development Edition requires that the base SDK be installed before the desired CTTS language that is being installed. For example, if the base SDK is installed for French, you can only install the French CTTS Development Edition onto the SDK. Follow the steps in 7.3.4, “Unpacking the SDK” on page 330, 7.3.5, “Unpacking the SDK language” on page 332, and 7.3.6, “Running the SDK setup” on page 334 to extract and install the base SDK and language files.

Concatenative text-to-speech is available in selected languages for the WebSphere Voice Server Version 3.1 deployment platforms, as shown in Table 7-2 on page 340.

Table 7-2 Available languages for Voice Server deployment languages

WebSphere Voice Server deployment platform	Available CTTS languages
WebSphere Voice Response for AIX	Brazilian Portuguese, Canadian French, French, German, Italian, Japanese, Simplified Chinese, Spanish, UK English, and US English
WebSphere Voice Response for Windows	Brazilian Portuguese, Canadian French, French, German, UK English, and US English
Cisco	Japanese
Intel Dialogic	Japanese, UK English, and US English

Since we have extracted the base SDK for US English, French, and Brazilian Portuguese, we will extract the same files from the CTTS Development Edition. Follow these steps to extract the CTTS Development Edition:

- In our example, we will run the ibmvoiceserver_CTTSenUS.exe, ibmvoiceserver_CTTSfrFR and ibmvoiceserver_CTTSptBR.exe files to install the US English, French and Brazilian Portuguese languages. You must repeat this extraction per language set. The download executable will present a wizard interface to allow you to extract the SDK language setup in a selected directory as shown in Figure 7-15 on page 341.

  Please review the hardware and software requirements located in 7.3.4, “Unpacking the SDK” on page 330.

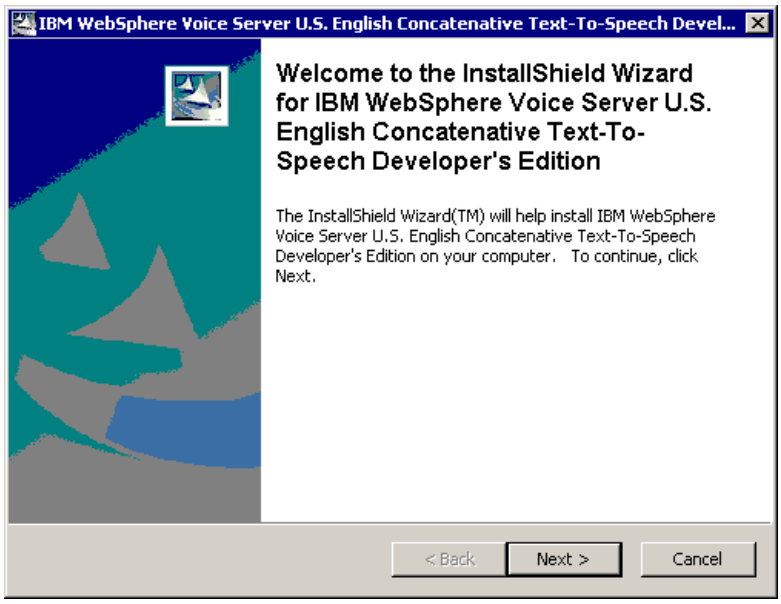


Figure 7-15 Initial CTTS Development Edition unpack wizard window

- The CTTS Development Edition executables install files should be extracted to a different temporary directory from the base (vssdkintall_*.exe) file. The default location is c:\temp\WVSCTTS_., as shown in Figure 7-16 on page 342. Click **Finish** in Figure 7-17 on page 342 to complete the extraction.

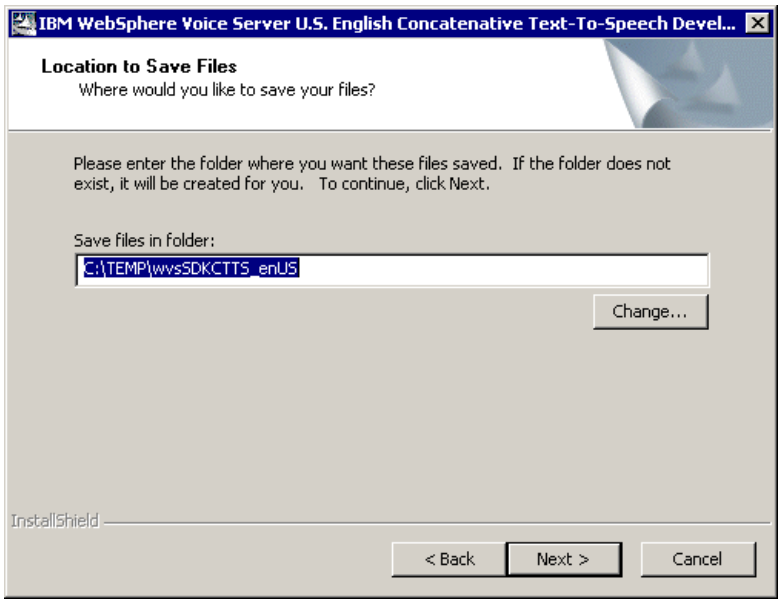


Figure 7-16 Selecting the unpack directory for the CTTS Development Edition setup

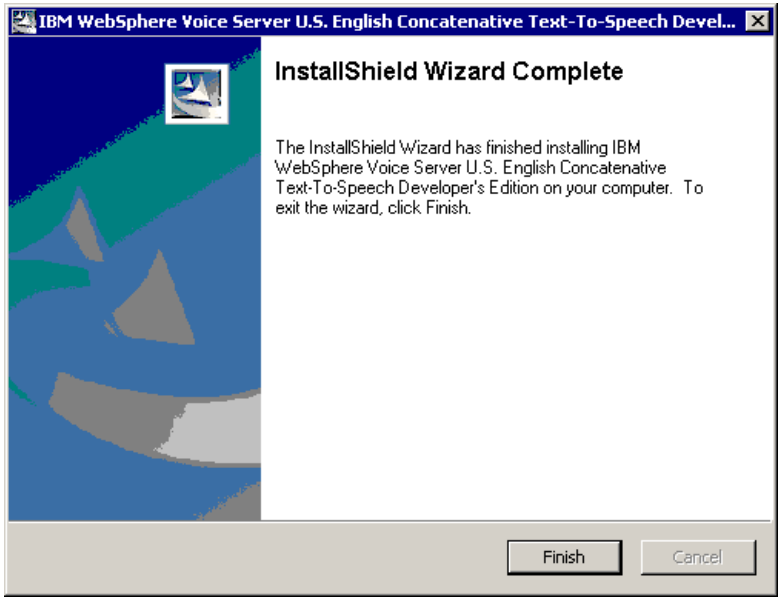


Figure 7-17 Unpacking the CTTS Development Edition complete

- From the temporary directory of the CTTS install file extraction, launch the CTTS installation by executing the setup.exe for each CTTS language. You will see the installation wizard window, asking you to close all of your running applications as shown in Figure 7-18. Click **Next** to continue.

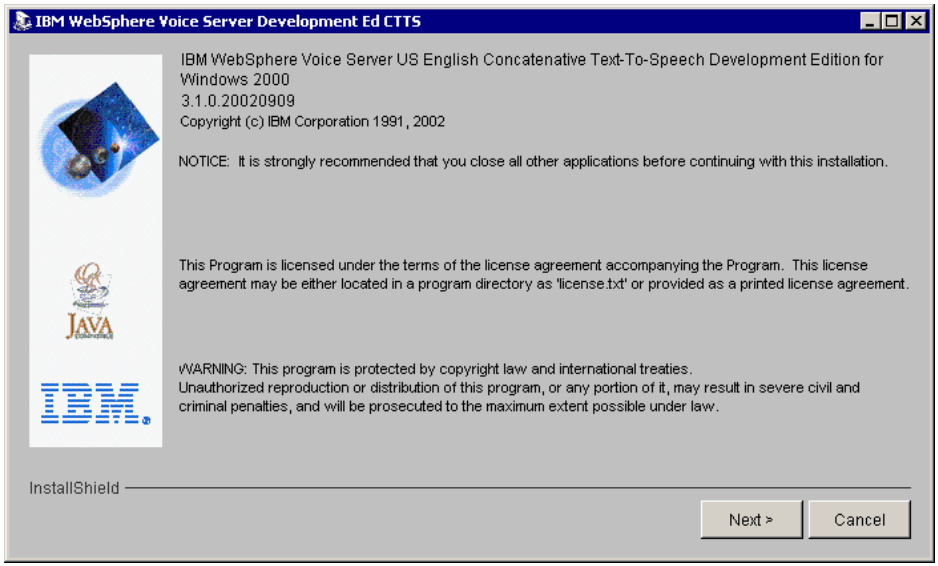


Figure 7-18 Installation wizard prompt to close running applications

- You will be prompted to review the Software License Agreement and agree to the terms in order to continue with the installation, as shown in Figure 7-19 on page 344.

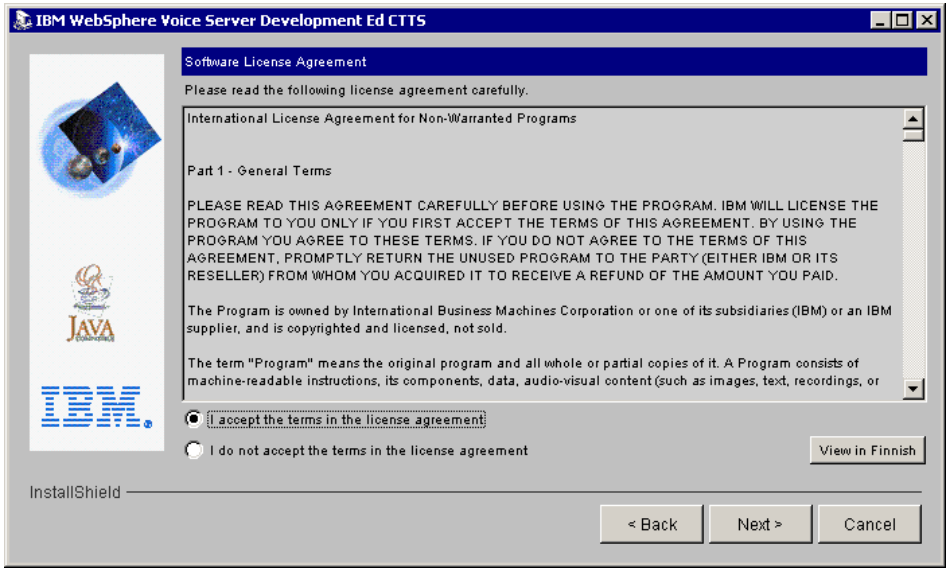


Figure 7-19 CTTS Development Edition software license agreement

- The setup program will evaluate your system for the CTTS minimum requirements as shown in Figure 7-20.

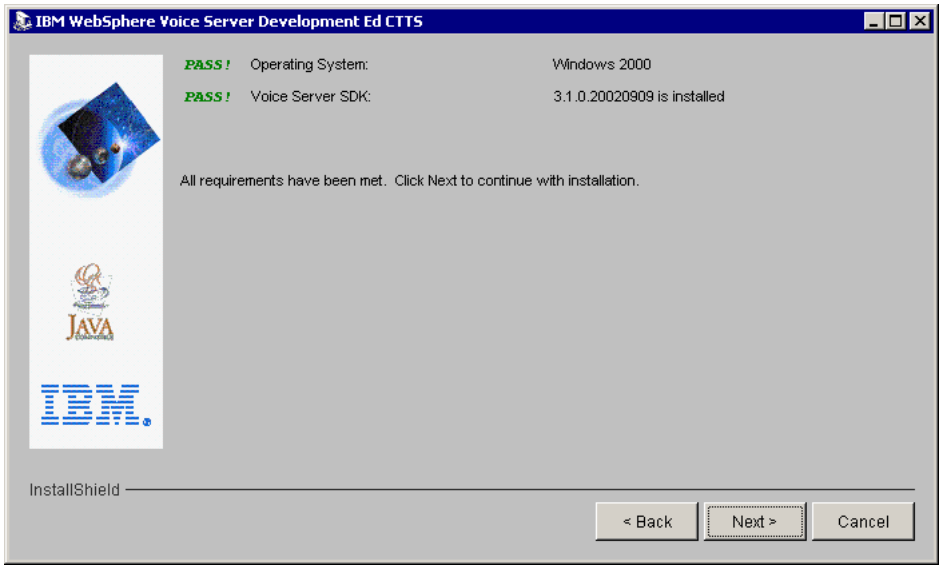


Figure 7-20 Checking the system requirements

- Once the system requirements have been verified, review the SDK README installation notes as shown in Figure 7-21. Click **Next** to proceed.

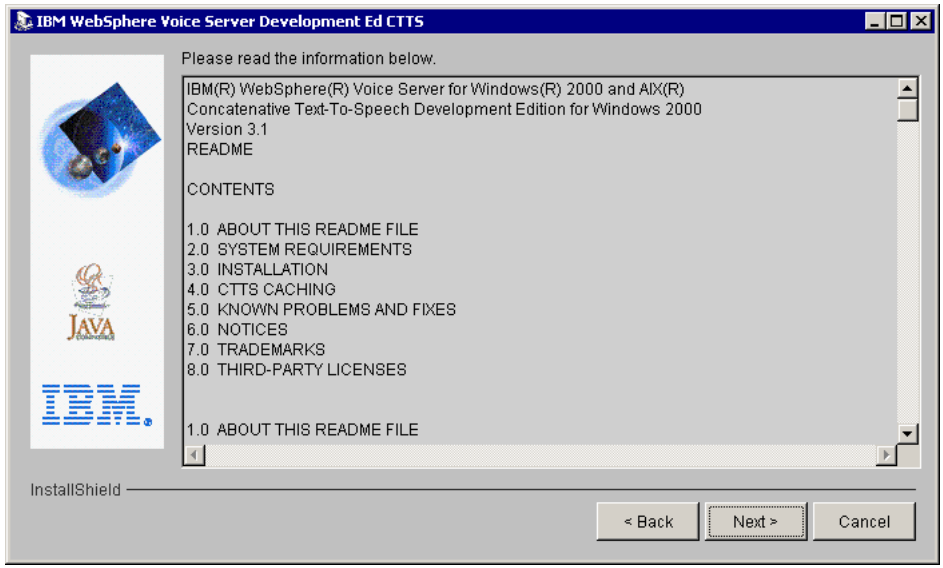


Figure 7-21 CTTS Development Edition README

- You will now be notified that the CTTS installation will commence (Figure 7-22 on page 346). Click **Next** to begin the installation.

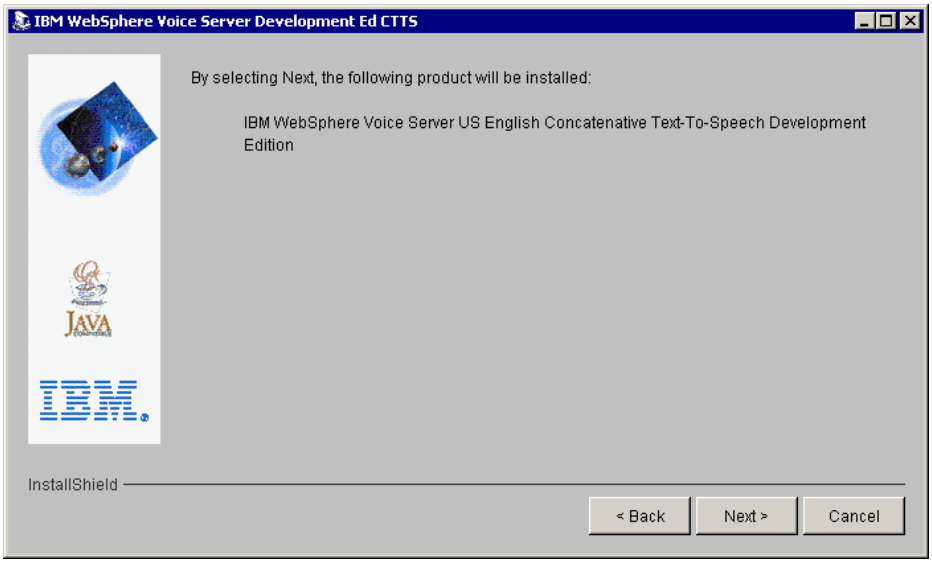


Figure 7-22 Installation notification

- When the installation is complete, you will be notified to reboot your system. Click **Finish** and reboot, as shown in Figure 7-23.

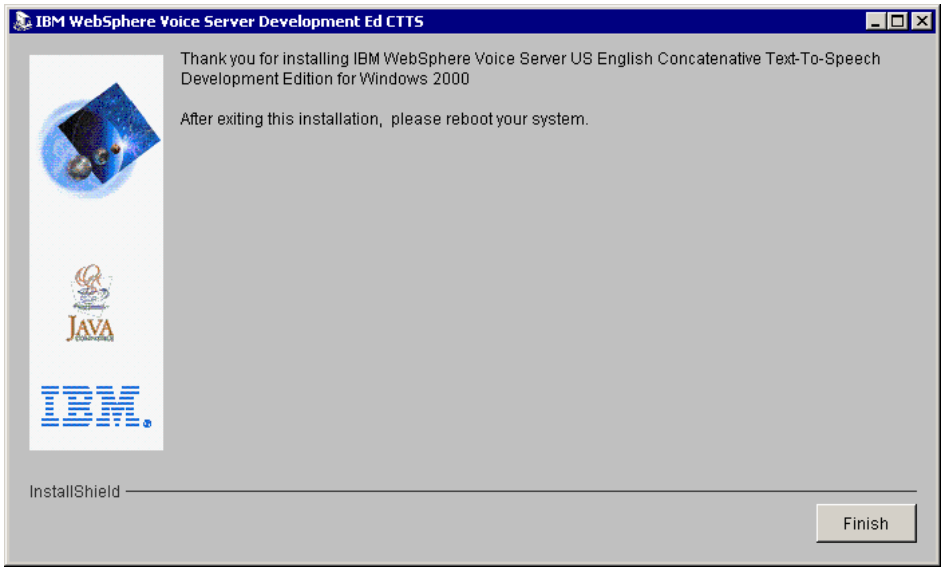


Figure 7-23 CTTS Development Edition installation complete

7.3.8 Audio setup for the SDK

Perform the following steps for the audio setup for SDK:

1. Run the audio setup from **Start -> Programs -> IBM WebSphere Voice Server SDK -> Audio Setup-** ”language, i.e US English” to calibrate the microphone and speaker volume for your hardware as shown in Figure 7-24. The audio may be adjusted in any of the installed languages.

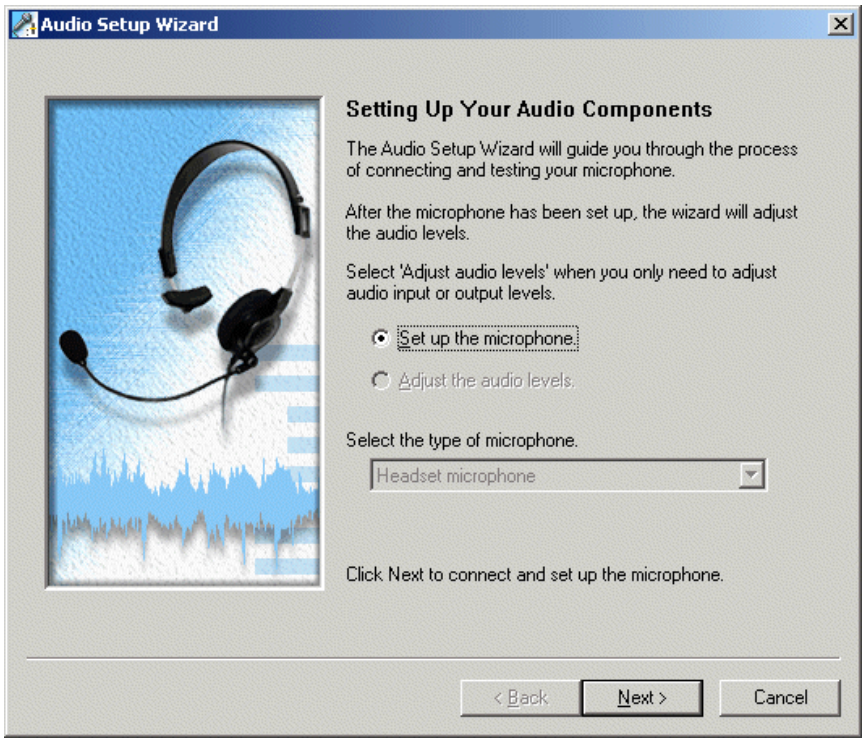


Figure 7-24 Starting audio calibration

2. The audio can be configured for either output to speakers or output to a headset as shown in Figure 7-25 on page 348.

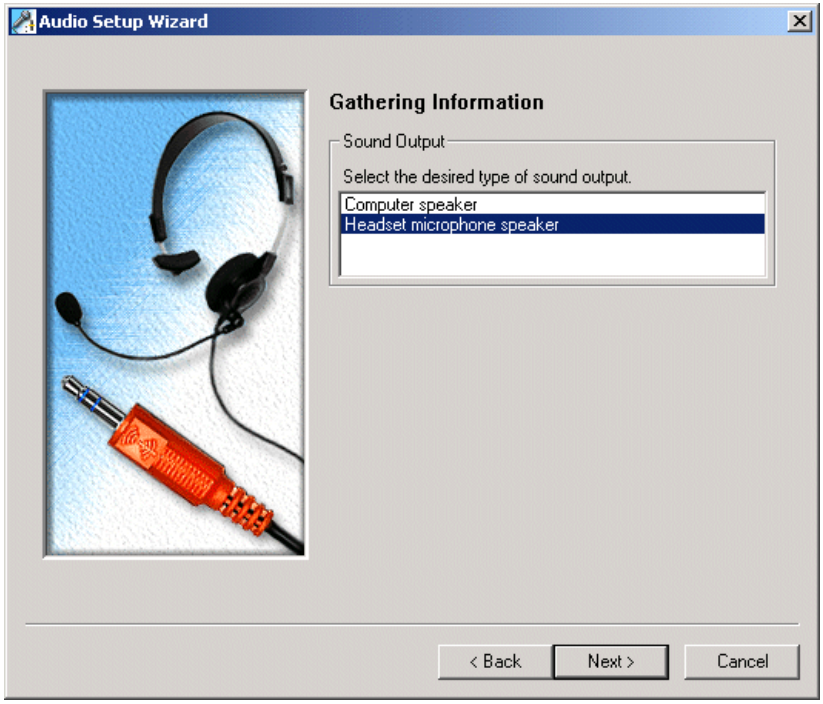


Figure 7-25 Selecting the output type

3. Several instruction windows will be displayed to show you how to install your audio input and output devices as shown in Figure 7-27 on page 350.

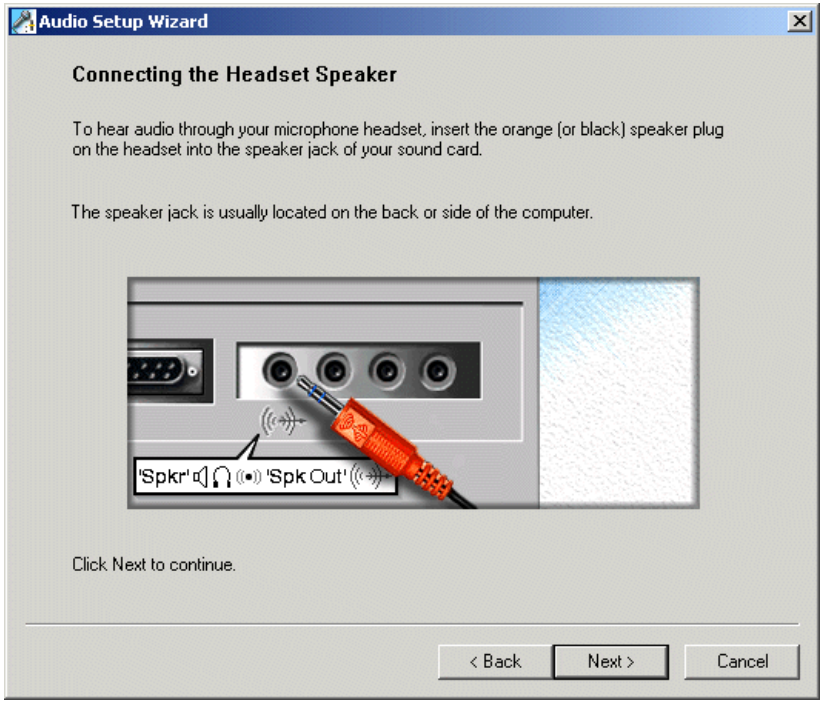


Figure 7-26 Headset speaker connection instructions

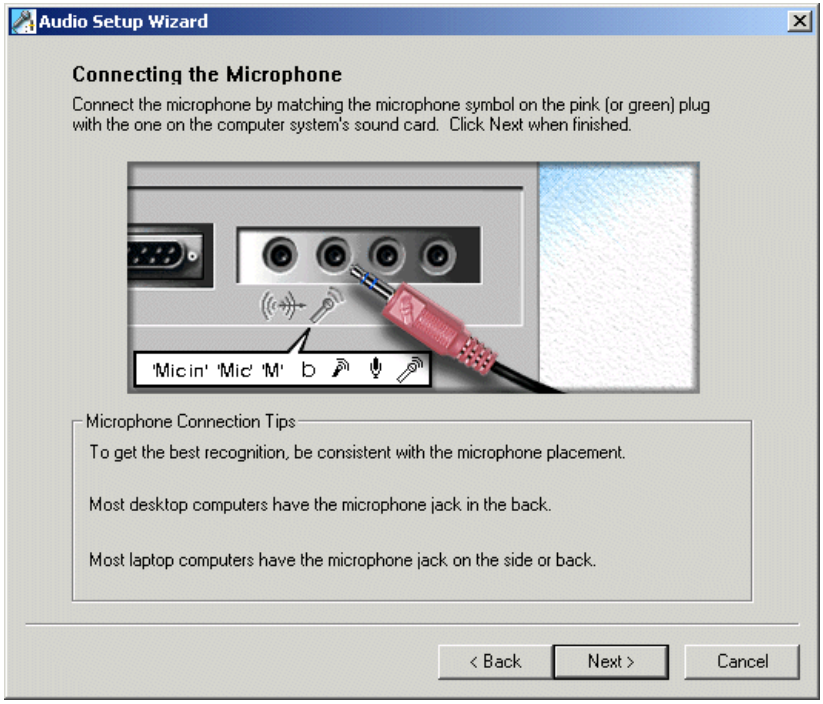


Figure 7-27 Audio connection instructions

4. Position and adjust the headset as shown in Figure 7-28 on page 351.

Figure 7-28 Adjust your headset

5. Calibrate the audio playback as shown in Figure 7-29 on page 352.

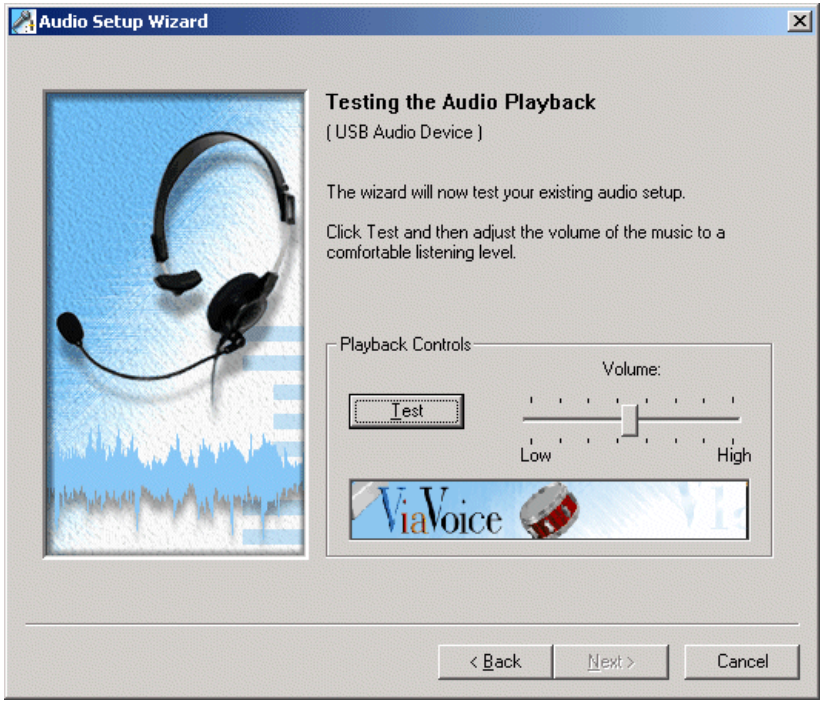


Figure 7-29 Playback calibration

6. Test the microphone as shown.

   The test will record background room noise as shown in Figure 7-30 on page 353.

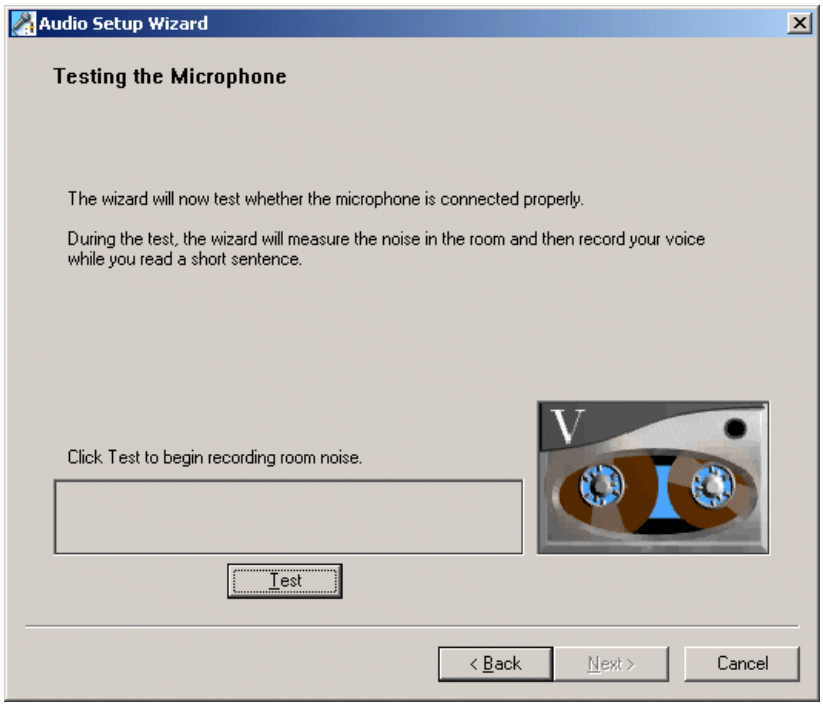


Figure 7-30 Testing the microphone for assessing background room noise

- This test will check whether your microphone is connected properly by asking you to read a sentence, as shown in Figure 7-31 on page 354.

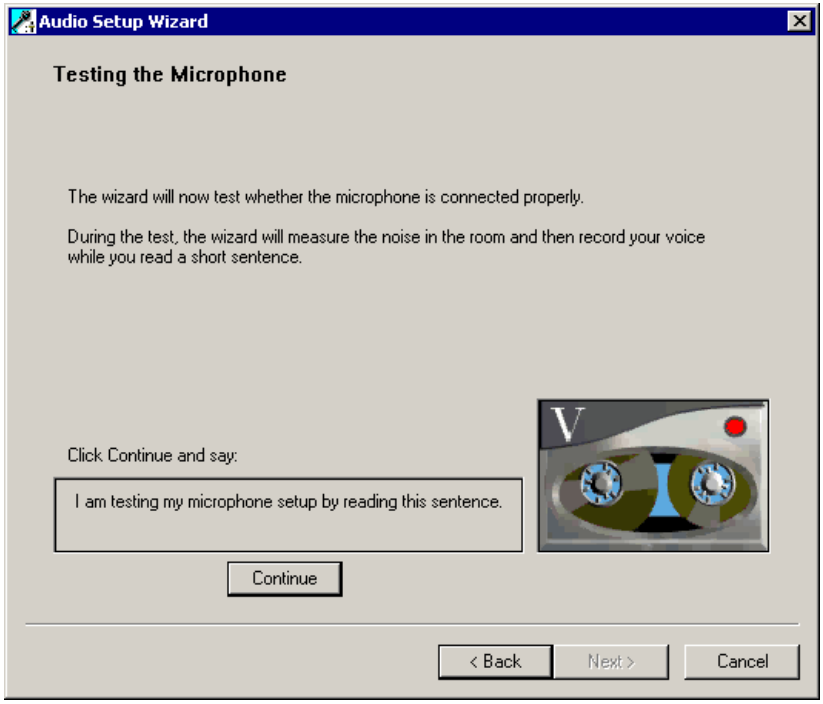


Figure 7-31 Testing the microphone by recording a sentence

7. Click **Continue** and say the statement. Click **Next** to continue.

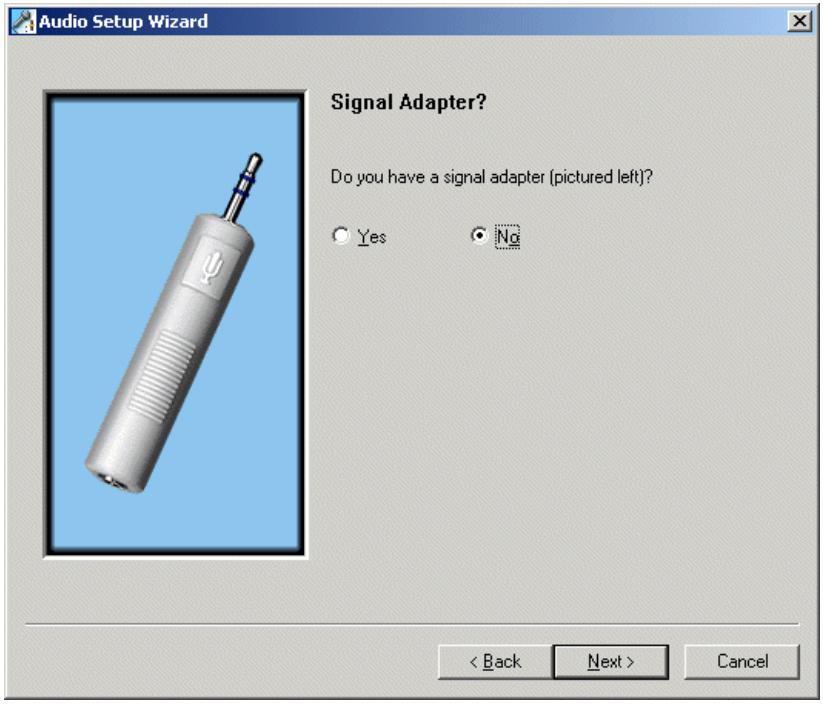


Figure 7-32 Signal adapter

8. If you are using a signal adapter, such as those use with earlier IBM ThinkPads, enable support for the adapter as shown in Figure 7-32. Click **Next**.
9. After successfully completing the microphone setup, you will see the following prompt shown in Figure 7-33 on page 356, click **Next** and continue to the last step.

Figure 7-33 Microphone connection complete

10. Once the basic microphone and speaker, or headset, connections have been tested, a microphone level calibration window will be presented as shown in Figure 7-34 on page 357. Read the paragraph and click **Finish** to exit the audio setup.

Figure 7-34 Reading the calibration paragraph

This completes the installation of the Voice Server SDK 2.1.

7.3.9 Starting the VoiceXML browser

To use the WebSphere Voice Server SDK, you start the VoiceXML browser and pass it the URL of the first VoiceXML document in your application.

The VoiceXML applications will generally reside on a Web application server, although they could reside on your desktop workstation during early testing. The VoiceXML browser uses HTTP, over a LAN or the Internet, to fetch the documents.

VoiceXML is an extension to the XML specification designed for creating audio windows that feature synthesized speech, digitized audio, recognition of spoken and DTMF key input, recording of spoken input, telephony, and mixed-initiative conversations.

7.3.10 Testing the application in text mode

You can test your code in text mode if you have the IBM WebSphere Voice Server for Windows 2000 SDK (Version 3.1) installed. It is a good idea to test your code in text mode before testing in audio mode because you don't need to worry about the potential for recognition errors. To test in text mode from the Voice Toolkit, do the following steps:

1. Unless you have a reason not to do so, select **File -> Save all** to make sure you have saved any changes in all of your files (only saved versions of files are used in the test).
2. If open, select the VoiceXML file from which you want to start the test. Otherwise, open the desired VoiceXML file.
3. Select **Run -> Run in text mode**.
4. When the browser starts the DTMF (phone pad) simulator, give the command window focus.
5. Once you see a prompt from the browser, type any phrase you want to test. If the browser posts a message that appears in the middle of your phrase while you are typing, just ignore it and keep typing until you finish the phrase. After you finish the phrase, press Enter to send it to the browser.
6. Continue typing test phrases until you finish your test. To stop the browser, press Ctrl+C.

To test your application using the SDK, you can start the VoiceXML browser in the following way:

- Command line interface
  - To start the VoiceXML browser, use the following command:

    ```
    %IBMVS%\bin\batchfile URL
    ```

  - Where `URL` is the initial URL for your VoiceXML application and `batchfile` is the following as shown in Figure 7-35 on page 359:

    `vstext_<locale>` - When the VoiceXML application runs, the VoiceXML browser writes prompts and other output as text in the Java console window, and accepts input from your keyboard or the DTMF Simulator. If desired, you can provide input from a file by specifying a <inputfile> parameter. This mode is useful for automated testing purposes or if a microphone is not available on your system.

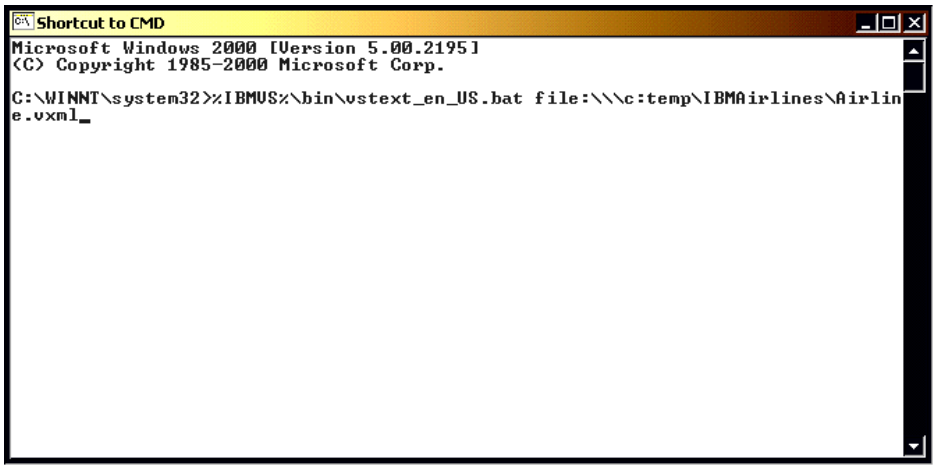


Figure 7-35 Command-line interface for vstext

7.3.11 Testing the application in audio mode

Once you are satisfied with your application's behavior in text mode, you should test your application in audio mode. As with testing in text mode, you must have the IBM WebSphere Voice Server for Windows 2000 SDK (Version 3.1) installed to test in audio mode. You must also have a microphone and speakers (or microphone headset) installed and configured.

It's a good idea to set the audio level before starting the test. To do this, click **Start -> Programs -> IBM WebSphere Voice Server SDK -> Audio Setup**, and follow the on-screen instructions.

To run your applications using the SDK in audio mode, you can start the VoiceXML browser in the following way:

- Command-line interface
  - To start the VoiceXML browser, use the following command:

    `%IBMVS%\bin\batchfile URL`

  - Where `URL` is the initial URL for your VoiceXML application and `batchfile` is one of the following as shown in Figure 7-36 on page 360:

    `vsaudio_<locale>` - When the VoiceXML application runs, the VoiceXML browser generates spoken output (using text-to-speech and/or recorded audio) and accepts spoken input and simulated DTMF input entered using the DTMF Simulator included in the IBM WebSphere Voice Server SDK.

Note: <locale> is one of the following .bat files:

- de_DE for German
- en_GB for UK English
- en_US for US English
- es_ES for Spanish
- fr_CA for Canadian French
- fr_FR for French
- it_IT for Italian
- ja_JP for Japanese
- pt_BR for Brazilian Portuguese
- zh_CN for Simplified Chinese

Figure 7-36 Command line interface for vsaudio

- To test in audio mode from the Voice Toolkit:
  a. Unless you have a reason not to do so, select **File -> Save all** to make sure you have saved any changes in all of your files (only saved versions of files are used in the test).
  b. If open, select the VoiceXML file from which you want to start the test. Otherwise, open the desired VoiceXML file.
  c. From the Voice Toolkit's menu bar, select **Run -> Run in audio mode**.
  d. Once you hear a prompt from the browser, say any phrase you want to test. The phrases provide the application input, and the output is spoken via the selected sound output device. While the application is running a text trace is also displayed

e. Continue saying test phrases until you finish your test. To stop the browser, give the command window focus, and press Ctrl+C.

7.3.12 Sample VoiceXML applications

The WebSphere Voice Server SDK includes sample VoiceXML applications . The samples are located in subdirectories of %IBMVS%/samples/< locale> (where %IBMVS% is an environment variable that contains the pathname of the installation directory, and where < locale>is en_US for US English For language versions other than US English, see the appropriate appendixes.) You can also run the Audio Sample from the Windows Start menu by choosing **Programs -> IBM WebSphere Voice Server SDK -> Audio Sample**, and then selecting the Audio Sample program for the desired language.

Note: The GrammarBuilder sample requires that you have a version of the IBM DB2 Universal Database in the same language because the database and table names are different for some language versions of DB2. For example, the US English GrammarBuilder sample will fail if you try to use it to access the French EXEMPLE database because it is expecting the US English database called SAMPLE. You can download a fully functional trial version of DB2 in the desired language from:

http://www.ibm.com/software/data/db2/udb/downloads.html

8

VoiceXML application development using Voice Toolkit 3.1

IBM WebSphere Voice Toolkit 3.1 helps developers to create voice applications using a VoiceXML application development environment. WebSphere Voice Toolkit features grammar and VoiceXML editors as well as a pronunciation builder so that application developers can develop personally tuned voice applications in a minimal amount of time.

8.1 Voice Toolkit

WebSphere Voice Toolkit works with WebSphere Voice Server for application deployment, WebSphere Voice Server SDK 3.1 for application testing, and the IBM Reusable Dialog Components for additional VoiceXML enhancements to voice applications.

The IBM WebSphere Voice Toolkit includes:

- Integrated development environment (IDE)
- VoiceXML editor and debugger
- Grammars editor and test tool
- Java Speech Grammar Format (JSGF)
- Speech Recognition Control Language (SRCL) (a variant of the Backus-Naur Form or BNF grammar format)
- Pronunciation Builder
- Basic Audio Recorder
- VoiceXML Reusable Dialog Components(RDCs)
- Wizard to use and customize the RDCs
- Samples

New features in Voice Toolkit 3.1

- Separate Help window. When you open **Help > Help Contents**, the window opens independently, so that you can continue to work in the Voice Toolkit while reading help topics at the same time.
- Ability to create pronunciations from audio files or microphone recordings.
- Ability to create audio files with the Audio Recorder by typing a text phrase or by providing a text file to synthesize, as well as by using a microphone.
- Ability to convert a Nuance Grammar Specific Language (GSL) grammar file to a JSGF file.
- Speech Recognition Grammar specification (SRG XML) grammar format for use in speech recognition.
- Compiled VoiceXML grammars will be supported in the Voice Toolkit by JSGF and SRGF grammars.
- Analysis tools to examine recognition log files (Voice Server event data from the WebSphere Voice Response) and the audio quality of audio files.
- Telephony tools to test the speech recognition and the text-to-speech engines from the WebSphere Voice Server for WebSphere Voice Response.

- New Reusable Dialog Components, including stocks, airports, US cities, and prerecorded audio files.
- New reference pages for customizing VoiceXML and grammar editors.

8.1.1 WebSphere Voice Toolkit prerequisites

Listed in this section are the hardware and software requirements for WebSphere Voice Toolkit:

- Hardware requirements
  - Minimum of a 500 MHz Intel Pentium processor or an equivalent
  - 256 MB of RAM (512 MB recommended)
  - 275 MB hard disk space
  - 350 MB temporary space on the drive specified in the user TMP variable (200 MB required if installing from a CD)
  - Environment variable (not required if installing from a CD)
  - 500 MB additional hard disk space for Web download
  - Sound card and quality microphone recommended to test applications
- Software requirements
  - Microsoft Windows 2000 (Service Pack 2 recommended)
  - WebSphere Voice Server SDK 3.1 is strongly recommended to take full advantage of the WebSphere Voice Toolkit functionality

8.1.2 WebSphere Voice Toolkit installation

To install the Voice Toolkit, do the following:

1. Run the setup.exe file found in the installation directory. This will start the installation wizard shown in Figure 8-1 on page 366. The Voice Toolkit should be installed after the WebSphere Voice Server SDK 3.1. Although the SDK is not required to create VXML applications, there are enhanced functions that will not be present without the use of the IBM speech engines provided in the SDK.

   Note: It may take some time before the first window of the Voice Toolkit setup appears, depending on your disk speed and CPU speed.

2. To launch the WebSphere Voice Toolkit 3.1, run the executable file, Voicetoolkit_setup.exe. The wizard will present you with a welcome window as shown in Figure 8-1 on page 366.

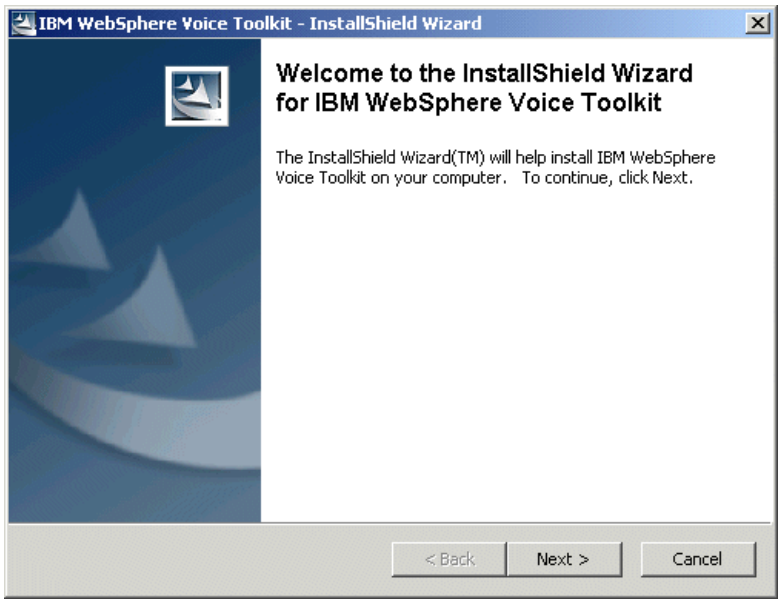


Figure 8-1 Installing the Voice Toolkit

3. You will be notified to close out all of your existing applications in order to have a smooth install. Once the applications have been closed, click **Next** as shown in Figure 8-2 on page 367.

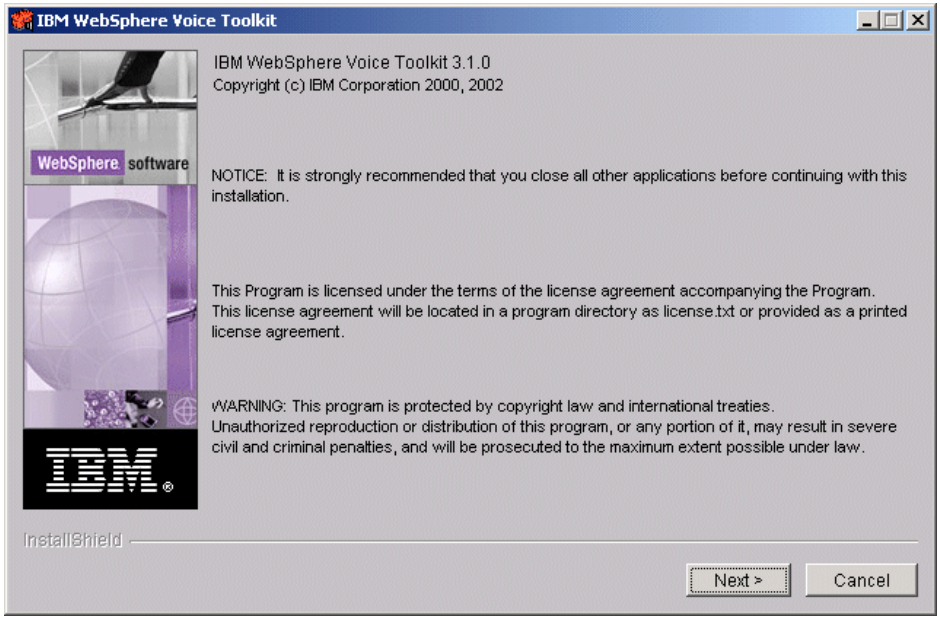


Figure 8-2 Close all other applications notice

4. Review and agree to the software license agreement as shown in Figure 8-3 on page 368.

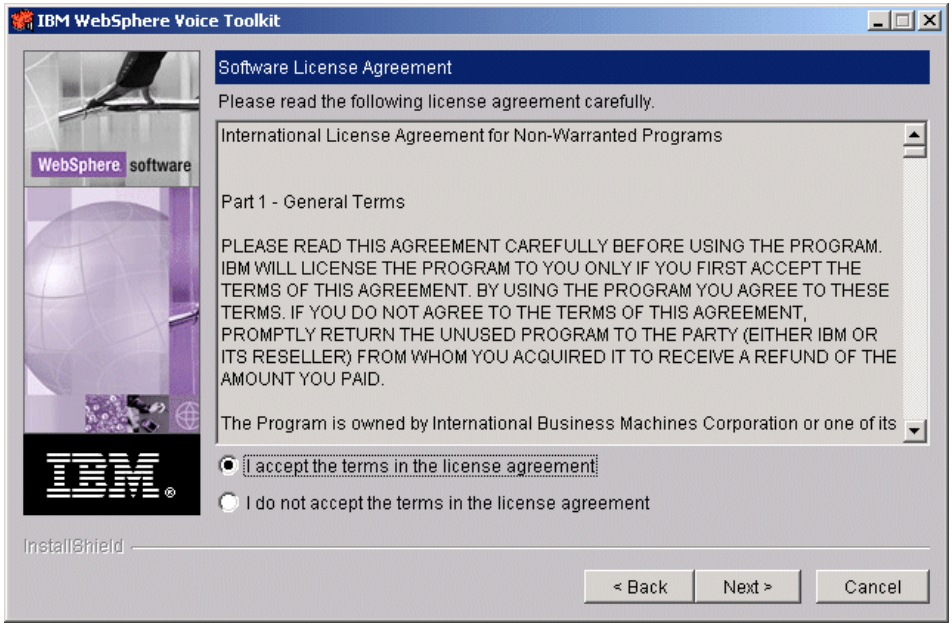


Figure 8-3 License agreement

5. Review the latest information regarding hardware and software prerequisites, as well as installation instructions about the Voice Toolkit 3.1 as shown in Figure 8-4 on page 369.

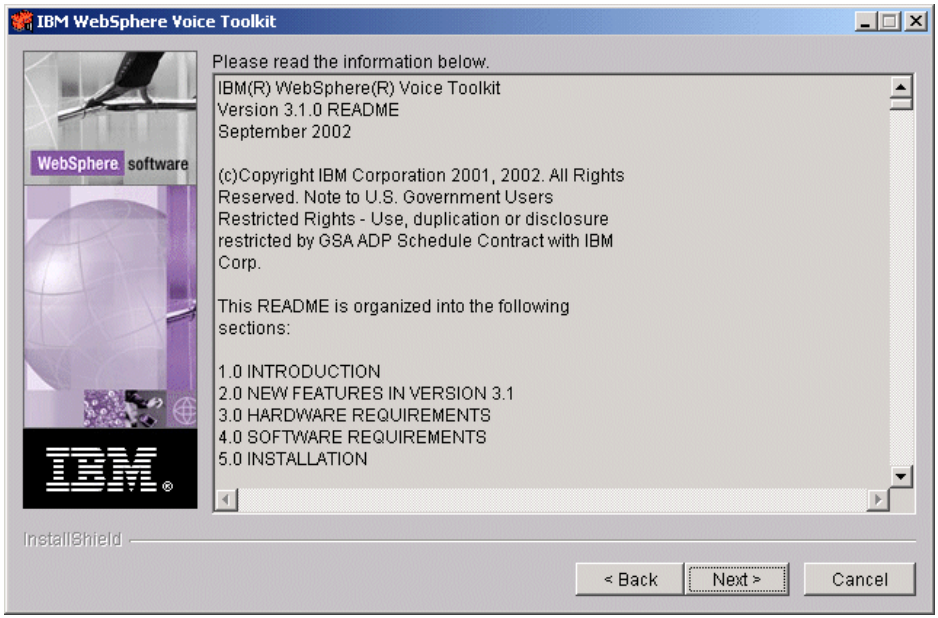


Figure 8-4 Voice Toolkit README

6. Select the directory to install the Voice Toolkit, as shown in Figure 8-5 on page 370.

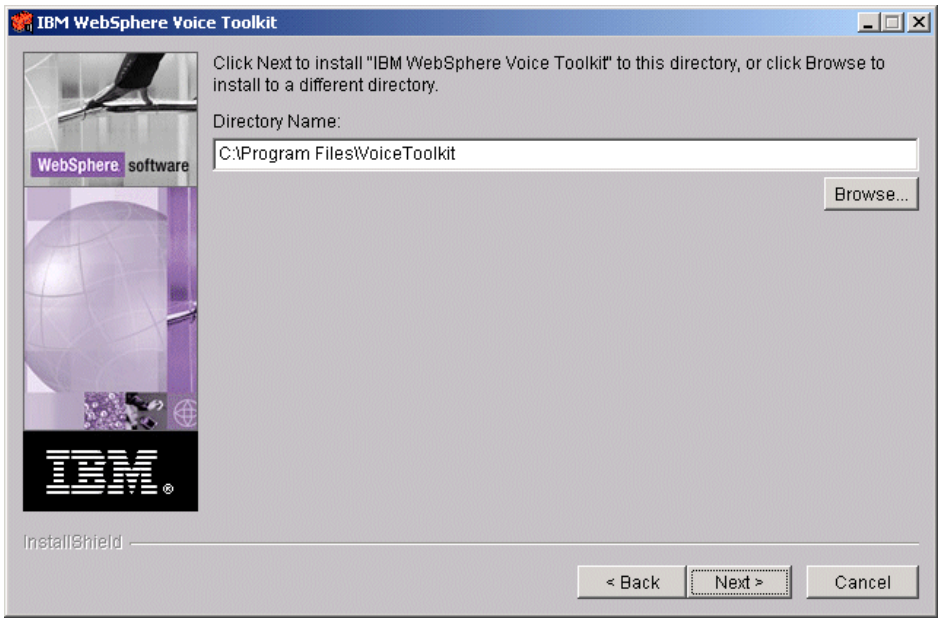


Figure 8-5 Select the Voice Toolkit directory

7. If you are installing the Voice Toolkit in a new directory, you will be prompted agree to create this new path, as shown in Figure 8-6.

Figure 8-6 Create directory

8. When the installation is complete, it is recommended that you reboot your system, as seen in Figure 8-7 on page 371.

Figure 8-7 Voice Toolkit installation complete

8.1.3 Voice Toolkit settings

Perform the following steps to review the Voice Toolkit settings:

1. Launch the Voice Toolkit by clicking **Start -> Programs -> IBM WebSphere Voice Toolkit -> IBM WebSphere Voice Toolkit.** The Toolkit will open up to the main welcome page as shown in Figure 8-8 on page 372.

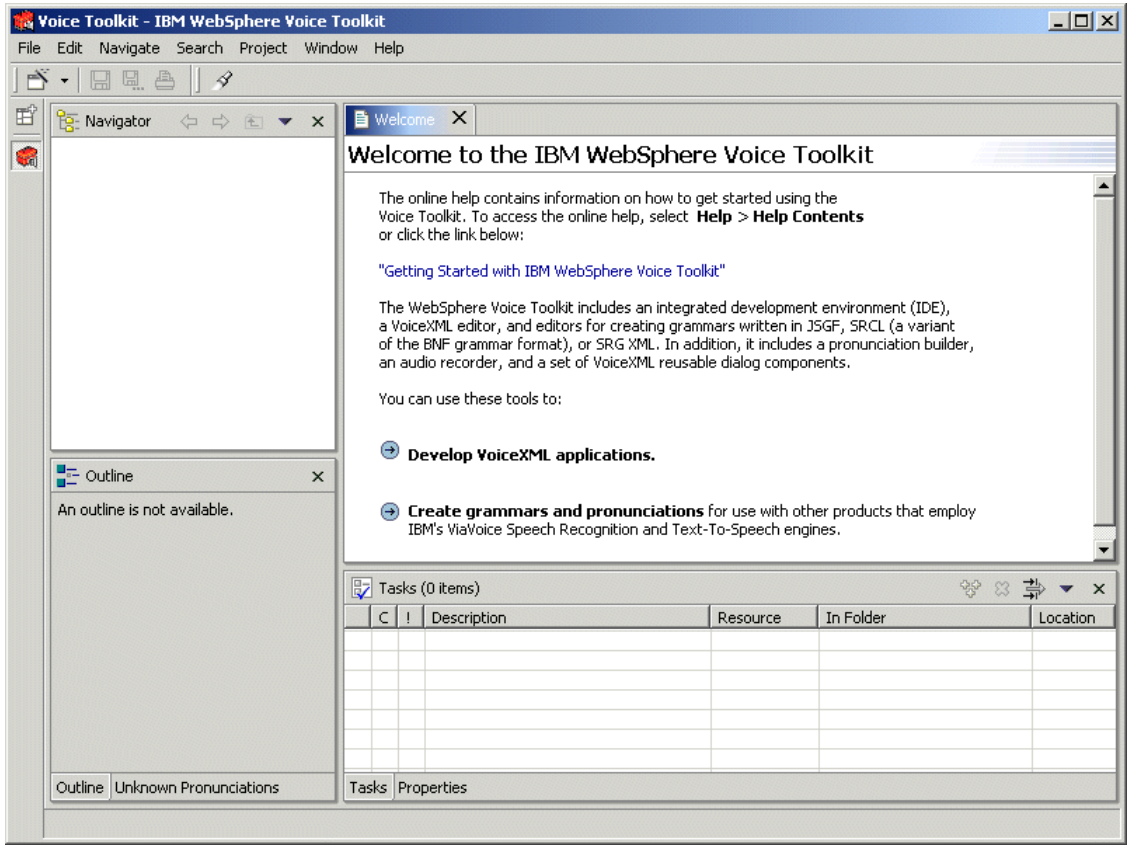


Figure 8-8 Voice Toolkit welcome window

2. The Voice Toolkit settings are found by clicking **Window -> Preferences**. This window will allow you to configure the Voice Toolkit after installation. When the Voice Toolkit is first installed, the platform default language is used as the Voice Toolkit default language. The language can be redefined by selecting a new language on the Voice Toolkit setting node as shown in Figure 8-9 on page 373.

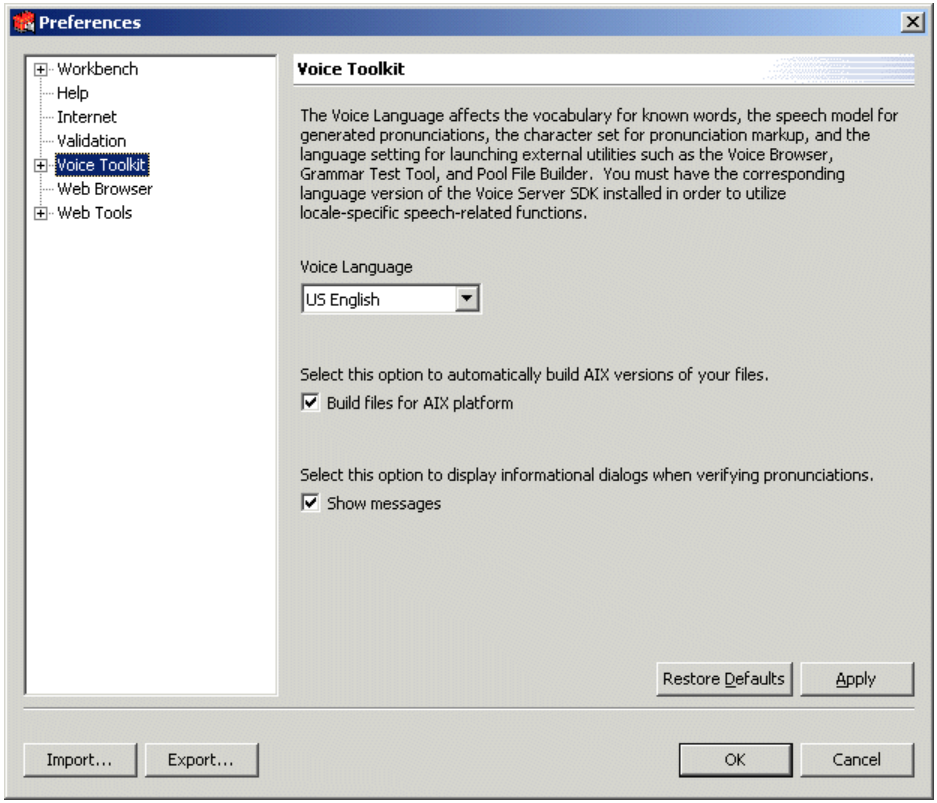


Figure 8-9 Voice Toolkit settings

Tip: If you are not deploying to a DirectTalk platform, you may wish to disable the Build files for AIX platform option, since it will consume extra time when manipulating grammars.

8.1.4 VoiceXML editor

The Voice Toolkit editor provides syntax coloring and prompting. Content assist shows valid elements and attributes, and code formatting facilities allow you to format your document or active elements. The Pronunciation feature will allow you to edit and test any words that may not be recognized by the speech engine.

Creating a project

The Voice Toolkit provides a fully integrated editing and debugging environment (IDE) based on the WebSphere Studio interface.

To create a new project, do the following:

1. From within the Voice Toolkit IDE (Figure 8-10) select **File -> New -> Voice Project**.

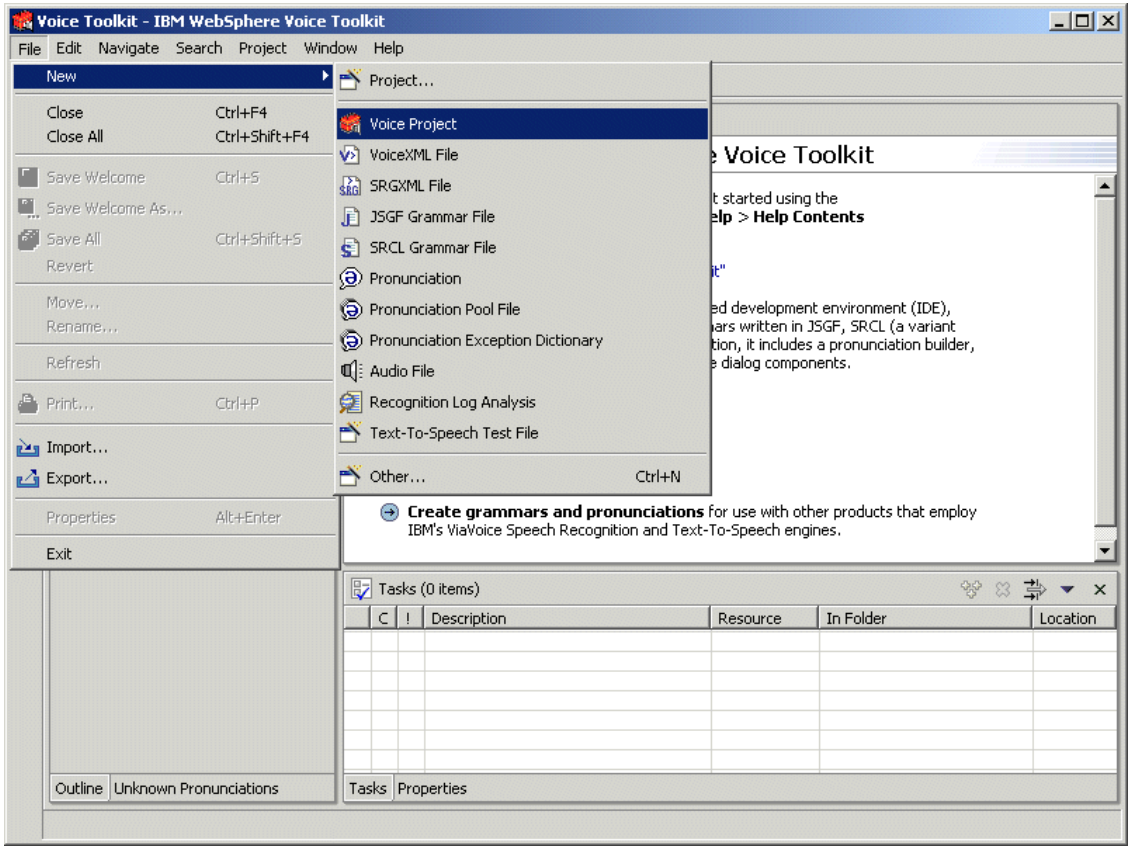


Figure 8-10 Voice Toolkit IDE

2. Enter your project name and specify the directory you wish to create the new project in as shown in Figure 8-11 on page 375. Click **Finish** and continue.

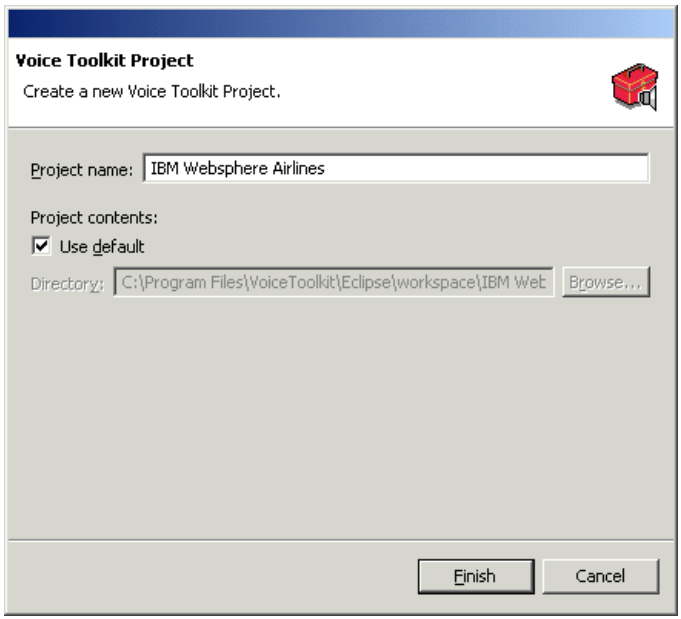


Figure 8-11 Project directory

Note: You may also use the Project wizard to create a new Voice Project. Click **File -> New -> Project ->Voice Toolkit**. Then select the Voice Toolkit wizard to proceed.

Tip: If you create a new project and want to copy files that already exist somewhere else, click **File -> Import** to import those files into the project. If you want to copy files from your project to another location, click **File -> Export**.

Create a VoiceXML File

To create a VoiceXML file, click **File -> New -> VoiceXML File** as shown in Figure 8-12 on page 376.

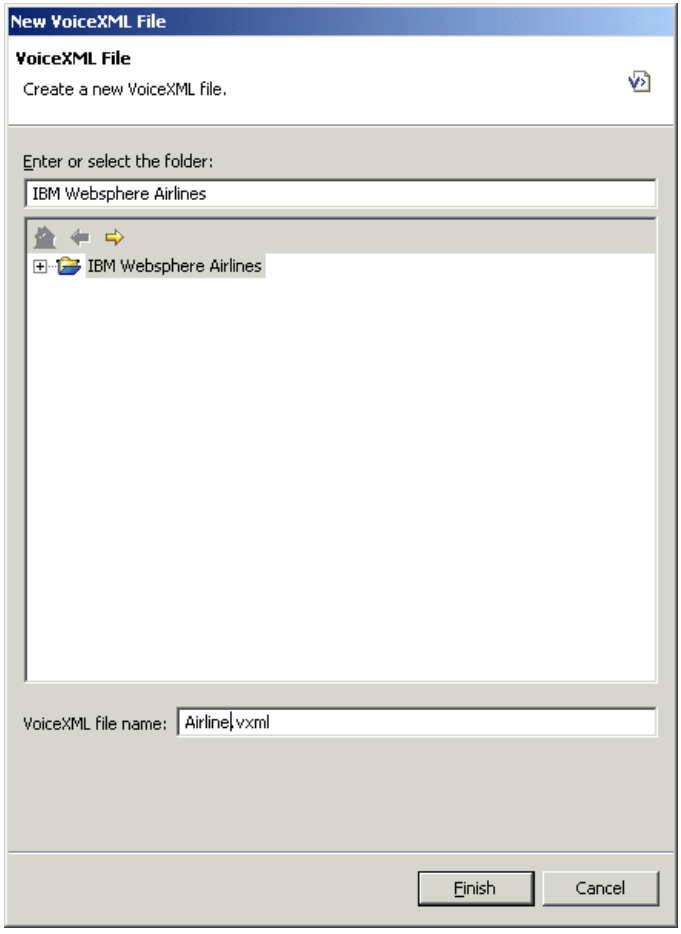


Figure 8-12 Create a new VoiceXML file

The VoiceXML editor provides two modes of building voice server applications. Voice XML may either be entered in directly via a source level editor, or tags and data may be entered via a content assist editor.

The content assist editor is suited to people with fewer programming skills, since it provides a context list of nodes and options available as the application is developed. When using the content assist editor, you should be aware that the add before or add after options refer to the current node. Often users select after when they wish to add a child to the current node, as seen in Figure 8-13 on page 377.

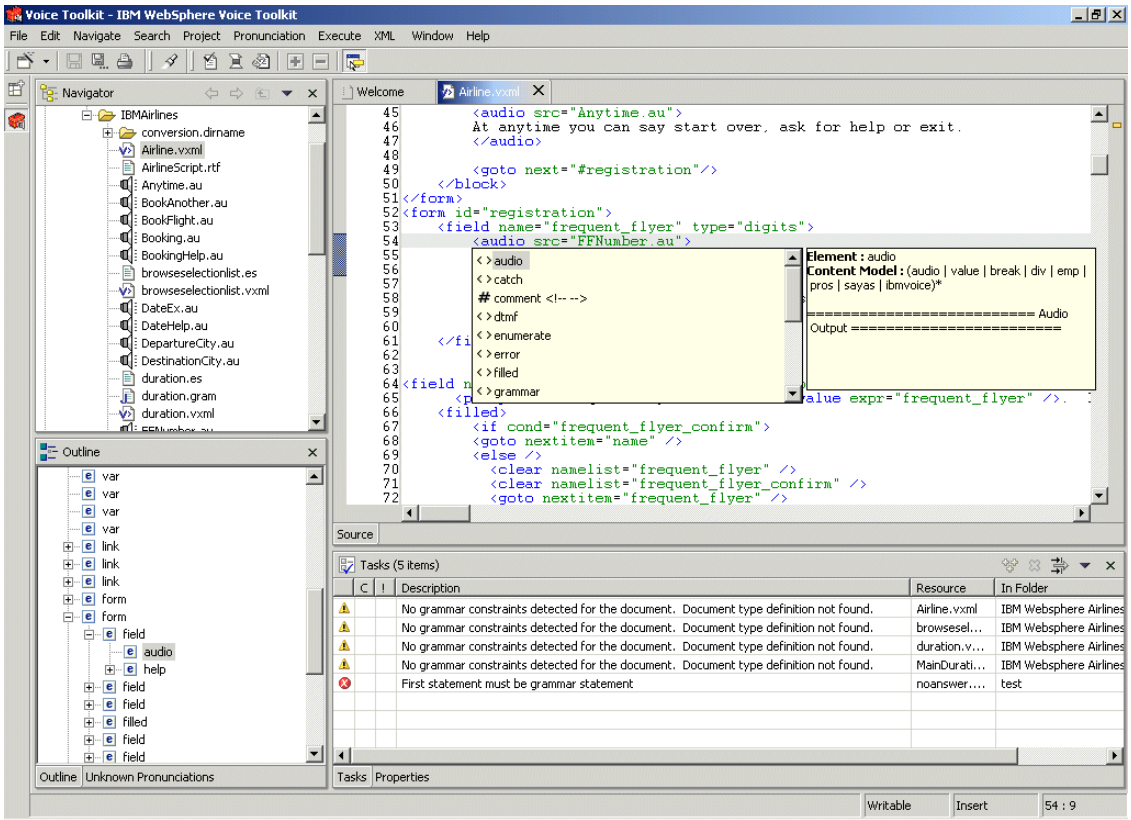


Figure 8-13 Content assist

A Voice XML Document Type Definition (DTD) is provided with the Voice Toolkit. The DTD is a set of rules about the structure, elements, and naming conventions for your VXML document. It allows a programmer to define various pieces of data you wish to model, along with designating the relationships between the data. The DTD in the Voice Toolkit will validate that the contents in your Voice XML file are consistent with your predefined rules.

8.1.5 Developing a VXML application

This section looks at developing applications for WebSphere Voice Server. We will start with a simple application and continue to enhance our application by adding key features of the VXML language as we proceed throughout the chapter. We will follow a basic flow chart to design our application (see Figure 8-14 on page 378).

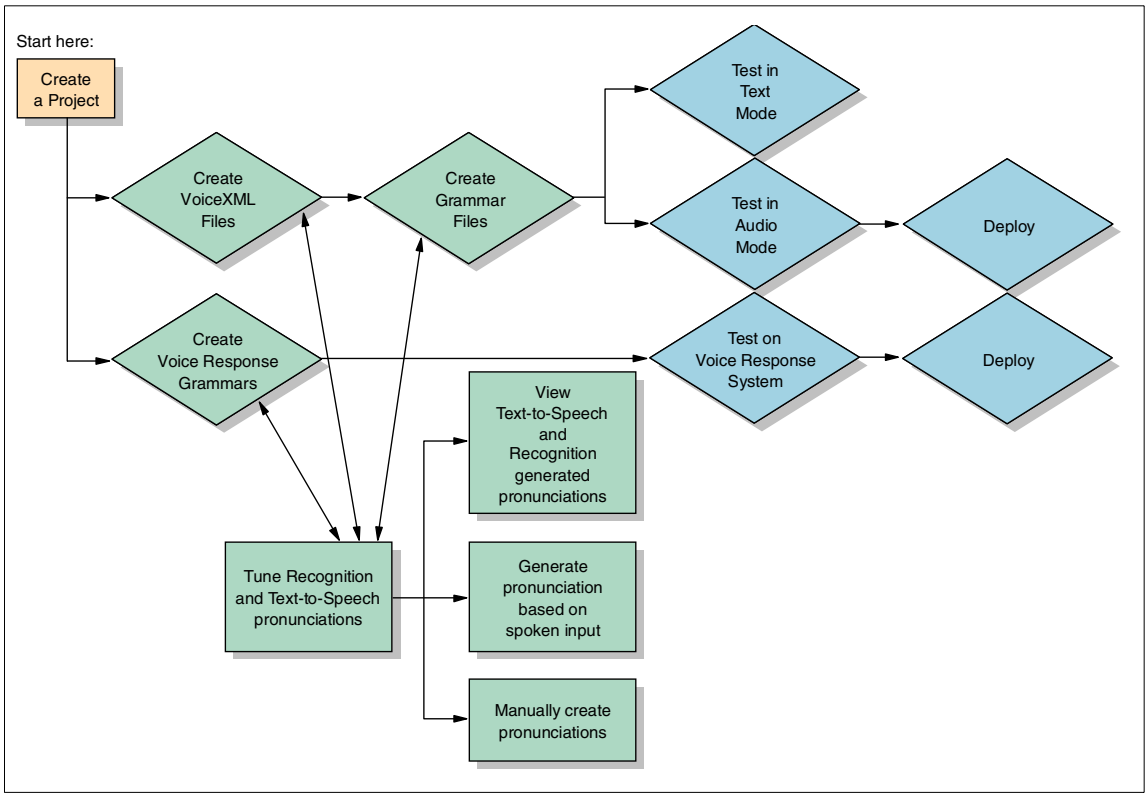


Figure 8-14 Flow of voice application design

For our application, we have chosen to develop an application for IBM WebSphere Airlines. This VXML application will prompt the caller for a frequent flyer number, last name and PIN. After receiving this input from the caller, the system will tell the caller how many frequent flyer points they have and give the caller an option to proceed to Award Bookings. The caller then will have the option of creating an itinerary. If the caller says no, the system will exit. If the caller wishes to proceed in creating an itinerary, the system will prompt the caller for the number of passengers travelling and if one of the passengers is the caller. After obtaining this information, the system will prompt the user for the type of award booking he or she wishes to make, departure city, destination, travel dates, travel times and seat preferences. After input has been processed for these values, the system will read a list of available flights for the specific dates and times and preferences. The caller will select a flight and will be given the option to book the flight.

Dialog structure

VoiceXML documents are composed primarily of top-level elements called dialogs. There are two types of dialogs defined in the language: <form> and <menu>.

Forms and form items

Forms allow the user to provide voice or DTMF input by responding to one or more <field> elements.

Fields: Each field can contain one or more <prompt> elements that guide the user to provide the desired input. You can use the count attribute to vary the prompt text based on the number of times that the prompt has been played.

Fields can also specify a type attribute or a <grammar> or <dtmf> element to define the valid input values for the field, and any <catch> elements necessary to process the events that might occur. Fields may also contain <filled> elements, which specify code to execute when a value is assigned to a field.

You can reset one or more form items using the <clear> element.

Subdialogs: Another type of form item is the <subdialog> element, which creates a separate execution context to gather information and return it to the form. For more information, see 8.1.9, “Adding components” on page 405.

Blocks: If your form requires prompts or computation that do not involve user input (for example, welcome information), you can use the <block> element. This element is also a container for the <submit> element, which specifies the next URI to visit after the user has completed all the fields in the form. You can also jump to another form item in the current form, another dialog in the current document, or another document using the <goto> element.

Types of forms: There are two types of form dialogs:

- *Machine-directed forms* — traditional forms where each field or other form item is executed once and in a sequential order, as directed by the system.
- *Mixed-initiative forms* — more robust forms in which the system or the user can direct the dialog. When coding mixed-initiative forms, you can use form-level grammars (<form scope=“dialog”>) to allow the user to fill in multiple fields from a single utterance, or document-level grammars (<form scope=“document”>) to allow the form’s grammars to be active in any dialog in the same VoiceXML document; if the document is the application root document, then the form’s grammars are active in any dialog in any document within the application. You can use the <initial> element to prompt for form-wide information in a mixed-initiative dialog, before the user is prompted on a field-by-field basis.

In our application, we will use the directed dialog method.

Let's create our first form called "Welcome". This form will be used for the introductory announcement for IBM WebSphere Airlines.

```
<VXML verion="1.0">
<form id="welcome">
<block name="Hello">
<prompt>
Welcome to IBM WebSphere Airlines Frequent Flyer Award Bookings.
</prompt>
<prompt>
At anytime you can say start over, ask for help or exit.
</prompt>
<goto next="#registration"/>
</block>
</form>
```

We have used the <goto> element to lead the application into a form called "registration". This form will ask the caller for a Frequent Flyer Number.

```
<form id="registration">
<field name="frequent_flyer" type="digits">
<prompt> What is your IBM WebSphere Airlines frequent flyer number? </prompt>
<help>You can say any sequence of digits.</help>
</field>
```

In order to process the Frequent Flyer Number, a new field with an attribute type of boolean will be added to the registration form. A boolean attribute allows us to verify if the number provided was correct. We will add the <filled> tag to accept the caller's response. If he/she says "yes" we will use the <goto> to proceed to the next form and if the response is "no", we will repeat the form registration.

```
<field name="frequent_flyer_confirm" type="boolean">
<prompt>You frequent flyer nunmber is <value expr="frequent_flyer" />.  Is this
correct?</prompt>
<filled>
<if cond="frequent_flyer_confirm">
<goto nextitem="name" />
<else />
<clear namelist="frequent_flyer" />
<clear namelist="frequent_flyer_confirm" />
<goto nextitem="frequent_flyer" />
</if>
<clear/>
</filled>
</field>
```

In this next form, digits will be the main focus. We will use the digits attribute to accept a DTMF input of the caller's PIN number. The DTMF input will allow a secure passage of the caller's PIN number to the application. The result of a digits attribute is stored as a string and rendered as digits, that is "two-three-four" not "two hundred and twenty four". The <filled> action will test the field to see if it has five digits. If not, the user hears the message, and is directed to input the PIN again.

Example 8-1 Digits attribute example

```
<field name="PIN"  type="digits" dtmf="true">
<prompt >What is your PIN number? </prompt>
<help>You can say any sequence of digits.</help>
<filled>
<if cond="PIN.length !=5">
<prompt> Sorry, You must have 5 digits for a Pin number </prompt>
<assign name="PIN" expr="undefined"/>
</if>
</filled>
</field>
```

Other built-in data types include date, boolean, currency, number, phone and time. The currency supports ISO currency names, and defaults the system currency.

We have constructed the following forms in the same format as above:

- Itinerary
- Passengers
- Booking
- TravelClass
- Flights
- SDOW
- Book_another
- Good_bye

You can view the working Airline.VXML code in Appendix A, "VXML application" on page 505.

Menus

A menu is essentially a simplified form with a single field. Menus present the user with a list of choices, and associate with each choice a URI identifying a VoiceXML page or element to visit if the user selects that choice. The grammar for a menu is constructed dynamically from the menu entries, which you specify using the <choice> element or the shortcut <enumerate/> construction; you can use the <menu> element's <scope> attribute to control the scope of the grammar.

Note: The <enumerate> element instructs the VoiceXML browser to speak the text of each menu <choice> element when presenting the list of available selections to the user. If you want more control over the exact wording of the prompts (such as the ability to add words between menu items or to hide active entries in your menu), simply leave off the <enumerate> tag.

Menus can accept voice and/or DTMF input; you can specify the acceptable type(s) of input using the construct <property name=“inputmodes” value=“ mode”>, where mode is “dtmf”, “voice”, or “dtmf voice” (the default). If desired, you can implicitly assign DTMF key sequences to menu choices based on their position in the list of choices by using the construct <menu dtmf=“true”>. The following code is an example of a menu that accepts voice and/or DTMF input.

```
<menu id=”IBM_Welcome” dtmf=”true”>
<prompt> Welcome to IBM WebSphere Airlines Frequent Flyer Award Bookings. For
reservations press one or say Reservations, for balance of frequent flyer
points press 2 or say Points.
</prompt>
<choice next=”#option1”/>
<choice next=”#option2”/>
</menu>
```

Our application will consist of a series of forms rather than a menu selection format.

Dynamic content

There are several methods of creating dynamic content for Voice Server applications. Voice XML may itself create dynamic responses via ECMA Script or call-out via Java modules. The Voice XML files themselves may be dynamically generated from other systems, such as WebSphere Application Server.

Adding functions and content via direct Java calls

A VoiceXML implementation platform may have platform-specific functionality that an application wants to use, such as speaker verification, native components, additional telephony functionality, and so on. Such platform-specific objects are accessed using the <object> element, which is analogous to the HTML <object> element. For example, a native credit card collection object could be accessed like this for our “Booking” form:

```
<object name="debit"
classid="method://credit_card/gather_and_debit"
data="http://www.recordings.example/prompts/credit/jesse.jar"/>
<param name="amount" expr="document.amt"/>
<param name="vendor" expr="vendor_num"/>
</object>
```

In this example, the <param> element is used to pass parameters to the object when it is invoked. When this <object> is executed, it returns an ECMAScript object as the value of its form item variable. This <block> presents the values returned from the credit card object:

```
<block>
<prompt>The card type is <value expr="debit.card"/>. </prompt>
<prompt>The card number is <value expr="debit.card_no"/>. </prompt>
<prompt>The expiration date is <value expr="debit.expiry_date"/>. </prompt>
<prompt>The approval code is <value expr="debit.approval_code"/>. </prompt>
<prompt>The confirmation number is <value expr="debit.conf_no"/>. </prompt>
</block>
```

Dynamic content from WebSphere Application Server

For systems requiring complex dynamic data, application servers such as WebSphere Application Server may be used to produce entire VoiceXML scripts directly from the server applications. The application server should track user sessions and other stateful information.

Using other facilities

The following are some useful facilities that may be used when developing a system or application.

Accessing session variables

Depending on the deployment platform, some standard session variables may be defined. These variables may be used within VoiceXML scripts to extend the script functionality. It is important to understand that DNIS and ANI can only be read when they are passed from the PBX. The following example shows a script to read out the ANI and DNIS values of the current call:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<VXML version="1.0">
 <form id="info">
    <block>
    <prompt>The ANI value is <value expr="session.telephone.ani"/></prompt>
    <prompt>The DNIS value is <value
expr="session.telephone.dnis"/></prompt>
    </block>
  </form>
</VXML>
```

Transferring calls

The following is an example of transferring a call to another number:

```
<?xml version="1.0"?>
<VXML version="1.0">
  <form id="transfer">
```

```
    <var name="mydur" expr="0"/>
    <block>Please wait while you call is transfered</block>

    <transfer name="mycall" dest="phone://1234" connecttimeout="30s"
bridge="true">
    <filled>
    <assign name="mydur" expr="mycall$.duration"/>
    <if cond="mycall == 'busy'">
    <prompt>Sorry, the phone is busy..</prompt>
    <elseif cond="mycall == 'noanswer'"/>
    <prompt>Sorry, there is no answer.</prompt>
    </if>
    </filled>
    </transfer>
  </form>
</VXML>
```

This code will transfer a call to number 1234 and wait for the call to complete. Change the bridge parameter to false to do a blind transfer. The transferring feature does not work on all platforms. The Dialogic platform does not support the transferring feature. The Cisco VoIP platform supports the tromboning feature of transferring a call, whereas the WebSphere Voice Response for AIX or Windows for WebSphere Voice Server supports the transferring feature fully.

Non-VoiceXML applications for WebSphere Voice Server

VoiceXML is the primary application interface for WebSphere Voice Server. The WebSphere Voice Server Speech Technologies group provides an OEM product that allows direct access to the TTS engine via a C callable interface. Please contact IBM directly for more information on this product.

8.1.6 Grammars

In VXML, a grammar is basically a “blueprint” used to recognize a group of words that are valid for speech input. The two types of grammar are external and inline. External grammar is a set of words that are retrieved from an external grammar file. Inline grammar is a group of words imbedded directly in the VXML file.

We will take a look now at how the grammar code varies in each of the following grammar formats.

The inline format is grammar that is used directly from your VXML file. Example 8-2 is an example of an inline grammar in a form identified by “name” for our application.

Example 8-2 Inline grammar

```
<field name="name" >
   <prompt>
```

```
   What is your last name?
   </prompt>
   <grammar>
   Dadhich | Dietrich | Kempny | Hu |  Jiong | Poggioli | Credle |
   </grammar>
</field>
<filled>
<assign name="User" expr="name" />
</filled>

<field name="name_confirm" type="boolean">
<prompt>You stated your name is<value expr="name" />.  Is this
correct?</prompt>
<filled>
<if cond="name_confirm">
<goto nextitem="PIN" />
<else />
<clear namelist="name" />
<clear namelist="name_confirm" />
<goto nextitem="name" />
</if>
<clear/>
</filled>
</field>
```

The inline grammar form name prompts the user for his or her last name. After receiving input, the TTS engine will repeat the name spoken using the <filled> tag and ask if the name is correct. If the user says “yes” the application will go to the form PIN, which was discussed in Example 8-1 on page 381.

On the other hand, external grammar code is quite different. The Grammar Editor is the tool designed to generate a grammar file. The Voice Toolkit simplifies the development of command and control grammars and aids in testing these grammars to enhance robustness and performance. The following are three types of file extensions you can use to develop your external grammars:

1. Java Speech Grammar Format (JSGF)
   - A platform-independent text representation of grammars used in speech recognition. Grammars are used by speech recognizers to determine what the recognizer should listen for and to describe the utterances a user says. It adopts the style and conventions of Java as well as the traditional grammar notations.
2. Augmented Backus Naur Form (ABNF)[3]
   - This is a plain-text (non-XML) representation that is similar to traditional BNF grammar and to many existing BNF-like representations commonly used in the field of speech recognition, including the Java Speech

Grammar Format (JSGF) from which this specification is derived. Augmented BNF should not be confused with Extended BNF, which is used in DTDs for XML and SGML.

3. SRGF (Speech Recognition Grammar Form) XML Form[4]
   – This syntax uses XML elements to represent the grammar constructs and adapts designs from the PipeBeach grammar, TalkML [TALKML] and a research XML variant of the Java Speech Grammar Format (JSGF).

The Voice Toolkit simplifies the development of command and control grammars and aids in testing these grammars to enhance robustness and performance.

Creating a grammar file

This Voice Toolkit can be used to develop external grammars. (You can also use the built-in grammar included in the IBM WebSphere Voice Server SDK's VoiceXML browser or an inline grammar specified within the VoiceXML file.) You must create a Voice Project before starting the grammar.

Creating an external JSGF grammar file

1. Select **File -> New -> JSGF Grammar File**, or right-click in the Navigator to display the contextual menu, and select **New -> JSGF Grammar File**.
2. In the New File wizard, select the desired Voice Project folder.
3. Type a name for your grammar file. The default extension for a JSGF grammar file is .jsgf. Other valid extensions include .jsg, .gram, and .gra. (If desired, you can rename the file later in the grammar editor.)
4. Click the **Finish** button, and the grammar file is added to the Project directory in the Navigator.
5. The grammar editor launches and opens a default grammar file. Type the grammar source code in the editor panel.

Tip: You can also launch the grammar editor by right-clicking the reference to the external grammar file name in your VoiceXML file (<grammar src = "x.jsg">). From the contextual menu, select **Edit Grammar File**.

Creating a SRCL grammar file

Speech Recognition Control Language (SRCL) is a variant of Backus-Naur Form (BNF) grammars.

1. Select **File -> New -> SRCL Grammar File**, or right-click in the Navigator to display the contextual menu, and select **New -> SRCL Grammar File**.

[3] W3C Speech Recognition Grammar Specification Version 1.0
[4] W3C Speech Recognition Grammar Specification Version 1.0

2. In the New File wizard, select the desired Voice Project folder.
3. Type a name for your file. The default extension for a SRCL file is .bnf. (If desired, you can rename the file later in the grammar editor.)
4. Click the **Finish** button, and the grammar file is added to the Project directory in the Navigator.
5. The grammar editor launches and opens a blank or default grammar file. Type the grammar source code in the editor panel.

Creating an SRG XML grammar file

Speech Recognition Grammar Format (SRG) XML is a markup for grammars for use in speech recognition.

1. Select **File -> New -> SRGXML File**, or right-click in the Navigator to display the contextual menu, and select **New -> SRGXML File**.
2. In the New File wizard, select the desired Voice Project folder.
3. Type a name for your file. The default extension for a SRG XML file is .grxml. (If desired, you can rename the file later in the grammar editor.)
4. Click the **Finish** button, and the grammar file is added to the Project directory in the Navigator.
5. The grammar editor launches and opens a basic grammar file. Type the grammar source code in the editor panel. Use content assist to open a list of valid tags for the element or attribute at the cursor location.

Grammar syntax

The grammar syntax consists of basically two items, a grammar header and a grammar body. The grammar header declares the version of .JSGF, .GRAM or .GRXML and the grammar name, and (optionally) any imported grammars or rules. The form of the grammar name can be either a “simple grammar name” (that is, grammarName) or a “full grammar name” (that is, packageName.simpleGrammarName). To import all public rules from another grammar, specify fullGrammarName.*. To import specific public rules from another grammar, use a “fully qualified rule name” (that is, fullGrammarName.rulename). The grammar body consists of one or more rules that define the valid set of utterances. We have chosen to describe the noanswer grammar file for our application.

Example 8-3 shows what the grammar will look like in the JSGF format.

Example 8-3 JSGF grammar

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

grammar noanswer;
```

```
public <noanswer>  =  skip {this.$value = "undefined";}
| [I] (don't | do not) (have | know) [one] {this.$value = "undefined";} ;
```

For even more complex grammar descriptions, an ECMA script may be embedded with the grammar files. ECMA is a variation of the JavaScript language and allows flow control and variables to be used within the grammar description.

Speech Recognition Control Language (SRCL) grammar

SRCL and BNF grammar files may be used for WebSphere Voice Server under DirectTalk applications. SRCL and BNF files are compiled into WebSphere Voice Server for DirectTalk grammar pool files and are accessible as global grammar files, which is discussed in “Grammar pool files” on page 389.

Example 8-4 shows what the grammar will look like in the ABNF format.

Example 8-4 ABNF grammar

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

language en_US;
mode voice;
root$noanswer;

public $noanswer  =  skip {this.$value = "undefined";}
| [I] (don't | do not) (have | know) [one] {this.$value = "undefined";} ;
```

Example 8-5 shows what the grammar will look like in the XML format.

Example 8-5 XML 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" mode="voice" root="noanswer">
   <rule id="noanswer" scope="public">
   <tag> this.$value="undefined" </tag>
    <one-of>
      <item> skip </item>
      <item>
         <item repeat="0-1"> I </item>
         <one-of>
            <item> don't  </item>
            <item> do not </item>
         </one-of>
         <one-of>
            <item> have </item>
```

```
            <item> know </item>
          </one-of>
          <item repeat="0-1"> one </item>
        </item>
    </one-of>
    </rule>
</grammar>
```

Grammar pool files

Grammar pool files are global grammar files used only by the telephony connection WebSphere Voice Response for the WebSphere Voice Server. The Voice Toolkit can produce grammar pool files from other grammar file types when creating a Voice XML project.

You can define pronunciations for speech recognition in either a pool file or an <ibmlexicon> tag. The differences between the two are shown in Table 8-1.

Table 8-1 Pool file vs. <ibmlexicon> tag

Pool file	<ibmlexicon> tag
Pronunciation information is global for all applications that use the same speech recognition engine.	Pronunciation information is global for the duration of the application session.
Pronunciation information is used only for speech recognition, *not* for text-to-speech.	Pronunciation information is used for both speech recognition and text-to-speech.
Pronunciation information can be used in VoiceXML applications, and with other platforms that use the IBM ViaVoice Speech Recognition engine.	Pronunciation information can only be used by VoiceXML applications that will be executed using the IBM WebSphere Voice Server for Windows 2000 or the IBM WebSphere Voice Server for Windows 2000 SDK (Version 3.1).

Creating a pronunciation pool file

Pronunciation pool source files (file extensions .pbs and .wrd) must exist within a Voice Project. When you save the pool source files, the pool files are built.

Choose one of the following methods:

1. With the VoiceXML or grammar editor open, select **Pronunciation -> Generate Pool File**.
2. With the VoiceXML or grammar editor open, right-click in the source editor to open the contextual menu, and select **Generate Pool File**.

3. From the Voice Toolkit menu bar, select **File -> New -> Pronunciation Pool File.**
4. Without opening a Voice Project, select **File -> New -> Pronunciation**. On the Create New Pronunciation window, click the **Pronunciation Pool** file radio button to add or edit pronunciations in a pool file, and then click the **Finish** button. Continue by creating pronunciations. Any words you add will be saved to the Voice Project that you select and source file that you select or create when closing the Pronunciation Builder.
5. In the New Pronunciation Pool File window, select a Voice Project from the list, or type the name in the text box as seen in Figure 8-15.

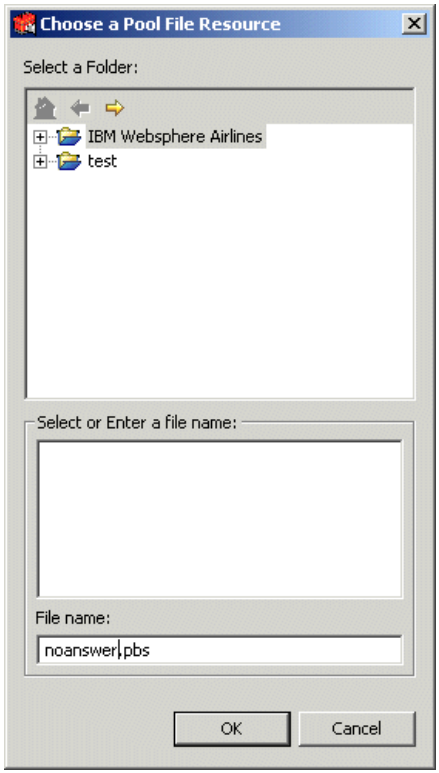


Figure 8-15 Pool file window

6. In the Pool file name text box, select an existing pool source file type the name with the file type .pbs or create a new pool file in the File Name section of the window.
7. Select **OK**. You will see a window saying the pool file has been successfully generated (Figure 8-16 on page 391). This adds the words and pronunciations to a pool source file (.pbs) and a pool word file (.wrd) in the

Voice Project. The .pbs file lists all the words and their default pronunciations. The .wrd file is the word list that includes all the words defined in your pool file (without pronunciations or duplications). Only words listed here are included in the pool.

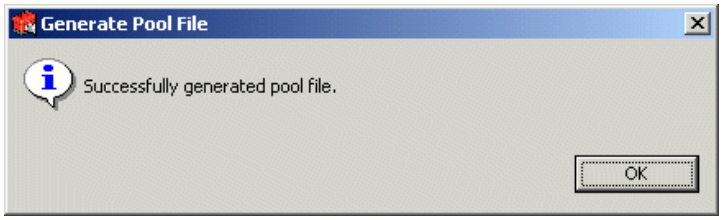


Figure 8-16 Generated pool file message

Grammar test tool

You can test grammar files before you integrate them into your VoiceXML application by using the grammar test tool. You must have the IBM WebSphere Voice Server for Windows 2000 SDK V3.1 installed to test in audio mode.

To use the grammar test tool, do the following:

1. If open, select the grammar set with which you want to start the test. Otherwise, open the desired .jsgf, .gram or .grxml grammar file (only saved versions of files are used in the test).

Note: You can only run the grammar test tool for JSGF and SRGF XML grammar files. The grammar test tool will not support testing grammar on a SRGF ABNF grammar file.

2. Select **Execute -> Test Grammar** to start the grammar test tool. Wait for a moment for the system to load the grammar into the test tool. You'll know this has happened when the name of the grammar appears under Vocab Name in the grammar test tool.
3. Select the desired test view. The default test view is to test in audio mode. Click the microphone icon to turn the microphone on and off. You will be able to test the grammar and see the successful recognition of the grammar, as shown in Figure 8-17 on page 392.

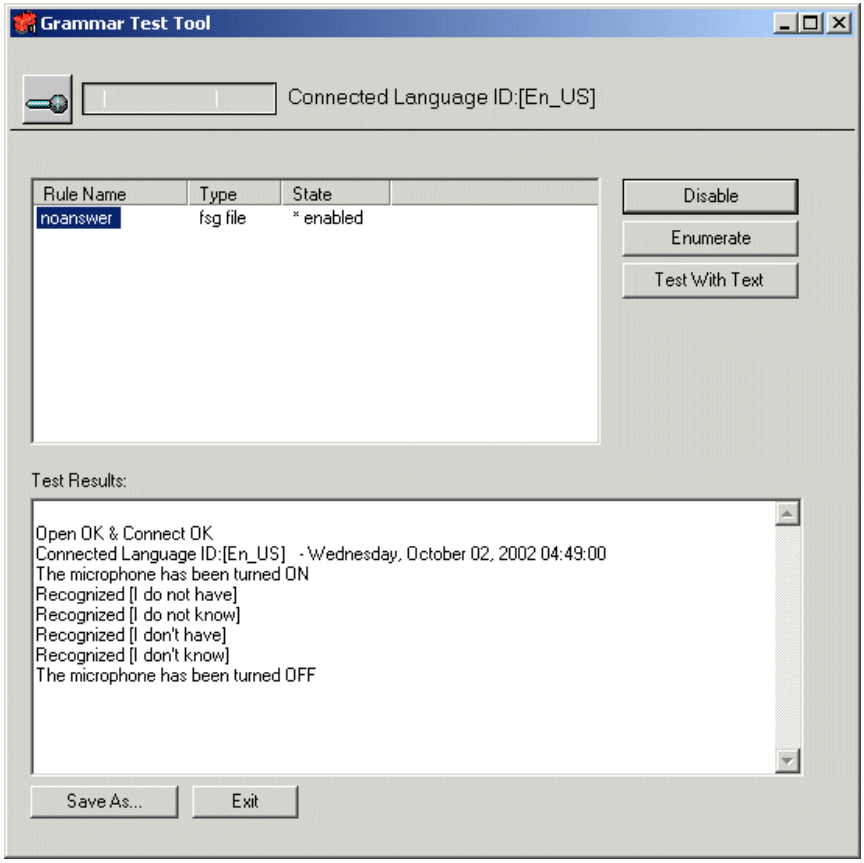


Figure 8-17 Grammar test tool

You can also change test views to enable testing in text mode (either one typed phrase at a time or a file of test phrases) or to enumerate phrases that the grammar can recognize. All the views have a results panel at the bottom of their windows in which the tool posts the results of the various tests.

Note: Each time you change the grammar file, be sure to save it. Before testing the change, close the test tool and restart it to load the current version of the grammar.

8.1.7 Pronunciation Builder

The Voice Toolkit provides an easy-to-use Pronunciation Builder to help create, test, and tune pronunciations of words in your grammar, VoiceXML, or pool files. It uses a keyboard or a microphone to create and enhance how a word will be recorded and heard through the text-to-speech engine.You can also create pronunciations for words in exception dictionaries to be used by the text-to-speech engine.

In order for the speech recognition engine to recognize a word when the user speaks it, the engine must know what pronunciation to expect. The IBM text-to-speech (TTS) engine automatically creates a basic pronunciation based on the spelling. You can revise those pronunciations, create alternative pronunciations, or create pronunciations for "unknown words" (that is, words that are not in the speech recognition engine's vocabulary).

If any word is unknown, a default pronunciation will be generated by the TTS engine, and you can edit the pronunciation by using the Pronunciation Builder.

The <ibmlexicon> VoiceXML extension allows different gender and speaker ages to be used for TTS generation. This allows applications to be tailored to specific users with a minimum of effort. We will have our TTS engine be a female child's voice by designating the following parameters:

Example 8-6 <ibmvoice> example

```
<ibmvoice gender="female" age="child"></ibmvoice>
```

To check/create a pronunciation:

1. Highlight and right-click the desired word in your file.
2. From the contextual menu that appears, select **Pronunciation**.Choose from the following options:
   - Verify Pronunciation

     Cross references the pronunciation with the speech engine and returns either a 0 Unknown Words Found or a 1 Unknown Word Found Warning, as seen in Figure 8-18.

Figure 8-18 Verify pronunciation warning

– Play Pronunciation

  Speaks the highlighted word.

– Compose Pronunciation

  The Pronunciation Builder window appears (Figure 8-19) with the target word filled in and the default pronunciation already generated.

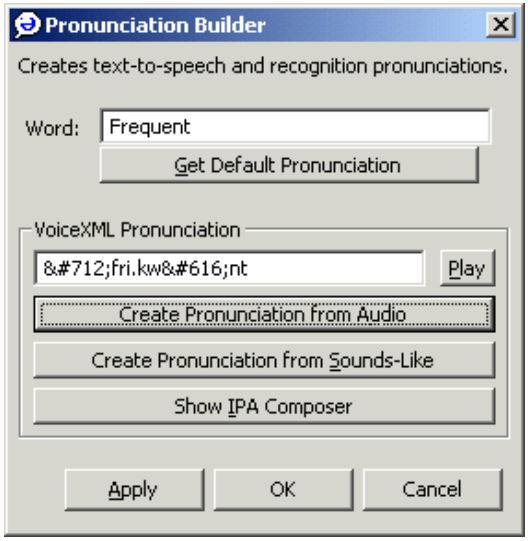


Figure 8-19 Pronunciation Builder

1. Click the **Play** button to hear the default pronunciation.
2. If the default pronunciation is acceptable, you are finished, so just click the **OK** button. If you started your search from a VoiceXML file, the system adds the necessary <ibmlexicon> and <word> tags to the VoiceXML file. If you started your search from a grammar file, follow the on-screen instructions to select the VoiceXML file to which you want to add this pronunciation.
3. If the default pronunciation does not sound right, click one of the following:

- Create Pronunciation from Audio

  An audio utility will open up and prompt you to either record the word using a microphone or by using an existing audio file, as shown in Figure 8-20 on page 395.

Figure 8-20 Record pronunciation

To use the microphone utility, click the **Record** button, say the word “frequent” and click the **Stop** button. The VoiceXML pronunciation utility will create a phonetic pronunciation of the utility that you can play. Once you are satisfied with the pronunciation, click **OK** then click **Apply** in the Pronunciation Builder window and an <ibmlexicon> tag will be inserted into your VXML file, as shown in Example 8-7.

Example 8-7 IBM lexicon for the word “frequent”

```
<ibmlexicon>
<word spelling="Frequent" pronunciation="fr&#616;kw&#601;n&#809;t"/></word>
</ibmlexicon>
```

To use the audio utility, Click the **Record** button, and you will be prompted to select an audio file with the recording of the word, as seen in Figure 8-21 on page 396. Select a saved audio file and Click **OK**. Then click **Apply** in the Pronunciation Builder window and an <ibmlexicon> tag with a pronunciation attribute will be inserted into your VXML file.

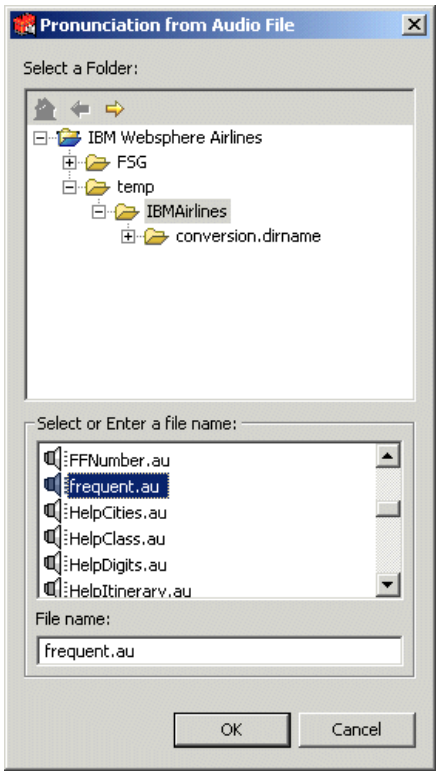


Figure 8-21 Choosing an audio file for your pronunciation

- Create Pronunciation from Sounds-Like.
  - Use the <ibmlexicon> and apply the sounds-like attribute as shown in Example 8-8.

Example 8-8 Sounds-like

```
<ibmlexicon>
<word spelling="Jiong" sounds-like="Jhong"/>
</ibmlexicon>
```

  - Test the sounds-like attribute as shown in Figure 8-22 on page 397.

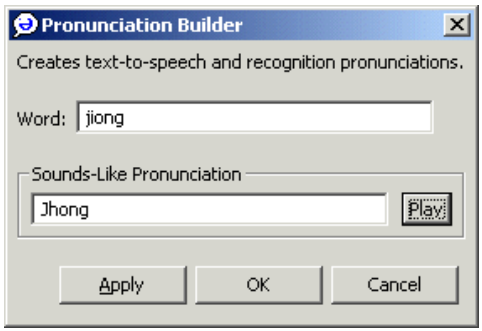


Figure 8-22 Sounds-like pronunciation

Restrictions:

1. The word element is not available on the DirectTalk deployment platform.
2. When creating multiple pronunciations of the same word, the last instance is used by the text-to-speech engine when the word is said. All instances will be used by the recognition engine. This also applies to words in the global dictionary.

- Show IPA Composer
  - Use the IPA Composer (International Phonetic Alphabet system) as shown in Figure 8-23 on page 398 to edit the phonemes from the default pronunciation until the word sounds right. Once the word sounds correct, click the **OK** button. Then click **Apply** in the Pronunciation Builder window and an <ibmlexicon> tag with a pronunciation attribute will be inserted into your VXML file.

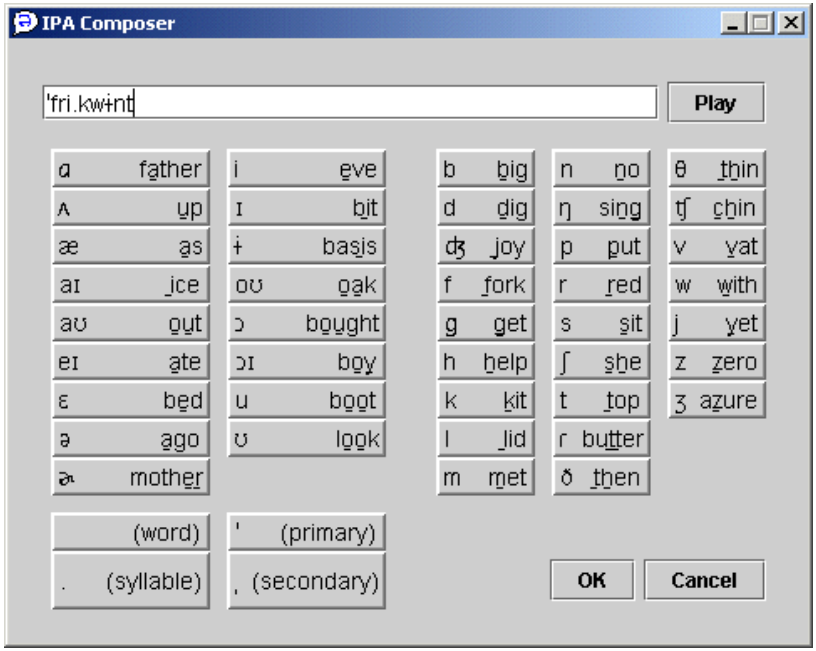


Figure 8-23 IPA Composer

Points to keep in mind:

- Click the **Play** button at any time to check the pronunciation.
- The IPA Composer has four main sections to use when creating a pronunciation: vowels, consonants, word/syllable breaks, and primary/secondary emphasis.
- You cannot have only consonants in a word or syllable - there must always be at least one vowel. If you have not done this, a message appears (see Figure 8-24). Each part of a pronunciation must contain at least one vowel for valid audio playback when you click the **Play** button.

Figure 8-24 IPA Composer error message

Tip: There is a trick that you can try if you experience any problems getting a word to sound right. In the Word field of the Pronunciation Builder, re-type the word into something that you think would sound like the pronunciation you are trying to get, that is, how the word is phonetically spelled. For example, you might type `IEEE` as “eye triple ee”. Click the **Generate Default Pronunciation** button, and play it to see how it sounds. If you do this, you must change the spelling in the Word field back to the original word before you click the **OK** button to accept the word and its new pronunciation.

Note: The <ibmlexicon> tags do not work withWebSphere Voice Server for WebSphere Voice Response on AIX 3.1. It will ignore these tags.

8.1.8 Audio recorder

Synthesized speech (text-to-speech) is useful as a placeholder during application development, or when the data to be spoken is “unbounded” (not known in advance), making it impossible to record.

When deploying your applications, however, you may want to consider using professionally recorded prompts. Users expect commercial systems to use high-quality recorded speech because only recorded speech can guarantee highly natural pronunciation and prosody. The Voice Toolkit includes a basic audio recorder. This basic recorder is not a substitute for a professional digital recorder, but can be useful for recording and testing recorded prompts that are placeholders for professionally recorded audio files.

Adding prerecorded audio

For more natural output, prerecorded audio may be used. The VoiceXML browser plays an audio file when the corresponding URI (<audio src=”file”>) is encountered in a VoiceXML document.

Using prerecorded audio files

Prerecorded audio files must be in one of the following formats:

- An 8 KHz 8-bit mu-law .au file
- An 8 KHz 8-bit mu-law .wav file
- An 8 KHz 8-bit a-law .au file
- An 8 KHz 8-bit a-law .wav file
- An 8 KHz 16-bit linear .au file
- An 8 KHz 16-bit linear .wav file

Note: 16-bit linear files will take twice as much storage and download time. For better performance, 8-bit files are recommended.

Recording spoken user input

You can use the VoiceXML <record> element to capture spoken input. The recording ends either when the user presses any DTMF key, or when the time you specified in the maxtime attribute is exceeded. To allow a spoken command to terminate the recording, you can specify a <grammar> element within the <record> element; users must pause briefly before and after speaking an utterance from this grammar. All other grammars are turned off while recording is active. To record an audio file:

1. Select **File -> New -> Audio File**.
2. Select the desired Voice Project or folder, type a name for the audio file, and then click the **Next** button, as shown in Figure 8-25 on page 401.

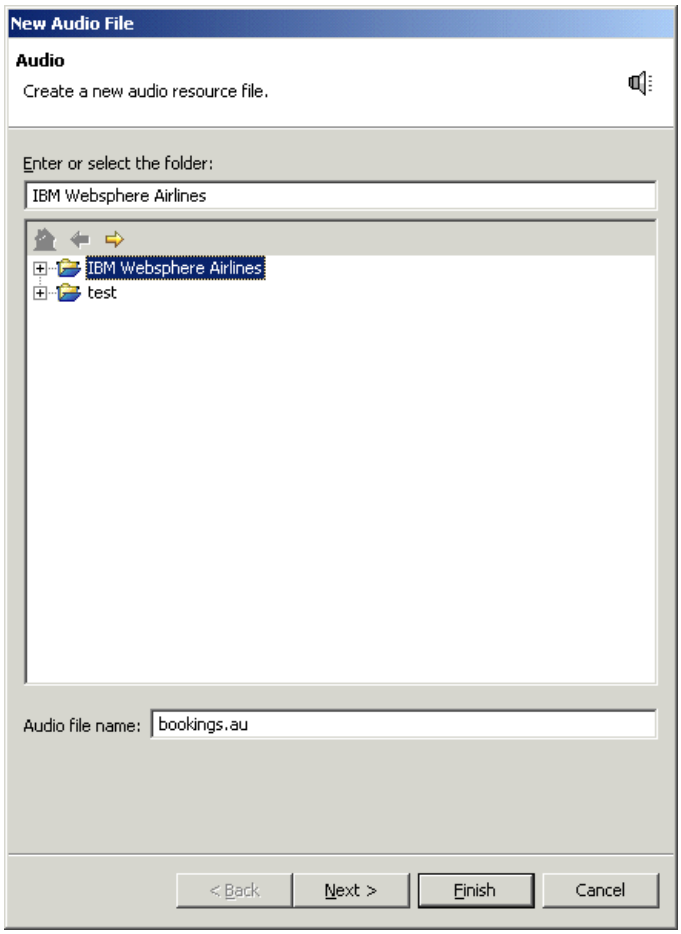


Figure 8-25 New audio file

3. Choose the source you want to generate the audio file from (Figure 8-26 on page 402).

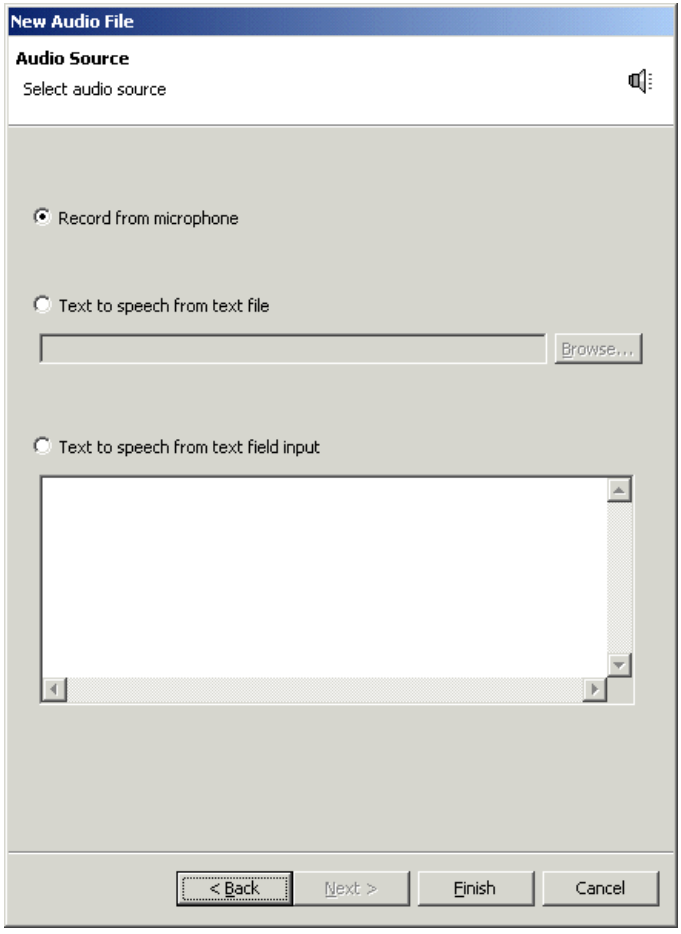


Figure 8-26 Audio Source

The three options are:

a. Record from microphone

- Records audio from spoken words.
- Select the **Finish** button to bring up the recorder (Figure 8-27 on page 403).

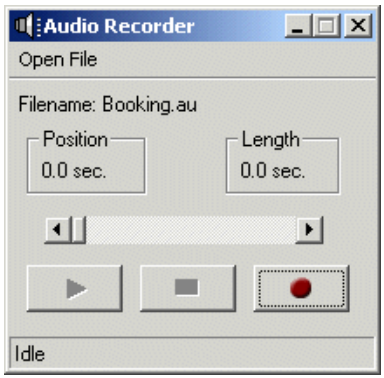


Figure 8-27 Audio Recorder

- To begin recording, press the **Record** button. Use the **Stop** button to end the recording. The tool automatically saves the audio file in your Voice Project.
- To listen to the recording, click the **Play** button.

b. Text to speech from text file

Extracts text from a pre-written document.

c. Text to speech from text field input

Records audio from typed text in the comments box.

Tip: To use the audio file, specify the URI in your VoiceXML file: <audio src="filename"/>. Also, because audio doesn't play when testing in text mode, you can provide a default test in the same command: <audio src="filename”> the text goes here </audio>.

Playing and storing recorded user input

You can play the recorded input back to the user immediately (using a <value> element), or submit it to the server (using <submit next=”URI” method=”post” enctype=”multipart/form-data”/>)to be saved as an audio file.

Add a segment of prerecorded audio in our existing introduction:

```
<block name=”Hello”>Welcome to IBM Airlnes Frequent Flyer Award
Bookings.</block>
```

with an audio file:

```
<audio src=”Welcome.au”></audio>
```

Note: “`Welcome.au`**”** is “youraudiofilename"**.**

The code will now look like this:

```
<block name=”Hello”>
<audio src=”Welcome.au”>
</audio>
</block>
```

When the VoiceXML application is loaded the audio will be played back. The <audio> tag allows alternate text to be used when the specified audio is not available. For example:

```
<block name=”Hello”>
<audio src=”Welcome.au”>
Welcome to IBM Airlnes Frequent Flyer Award Bookings
</audio>
</block>
```

Important: Some platforms allow concatenative speech generation for TTS. This provides more natural speech than the default formant algorithm at the cost of more system memory. There is no setting to switch from one method of TTS to another.

Capabilities and limitations of TTS

TTS prompts are easier to maintain and modify than prerecorded audio prompts. For this reason, TTS is typically used during application development.

TTS is also a powerful tool for use when the data to be spoken is “unbounded” or dynamic (that is, not known in advance) and cannot therefore be prerecorded.

However, TTS cannot yet mimic the complete naturalness of human speech. So, while TTS may be a necessary or convenient option when the input text is dynamic or under development, prerecorded audio is usually a better choice for deployed applications in which the input text is static. WebSphere Voice Server provides a choice of two TTS models, formant and concatenative. Future versions of WebSphere Voice Server may provide other improved TTS models for more realistic speech.

Some users may find the synthesized speech a bit “robotic” when using the formant TTS engine. If you desire a more natural sounding TTS engine, it is suggested that the concatenative TTS engine is additionally purchased with the WebSphere Voice Server.

8.1.9 Adding components

In place of or in addition to writing your own code, you can use the IBM Reusable Dialog Components to add common functions to your VoiceXML file. The Reusable Dialog Components included in the Voice Toolkit are a basic set of subdialogs, templates, and samples that provide VoiceXML source code for common functions, enabling you to quickly and easily add these functions to your applications.

There are four types of Reusable Dialog Components:

- Subdialogs

  These are simple pieces of code that provide basic functions used in typical VoiceXML applications. You can call them from multiple places in the application with only a single instance of code, and you can modify them globally or on a one-time or per-instance basis.

- Templates

  Templates are VoiceXML code that uses the subdialogs to provide a common function, or combines multiple subdialogs to provide a higher level of function. You can use these if the level of function meets your needs, or you can customize them to better suit your needs. The main purpose of a template is to provide an example of how you would create this type of interaction.

- Grammars

  Java Speech Grammar Format (JSGF) grammars are used with the subdialogs.

- Samples

  Samples are combinations of subdialogs, templates and other VoiceXML code that perform a complete function and demonstrate the use of the Reusable Dialog Components in real-life situations.

Reusable Dialog Components

Use the Reusable Dialog Components wizard to import and customize reusable dialogs. Let's add the browseselectionlist subdialog to a form identified as "flights". To start the wizard:

1. Make sure you are inside a <form> tag because this is the only place where you can add a dialog component.
2. Click **Edit -> Add Dialog Component**.
3. Follow the steps in the wizard to add the desired component by first selecting the component you wish to use (Figure 8-28 on page 406).

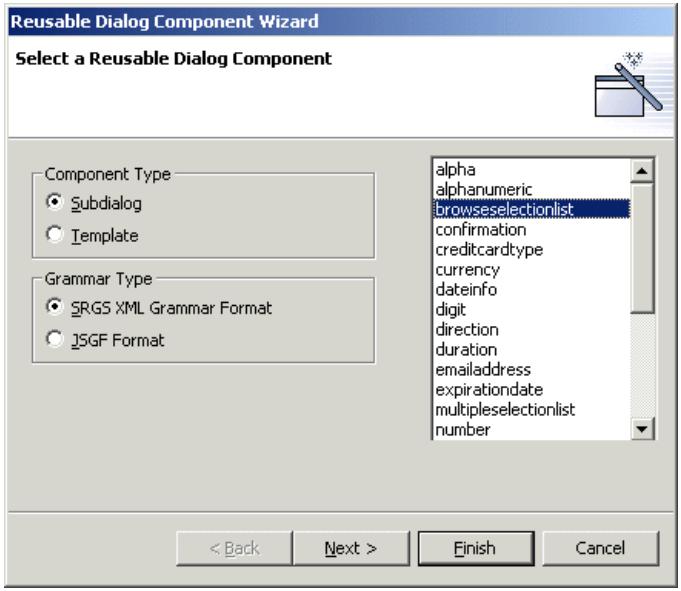


Figure 8-28 Selecting a reusable component

4. Select any optional parameters for the component as shown in Figure 8-29 and click **Finish**.

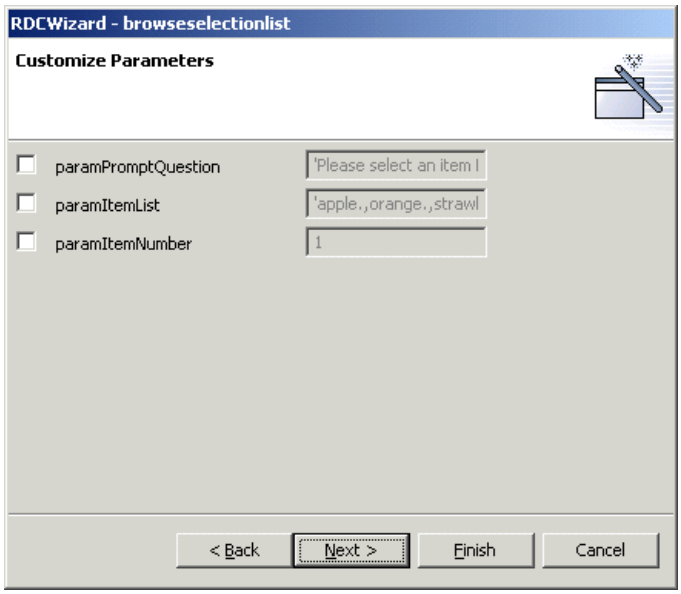


Figure 8-29 Setting parameters

5. If the component supports ECMA scripts, then click **Next** from Figure 8-30 and additional wizard windows will be displayed to designate the ECMA script name and set ECMA script parameters, as shown in Figure 8-31 on page 408.

Note: ECMAScript is a programming language adopted by the European Computer Manufacturer's Association as a standard for performing computations in Web applications. ECMAScript is the official client-side scripting language of VoiceXML. ECMAScript is a limited program model for simple data manipulation.

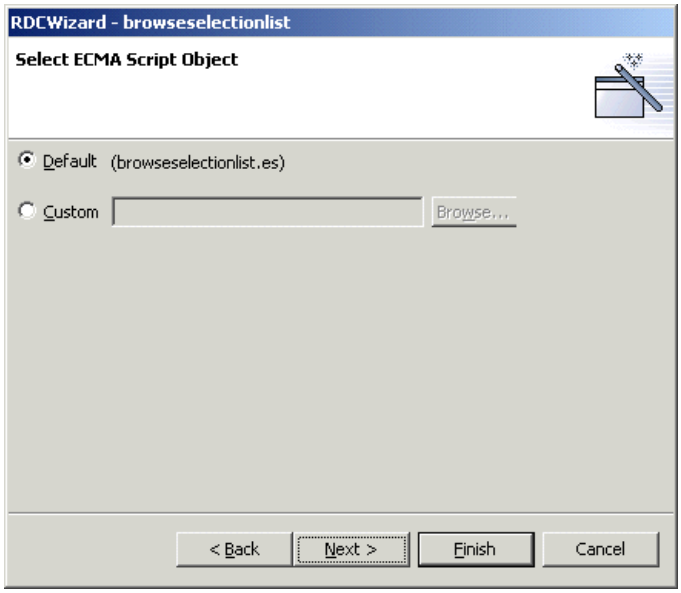


Figure 8-30 ECMA script name selection

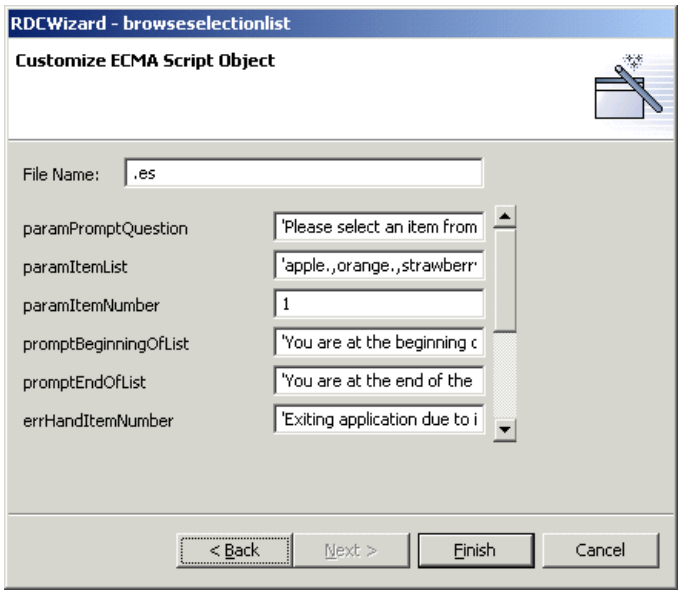


Figure 8-31 ECMA set script parameters

After adding the browseselectionlist subdialog to the “flights” form, the code that is generated in our VXML document will look like the following:

```
<script src="browseselectionlist.es">
var objBrowselelist = new BrowseSelectionList();
</script>

<form id="flights">
<subdialog name="browselelist" src="browseselectionlist.VXML">
<param name="paramSubdialogObj" expr="objBrowselelist"/>
<param name="paramPromptQuestion" expr="'Here are the flights with available '
+ TravelClass + ' class seats, departing from ' + DepartureCity + ' to ' +
DestinationCity + ' on ' + StringDate + '. Please select a flight from the
following '+
'list, using the commands Select, Next, and Previous.'"/>

<param name="paramItemList" expr="Flight1 + '.,' + Flight2  + '.,' +  Flight3
+ '.,'  + '.,' +   Flight4  + '.,' +  Flight5"/>

<filled>
<assign name="SelectedFlight" expr="browselelist.returnSelectedItem"/>
<prompt>
You have selected <value expr="browselelist.returnSelectedItem"/>
</prompt>
<goto next="#confirm" />
</filled>
```

```
</subdialog>
</form>
```

Restriction: If there are parameters duplicated in both the first parameter and ECMA dialogs, the first parameters will take precedence.

8.1.10 Creating and customizing components

The most-used Reusable Dialog Components are subdialogs. Templates are Voice XML code that combines two or more subdialogs. Each supplied subdialog is contained in its own folder. Currently the subdialogs are available in US English only, but they are enabled for National Language Support (NLS). The supplied Reusable Dialog Components do not use any tags that are specific to the IBM VoiceXML browser and should therefore be portable to other Voice XML browsers.

The architecture of Reusable Dialog Components is shown in Figure 8-32.

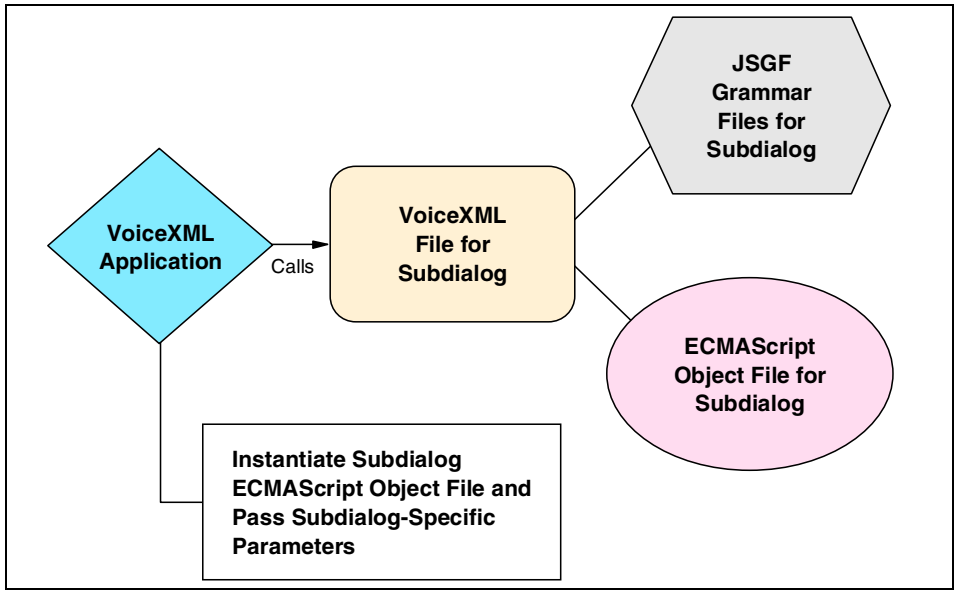


Figure 8-32 Reusable Dialog Components architecture

Each subdialog includes:

- VoiceXML file
- JSGF grammar files
- ECMAScript Object file

The main file for the subdialog is the VoiceXML file. This file contains the programming code plus a detailed comments section that outlines the subdialog's parameters and return values and sample calling code.

Adding your own subdialog

In order for your subdialog to be recognized by the Reusable Dialog Component wizard, the following are required:

- Create your subdialog subdirectory at the <install-directory>\reusable_comp\subdialogs level. This allows you to see the name of your subdialog when using the wizard.
- Create a locale directory (for example, en_us), which is where the associated subdialog files will reside. For examples of other locale names, see the confirmation subdialog, which is provided in various languages. If the locale directory is not found, then you will receive the following error message while using the wizard:

  ```
  A Reusable Dialog Component directory and/or its files were not found.
  ```

- An ECMAScript Object file and VoiceXML file must exist in the locale directory and must use the extensions *.es and *.VXML, respectively. They are case-sensitive and must exactly match the subdialog directory name. For example, if your subdialog directory name is mysubdialog, then your ECMAScript Object file should be named mysubdialog.es, and your VoiceXML file should be named mysubdialog.VXML. These two files are necessary in order to have a Reusable Dialog Component. If neither of these files exists, you will receive the following error message while using the wizard:

  ```
  A Reusable Dialog Component directory and/or its files were not found.
  ```

- The ECMAScript Object file must contain only one function. It must contain at least one default value, and the default values must use the proper syntax. In general, your ECMAScript Object file should follow the proper ECMAScript syntax. If syntax errors are found, you will receive the following error message using the wizard:

  ```
  An error occurred parsing the ES file: <file>.
  ```

Note: The Reusable Dialog Component wizard is *not* an ECMAScript parser. Its parsing capabilities are specific to the guidelines of creating a subdialog put forth by this document. It is the developer's responsibility to make sure your ECMAScript code conforms fully with proper ECMAScript syntax.

If it is a subdialog, then its VoiceXML file must contain a parameter named *paramSubdialogObj* and return values specified by the <return> tag. If other parameters are needed, they must use the prefix `param`; otherwise they will not show up as parameters in the wizard. The parameter names are case-sensitive and must exactly match those used in the ECMAScript Object file when assigning default values; otherwise your default values will not show up in the wizard. If the VoiceXML file does not contain the parameter `paramSubdialogObj` or does not have any defined return values, you will receive the following error message while using the wizard:

```
An error occurred parsing the VoiceXML file: <file name>.
```

If you need more information, consult the *Reusable Dialog Components for VoiceXML Applications* document in the Related Documents section of the online help.

The latest Reusable Dialog Components are available from:

http://www-3.ibm.com/pvc/products/voice/reusable_dialog.shtml

8.2 VoiceXML Debugger

The VoiceXML debugger is a tool that can help you track the behavior and state of VoiceXML programs and easily pinpoint logic errors in VoiceXML applications. Using the debugger, you can pause the running application, examine the working code, and locate the source of the bug, as well as bugs you might have previously missed.

The VoiceXML debugger helps to pinpoint logic errors by offering application developers the ability to perform the following functions:

- Trace/highlight code as it is being executed
- Display logs from the IBM WebSphere Voice Server SDK 3.1
- Inspect and set variables
- Step through VoiceXML applications
- Set conditional breakpoints
- Evaluate ECMAScript expressions during execution
- Simulate key-press (DTMF) input during execution
- Display the list of all documents visited during a session

To use the VoiceXML debugger, you must install the IBM WebSphere Voice Server SDK 3.1.

Opening the debugger

You can open the debugger in audio mode or text mode if you install the IBM WebSphere Voice Server SDK 3.1 on your computer. It is a good idea to debug your code in text mode before debugging in audio mode because you don't need to worry about the potential for recognition errors.

Note: You cannot debug your code in text mode if your application uses precompiled grammars.

1. To open the VoiceXML debugger in the IBM WebSphere Voice Toolkit, use one of the following methods:

- In the Navigator view, select a VoiceXML file and right-click to open the contextual menu.
- Open a file in the VoiceXML editor and right-click to open the contextual menu.
- Select **Execute** from the Voice Toolkit menu bar (while the VoiceXML editor is active).

2. Select one of the following options:

- Use the **Debug in Audio Mode** option if you want to interact with your applications in the VoiceXML debugger using voice recognition and audio output. This simulates how your application will be used when deployed in a production environment shown in Figure 8-33 on page 413.

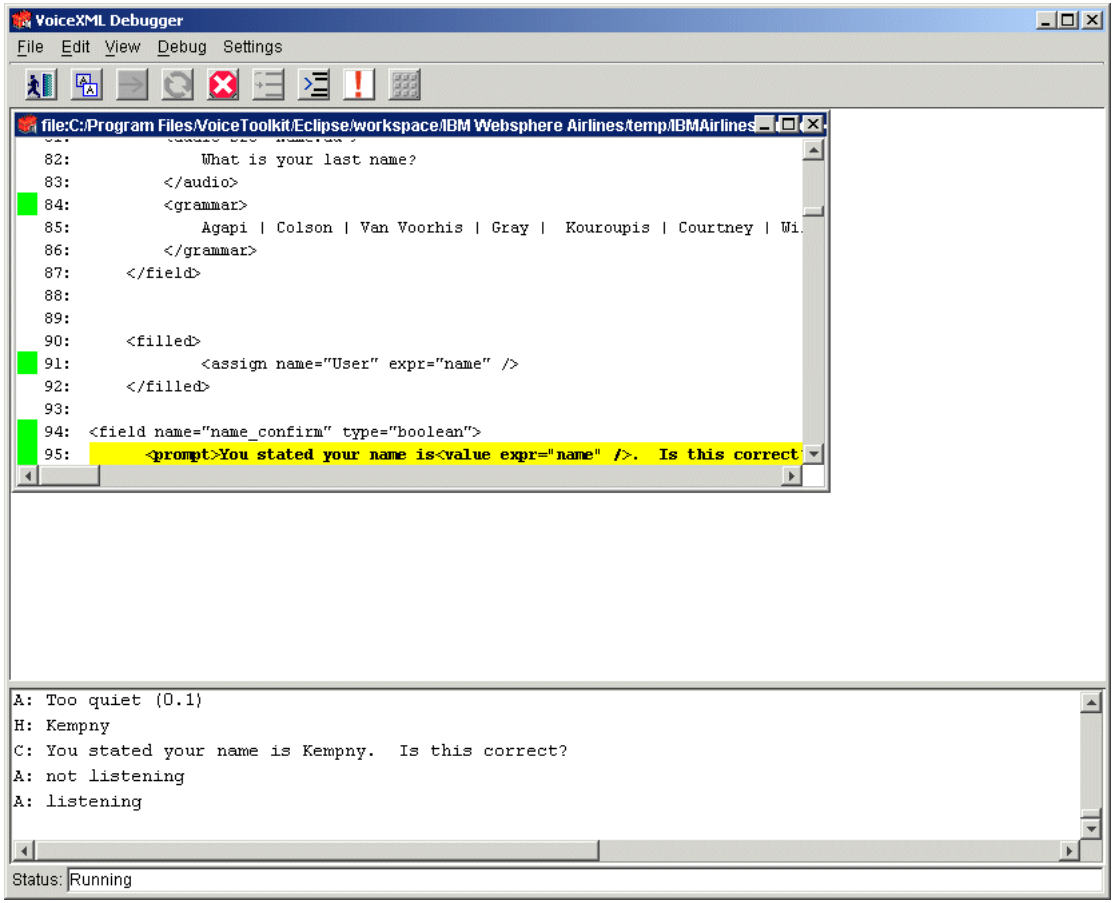


Figure 8-33 Debug audio mode

- Use the **Debug in Text Mode** option if you want to interact with your application entirely through a text interface. In this mode, no voice recognition or audio output of any kind will occur. Instead, a keyboard interface (Figure 8-34 on page 414) will prompt for the input and a console window opens displaying the output of your application, which is similar to Figure 8-33.

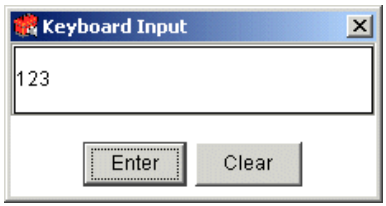


Figure 8-34 Text debug mode

Debugging a VoiceXML program

To debug your program, you can set breakpoints in the source code. A breakpoint is a condition that causes the application to pause temporarily during a debugging session to let you inspect the application's variables.

The VoiceXML debugger is a view-only program. You can edit variables, but to change source code, you must return to the Voice Toolkit environment. Review the working Airline.VXML code in Appendix A, “VXML application” on page 505 and run the debugger. To run the debugger, refer to “Using the Debugger window” on page 414.

Using the Debugger window

The debugger window, shown in Figure 8-33 on page 413, is divided into three sections: the upper area for source code and debugging utility windows, the lower area for the VoiceXML browser log, and a status line at the bottom of the window. It is important to pay close attention to the browser log to understand the current state of the program.

The Debugger toolbar

The toolbar helps you to access functionality quickly.

- **Exit Debugger button**

  Ends the current session and closes the debugger.

- **Copy button**

  Copies the highlighted text in the current source code window to the system clipboard (Ctrl+C).

- **Go/Resume button**

  Starts execution of a VoiceXML application session, or resumes following a breakpoint.

- Restart button

  Restarts the current session. All your breakpoints will remain intact.

-  **Stop button**

  Terminates the current VoiceXML browser session. The debugger remains open, and you are able to look through the most recent data.

-  **Step button**

  Steps forward to the next opportunity for a breakpoint. It continues from the current breakpoint, and then immediately stops at the next opportunity.

-  **Breakpoints button**

  Opens the breakpoint window.

-  **Break Immediately button**

  Breaks at the next available opportunity. It might take a few seconds to break if the VoiceXML browser is performing I/O operations, such as TTS, which cannot be interrupted.

-  **DTMF Input button**

  Opens the DTMF window, which lets you simulate pressing the buttons of a numeric keypad on a phone.

-  **Keyboard Input button**

  Opens a text window, which lets you type text into your VoiceXML program (available only when debugging in text mode). The text simulates spoken input in the debugger.

The debugger menus

The menu bar provides the access to all the functions in the debugger.

File menu

- **Close**

  Closes the current foreground source code window, if one exists.

- **Close All**

  Closes all open source code windows.

- **Exit Debugger**

  Ends the current session and closes the debugger (Ctrl+Q).

Edit menu

- **Find**

  Opens the Find window, which lets you to search the source code. **Note:** Verify that you have selected an active source window after the Find window is active so that the Find function knows where to start (Ctrl+F).

View menu

- **Variables**

  Opens the Variables viewer, which displays the system's variables and their values. Double-click a variable to set a new value. Note: Variables in the site document might not appear in the Variables viewer.

- **Evaluate Expression**

  Opens the Evaluate Expression window, which lets you type lines of ECMAScript code and see the results.

- **DTMF Input**

  Opens the DTMF window, which lets you simulate pressing the buttons of a numeric keypad on a phone.

- **Keyboard Input**

  Opens a text window, which lets you type text into your VoiceXML program (available only when debugging in text mode). The text simulates spoken input in the debugger.

- **History**

  Opens the History window, which displays the order in which every URL has been visited in this debugging session.

- **Stack Trace**

  Opens the Stack Trace window, which displays a trace of the calling documents.

Debug menu

- **Restart**

  Restarts the current session. All your breakpoints will remain intact.

- **Stop**

  Terminates the current VoiceXML browser session. The debugger remains open, and you can look through the most recent data.

- **Go/Resume**

  Starts execution of a VoiceXML application session, or resumes following a breakpoint.

- **Breakpoints**

  Sets a breakpoint on the current line (the line where the cursor is located, not the line that is highlighted).

- **Break Immediately**

  Breaks at the next available opportunity. It might take a few seconds to break if the VoiceXML browser is performing I/O operations, such as TTS, which cannot be interrupted.

Settings menu

- **Break on new file**

  When selected, causes a breakpoint to occur at the first executable statement any time a new URL is entered during execution (deselected by default). Click the option to select or deselect it.

- **Automatic Scroll**

  When selected, causes the current source code window that is executing to be in the foreground and scrolled to the appropriate line that is executing (selected by default). Click the option to select or deselect it.

- **Continuously Update Variables Window**

  When selected, the variables window updates at each line on which a breakpoint can occur (deselected by default). Selecting this option can negatively affect the performance of the application, especially for applications with large numbers of variables. When deselected, the variables window updates only when the application has been paused at a breakpoint.

Note: The default values for the Settings menu reflect the preferences specified on the VoiceXML debugger preferences window in the Voice Toolkit.

8.3 Testing the application

Now that we have completed building our VXML application, we can test our code to verify that it is working properly.

You can view the working Airline.VXML code in Appendix A, “VXML application” on page 505.

Test in text mode

You can test your code in text mode if you install the IBM WebSphere Voice Server for Windows 2000 SDK 3.1, and if your application does not use precompiled grammars. It is a good idea to test your code in text mode before testing in audio mode because you don't need to worry about the potential for recognition errors. To test in text mode (Figure 8-35 on page 419), do the following:

1. Unless you have a reason not to do so, select **File -> Save all** to make sure you have saved any changes in all of your files (only saved versions of files are used in the test).
2. If open, select the VoiceXML file from which you want to start the test. Otherwise, open the desired VoiceXML file.
3. Select **Execute -> Run in Text Mode**.
4. When the browser starts the DTMF (phone pad) simulator, click the command window so it has keyboard focus.
5. Once you see a prompt from the browser, type any phrase you want to test. If the browser posts a message that appears in the middle of your phrase while you are typing, just ignore it and keep typing until you finish the phrase. After you finish the phrase, press Enter to send it to the browser.
6. Continue typing test phrases until you finish your test. To stop the browser, presss Ctrl+C.

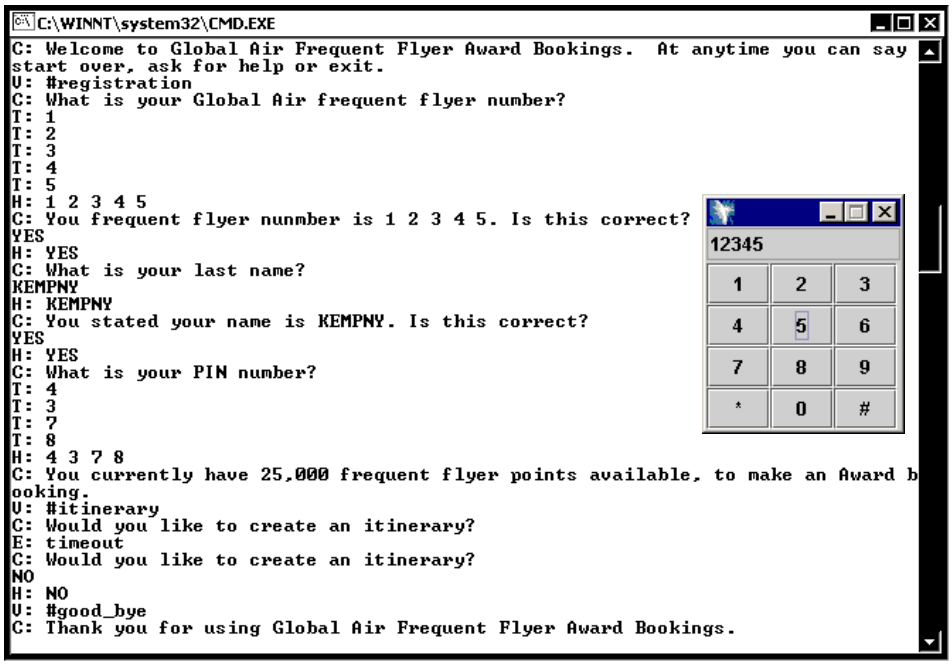


Figure 8-35 Testing in text mode output

Testing the application in audio mode

Once you are satisfied with your application's behavior in text mode, you should test your application in audio mode (Figure 8-36 on page 420). As with testing in text mode, you must install the IBM WebSphere Voice Server for Windows 2000 SDK 3.1 to test in audio mode. You must also have a microphone and speakers (or microphone headset) installed.

It's a good idea to set the audio level before starting the test. To do this, click the Windows Start button, select **Programs -> IBM WebSphere Voice Server SDK -> Audio Setup**, and follow the on-screen instructions.

To test in audio mode:

1. Unless you have a reason not to do so, select **File -> Save** all to make sure you have saved any changes in all of your files (only saved versions of files are used in the test).
2. If open, select the VoiceXML file from which you want to start the test. Otherwise, open the desired VoiceXML file.
3. From the Voice Toolkit's menu bar, select **Execute -> Run in Audio Mode**.
4. Once you hear a prompt from the browser, say any phrase you want to test.

5. Continue saying test phrases until you finish your test. To stop the browser, click the command window so it has keyboard focus, and press Ctrl+C.

Figure 8-36 Test in audio mode

VoiceSnoop Servlet

The VoiceSnoop Servlet is analogous to “snoop” programs used with a visual browser. It returns dynamic audio information to the VoiceXML browser about the requested URL, request method, URI, request protocol, path, query, server, and remote user. However, the VoiceSnoop Servlet returns this information in an audio format. You can use this servlet to ensure that your voice system is functioning properly.

When the VoiceSnoopServlet starts, it instantiates the servlet, and returns a VoiceXML page to the client. The speech browser renders this page, allowing the user to select (by speaking) different menu options. The user's audio input is translated by the IBM Speech Recognition Engine into a string, which is used to return queried information. The results are sent to the IBM text-to-speech (TTS) engine, which will render (speak) the information.

8.3.1 Access a deployed application

Once your voice applications are deployed, users simply dial the telephone number that you provide and are connected to the corresponding voice application. Figure 8-37 shows a flow chart of a typical call.

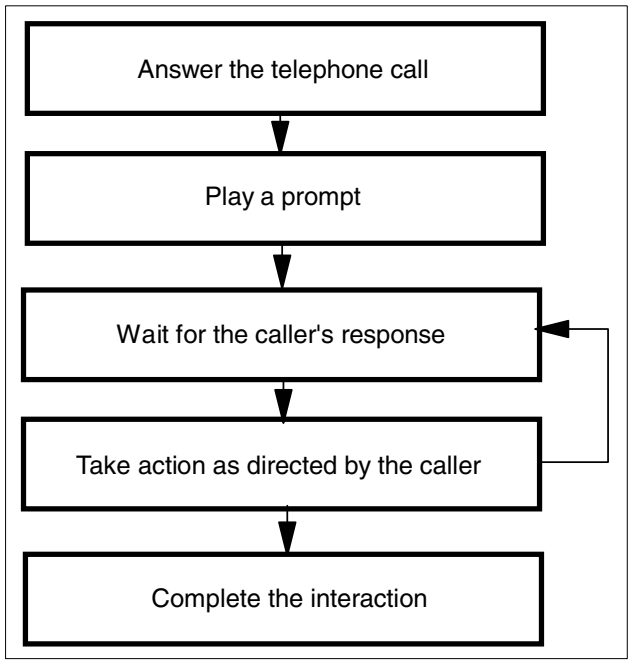


Figure 8-37 Flow chart of a typical call

1. A user dials the telephone number you provide. WebSphere Voice Server answers the call and executes the application referenced by the dialed phone number.
2. WebSphere Voice Server plays a greeting to the caller and prompts the caller to indicate what information he or she wants.
   - The application can use prerecorded greetings and prompts, or the application can have the greeting or prompt synthesized from text using the text-to-speech engine.
   - If the application supports barge-in, the caller can interrupt the prompt if he or she already knows what to do.
3. The application waits for the caller's response for a set period of time.
   - The caller can respond either by speaking or by pressing one or more keys on a DTMF telephone keypad, depending on the types of responses expected by the application.
   - If the response does not match the criteria defined by the application (such as the specific word, phrase, or digits), the voice application can prompt the caller to enter the response again, using the same or different wording.

- If the waiting period has elapsed and the caller has not responded, the application can prompt the caller again, using the same or different wording.

4. The application takes whatever action is appropriate to the caller's response. For example, the application might:
   - Update information in a database
   - Retrieve information from a database and speak it to the caller
   - Store or retrieve a voice message
   - Launch another application
   - Play a help message

   After taking action, the application prompts the caller with what to do next.
5. The caller or the application can terminate the call. For example:
   - The caller can terminate the interaction at any moment by hanging up. WebSphere Voice Server can detect if the caller hangs up and can then disconnect itself.
   - If the application permits it, the caller can use a command to indicate explicitly that the interaction is over (for example, by saying "Exit").
   - If the application has finished running, it can play a closing message and then disconnect.

   For information on deploying your voice applications, refer to the documentation provided with the applicable product:
   - *WebSphere Voice Server Version 3.1 - Use with Intel Dialogic Telephony Platform Administrator's Guide*, G210-1263, for information on deploying VoiceXML applications in a Dialogic environment.
   - *WebSphere Voice Server Version 3.1 - Use with Cisco Telephony Platform Administrator's Guide*, G210-1262, for information on deploying VoiceXML applications in a Voice over IP environment.
   - *WebSphere Voice Server Version 3.1 - Use with DirectTalk for AIX Telephony Platform Administrator's Guide*, G210-1259, for information on deploying VoiceXML applications in a DirectTalk environment.

It is recommended that you use one of the IBM WebSphere Voice Server Version 3.1 products; however, other vendors' VoiceXML deployment environments may also work.

8.4 WebSphere Studio

IBM WebSphere Studio is a Web development team environment for organizing and managing Web projects. IBM WebSphere Studio Entry Edition Version 4.0 has been enhanced to support VoiceXML files.

If you are developing your applications using IBM WebSphere Studio Entry Edition Version 4.0, you can start the VoiceXML browser by right-clicking a VXML file in any Studio view and then selecting **Preview file with -> WebSphere Voice Server** (Figure 8-38 on page 423).

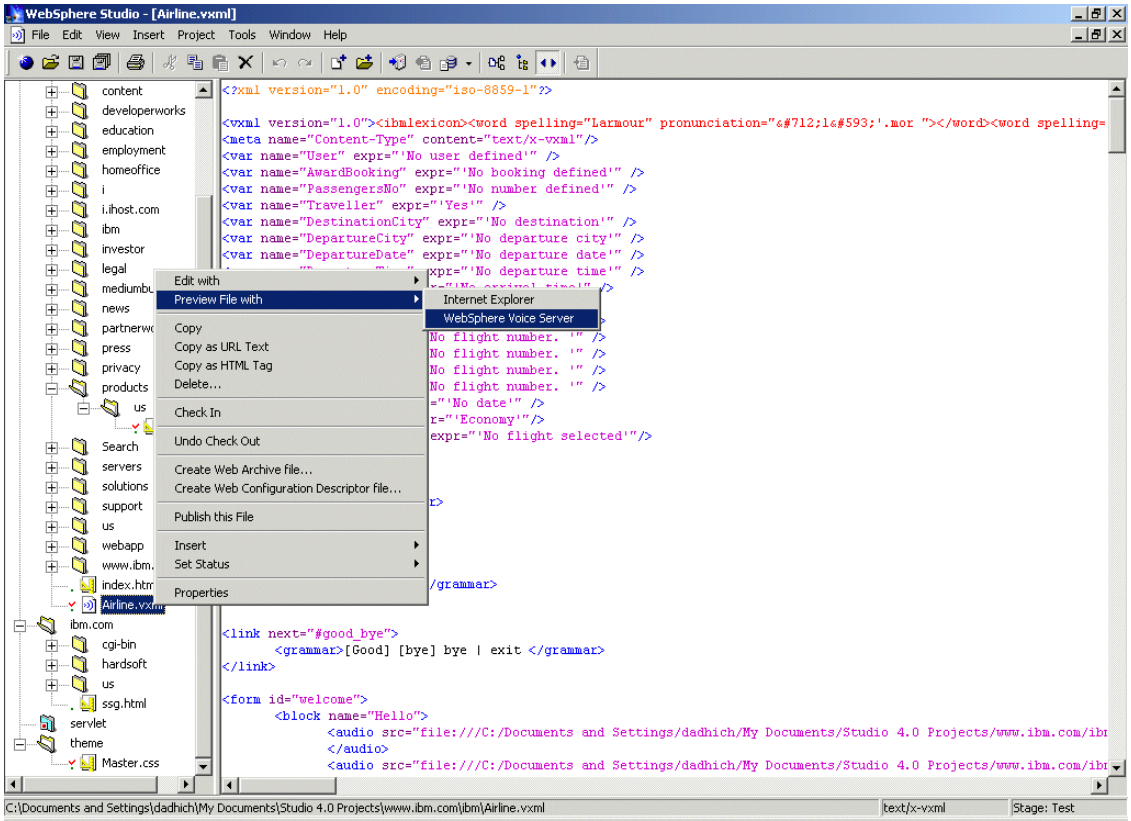


Figure 8-38 Preview VXML from WebSphere Studio Entry Edition

Studio is preconfigured to execute the first VoiceXML browser installed. For example, if you installed the US English and UK English language versions of the SDK, Studio launches the batch file vsaudio_en_US.bat (US English batch file), which invokes the VoiceXML browser and passes it a URL that refers to the selected file.

To execute the VoiceXML browser in another language, you must edit the setting in the system registry. To edit this setting, type `regedit` in a DOS command prompt and search for the key name:

My Computer\HKEY_LOCALE_MACHINE\SOFTWARE\IBM\WebSphere Voice Server SDK.

Change the value of the sub key “previewer” by right clicking and selecting **Modify**. Point to the desired file, for example vsaudio_en_GB.bat for UK English.

For more information on using the VoiceXML browser with IBM WebSphere Studio, refer to the IBM WebSphere Studio documentation.

8.5 Utilities

There are several utilities that can be used to help with developing applications for WebSphere Voice Server, including the following:

- NetMeeting

  NetMeeting can be useful in testing the WebSphere Voice Server (Cisco) platform. This platform uses Voice over IP (VoIP) and so you can connect to the VoIP service from an H.323 client application such as NetMeeting.

- Coolwave, Goldedit, Soundforge, and other professional sound recorders

  When recording audio for production systems, it is recommended that you use a professional recording studio or at least a high-quality recording environment. If you are setting up your own recording environment, you should consider a professional sound recording program such as Coolwave, Goldedit, or Soundforge that allow recorded audio to be filtered and edited. Many of these editors will integrate with the Voice Toolkit.

8.5.1 Multiple interface and other design considerations

When creating applications for multiple users interfaces, it is important to pay special attention to the application design. This can save a great deal of effort later in the project. There are many approaches to successful user interface and data abstraction. For Voice XML projects, a popular solution is to abstract the application data into an XML layer and use XML style sheets to produce the Voice XML files.

When planning Voice Server applications, sufficient time should be allocated for developing and testing application grammar, especially for non-US English applications. Many voice applications require lengthy testing and grammar tuning to get the recognition rate for the application across a broad range of customers to a commercial level.

Good design for most voice applications is focused around limiting the scope for voice input where possible. Where possible, prompts should recognize the smallest grammar set possible. This improves the recognition rate and allows effort to be concentrated on the words the application must understand and the overall application flow. There are a number of built-in datatype grammars to take advantage of when designing prompts. When accepting secure input such as credit card numbers, DTMF input should be allowed for those who do not wish to say their credit card number out loud.

As with other server applications, the number of calls a system must accept and the frequency of calls should be considered when designing applications. Large grammars and deeply nested grammar descriptions require more system resources.

8.5.2 Related publications

Information on creating VoiceXML voice applications is available from a variety of sources, including:

- *IBM WebSphere Voice Server Software Developers Kit (SDK) Programmer's Guide,* Version 3.1, which provides information on creating and testing voice applications. This document is part of the IBM WebSphere Voice Server SDK Version 3.1, which is packaged with the IBM WebSphere Voice Server Version 3.1 and can be downloaded from:

  `http://www-4.ibm.com/software/webservers/voiceserver/library.html`

- VoiceXML 1.0 Specification, available from the VoiceXML Forum Web site at:

  `http://www.voicexml.org`

  If you've installed the IBM WebSphere Voice Server SDK Version 3.1, it is accessible from the Windows Start menu by choosing **Programs -> IBM WebSphere Voice Server SDK -> VoiceXML 1.0 Specification**.

- Java Speech Grammar Format (JSGF) Specification, available at:

  http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html

- *ECMA Standard 262: ECMAScript Language Specification*, Third Edition, published by ECMA at:

  `http://www.ecma.ch/ecma1/stand/ECMA-262.htm`

9

VoiceXML application development using WebSphere Transcoding Publisher

In this chapter, we discuss WebSphere Transcoding Publisher and discuss the ways in which Transcoding Publisher can help you leverage existing content and facilitate VoiceXML development.

9.1 WebSphere Transcoding Publisher

IBM WebSphere Transcoding Publisher is a software solution to help simplify the complexity of the wireless Internet. It is server-based software that can filter, adapt, and reformat Web-based data to a variety of devices and environments. It can handle any type of input Web data, including HTML pages from a Web server, HTML output produced from a host application such as IBM's Host Publisher, and XML data from a back-end transaction system. Part of WebSphere Transcoding Publisher are the transcoders. These enable Transcoding Publisher to automatically convert data on-the-fly during transmission, transforming content written in HTML or XML into the language used by the end user's device. In our environment, we are interested in VoiceXML. WebSphere Transcoding Publisher provides a transcoder that can convert Web content into the format needed for VoiceXML-capable browsers. The transcoder converts HTML input into VoiceXML, which can, in turn, be converted to speech by a voice browser such as the one provided with the IBM WebSphere Voice Server.

WebSphere Transcoding Publisher can be deployed in several ways, either as a network proxy, as a filter (servlet) running in the IBM WebSphere Application Server, as JavaBeans, or as a plug-in to the WebSphere Edge Server caching proxy. The software is available for AIX, Linux, Solaris, Microsoft Windows NT and Windows 2000 operating systems. A separate version of WebSphere Transcoding Publisher supports IBM's iSeries platform.

9.2 How WebSphere Transcoding Publisher can be used

In most situations, you will want to code your voice applications natively in VoiceXML using the IBM WebSphere Voice Server SDK or an equivalent product. If, however, you fall into one of the following categories, you may find using WebSphere Transcoding Publisher to be a more productive solution, even in combination with a native VoiceXML application:

- You already have an existing visual Web site (in HTML) or data in XML.
- You want to present the same basic data on multiple output devices, one of which is a telephone.

To WebSphere Transcoding Publisher, VoiceXML is just another format, and the voice browsers that interpret VoiceXML into spoken text are just new devices to support. There are three basic ways to use WebSphere Transcoding Publisher to produce a voice application, or content for voice applications:

1. Converting XML data directly to VoiceXML through the application of XML style sheets.

2. Converting a Web site into a VoiceXML application.
3. Mining Web content for reuse in a VoiceXML application.

Each scenario has differing requirements that need to be considered before its selection.

9.2.1 Converting XML to VoiceXML

If XML source is available, WebSphere Transcoding Publisher can use its XMLHandler transcoder to apply user-developed style sheets. The XML data can then be converted into VoiceXML. The HTML-to-VoiceXML transcoder is not used. The benefit here is that there is no need to use HTML style guidelines and the annotation function.

However, if the style sheet applied to the XML produces HTML, you will then need to use the HTML-to-VoiceXML transcoder. This transcoder converts the HTML to a device-specific format. In this case it will be VoiceXML. Using this approach, the HTML becomes a common intermediate format from which more specific formats can be more easily derived.

Without a common intermediate format, converting XML data to several possible presentation formats implies having to generate a different style sheet for each resulting markup language, and a new set of such style sheets for each XML document. Creating a single style sheet that converts XML to HTML allows the various transcoders in WebSphere Transcoding Publisher to render in the desired markup language, such as VoiceXML and WML, as shown in Figure 9-1.

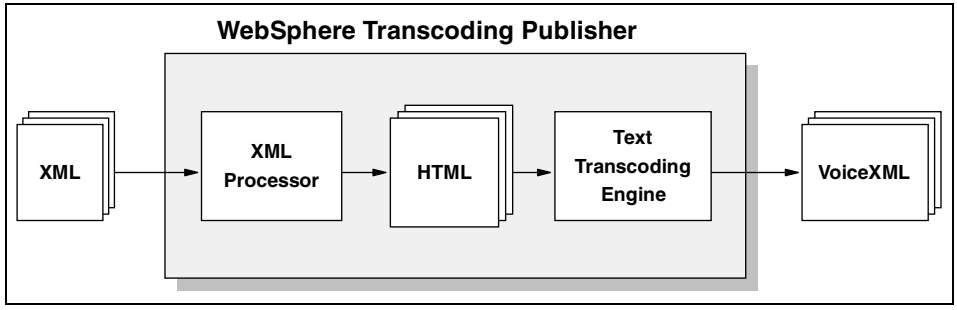


Figure 9-1 Using a single XML to HTML style sheet

Note: You need to consider carefully how the HTML will be formed if VoiceXML is one of your target formats.

9.2.2 HTML to VoiceXML transcoding

You may have a Web site that has had a considerable amount of effort and resources used to build it. This is where the HTML to VoiceXML transcoder can be utilized to voice-enable your site.

However, the visual interface of your Web sites differ greatly from an audio interface. Consideration must be taken when deciding how to approach this task. There are various ways to voice-enable your site, including:

- Using various WebSphere Transcoding Publisher functions to simplify your content to make it easier to transcode into VoiceXML.
- Reworking the HTML and the Web site in general to make it easier to traverse using voice.
- A combination of both.

There are functions available within WebSphere Transcoding Publisher that can simplify an HTML page and make it more usable as a voice application. This includes the use of annotation, clipping, or custom response editors. However, this would need to be applied on a page-by-page basis. WebSphere Transcoding Publisher functionality can only help improve the page, not the entire site as a whole. This is where your Web site design comes into play. Complex navigation bars and graphic hotspots may be too complex or even lost (you cannot hear a graphic) when transcoded into a voice medium.

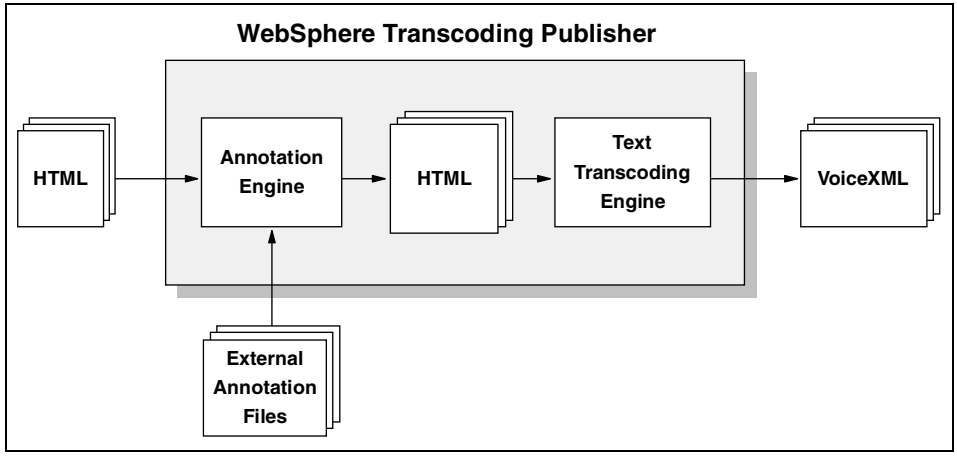


Figure 9-2 Transcoding HTML to VoiceXML

9.2.3 Mining content from HTML pages

What voice-enabling dynamic data may your Web site use? Potentially this is the information your customers are after. Examples may be stock quotes, phone number listings, or weather reports. In the voice environment, customers need to hear this rather than see it. The HTML-to-VoiceXML transcoder could be used to convert HTML content mined from existing pages that host dynamic data, and use it within a VoiceXML application.

The way to build this would be to maintain a separate VoiceXML application from its visual counterpart. There still is a level of difficulty to incorporating dynamic data within a static VoiceXML application. It would be easier for the VoiceXML application designer to reuse existing HTML data than to reproduce that data from some other source.

This is where WebSphere Transcoding Publisher comes into the picture. Not only does it provide the HTML-to-VoiceXML transcoder to render the HTML source, but it also has the necessary tools to extract the required information from a Web page filled with data that is useless in a voice environment, for example, graphics. The HTML-to-VoiceXML transcoder can be used to derive either entire VoiceXML pages for use within the voice application, or content portions that can be placed within a VoiceXML page.

Figure 9-3 on page 432 illustrates the concept of content mining.

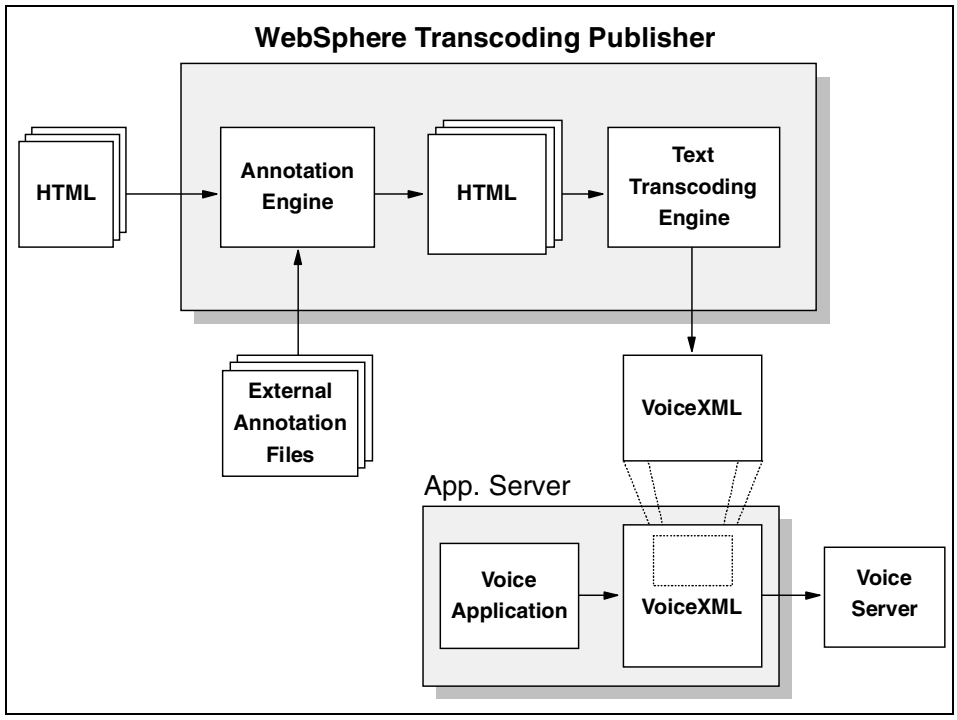


Figure 9-3 HTML content mining

9.2.4 The HTML-to-VoiceXML transcoder

The HTML-to-VoiceXML transcoder converts HTML to VoiceXML. When a VoiceXML-capable client requests an HTML document through WebSphere Transcoding Publisher, WebSphere Transcoding Publisher understands that the device expects VoiceXML content by the AcceptHeader portion of the HTTP <request>. It then converts the requested HTML document into VoiceXML using the transcoder, and sends the resulting document back to the client. The HTML-to-VoiceXML transcoder does not actually render the VoiceXML document into spoken text; that is the responsibility of the voice browser. To access the applications, users simply dial the telephone number that you provide and are connected to the corresponding voice application.

The transcoder does more than convert HTML tags to VoiceXML tags. It analyzes the HTML document and splits it into three sections:

1. Document body. This is further divided into subsections based on heading tags (<h1>,<h2>).

2. Links section. All the links on the original page are extracted from the source and provided here in a list.
3. Content section. This is a series of generated links that prompt the user to the main page content (first section), a links list (second section), or to exit the application.

Note: Main navigational links are inserted by the HTML-to-VoiceXML transcoder in the language of the requester. If the client requests a language that WebSphere Transcoding Publisher does not support, then English links are generated.

Isolating main content

The main content of the page is determined by the use of heading tags. It is assumed they indicate the primary subject matter for the page. If the document does not contain any heading tags, then the entire document is assumed to be the main content. All imbedded URL links are removed and added to a list of links on the page, or this list can be read separately from the main content. Imbedded images and other objects are also removed, leaving only the text to be spoken.

The first three words of the text between the heading tags are turned into links to the subject matter under that heading. These links are collected and listed at the top of this section to give the user a choice of what content to have read, as opposed to listening to the entire document. The end of each headed section contains a link back to the content list. The last link of the content list is a link back to the main prompt section.

Note: If the data you are transcoding isn't structured into paragraphs and topics, such as tabular data (for example, stock quotes or part numbers), it may not contain any header tags and so the main content will not be prefixed with any navigational links.

List of links found on the page

All HREF links found on the page are collected and added to a list that is voiced separately from the main content on the page, based on a selection made from the main menu. If the links are to remain embedded within the main content, then the voicing of the content would be interrupted with prompts to take that link to another page.

It is worth noting that the list of links on a page will contain both links embedded within text as well as those part of navigation areas. It may be difficult to understand the relevance of the link if taken out of context of the text around it.

List grammars

The easiest input for a voice application to process is a selection list, where the choices are limited to a finite set of items. The user selects an item by simply speaking it. When processing an input form, the HTML-to-VoiceXML transcoder generates a grammar out of the options under the <SELECT> tag and inserts that grammar into the result.

Tables

A table is a two-dimensional organization of data. Since voice applications are one dimensional, there is no concept of a table representation in voice. The HTML-to-VoiceXML transcoder converts tables to lists before being converted to VoiceXML. This function flattens a table vertically listing the table cells' contents row by row, from left to right. An original table would look like Table 9-1.

Table 9-1 simple table

	Breakfast	**Lunch**	**Dinner**
Monday	Cereal	Sandwich	Pizza
Tuesday	Eggs and bacon	Soup	Meatloaf
Wednesday	Pancakes	Salad	Steak

This would be transcoded to be read to the listener as follows:

```
Breakfast Lunch Dinner, Monday Cereal Sandwich Pizza, Tuesday Eggs and bacon
Soup Meatloaf, Wednesday Pancakes Salad Steak.
```

Use annotation to simplify a table before it gets converted into a list like the above. The removal of entire rows or columns will help limit the amount of information the end user has to process, focusing instead on the most important data.

Common command support

Voice Server has built-in commands of Help, Quiet\Cancel, and Repeat. These are always active and operate independent of the content being read. Two other common commands, Backup and Start Over, will execute depending on how they are used in the source. The points that you can back up to, or where you go when you start over, are specific to the document and cannot be automatically implemented.

WebSphere Transcoding Publisher lets you insert grammars and links for these common commands using a Java clipper. Refer to the *WebSphere Transcoding Publisher Developer's Guide* for more information on how to develop text clippers.

Barge-in

Barge-in is the ability of the user to interrupt prompts being read and provide a command or selection from a menu. The behavior of a barge-in is defined within Voice Server. However, it can be enabled and disabled on a per-prompt basis. Using the same Java clipper logic as described in the previous section, you can insert <prompt bargein="true/false"> wherever applicable in the transcoded document.

Tip: Disable barge-in if you are reading important information, such as legal notices or advertisements, and enable it for long sections of text.

9.3 Using annotation process

Annotation basically is a process of modifying an HTML document using XML-based language instructions. These alter the HTML document by way of clipping regions of a page, replacing existing content, or even inserting new content.

If an HTML page will not transcode to VoiceXML well enough for a usable page, annotation may be the solution. It is the easiest way to simplify the HTML page before it is transcoded, without affecting the original content.

Internal annotation

There are two levels of annotation, internal and external. Internal annotation resides within the HTML source. External annotation is provided using an external file, which gets applied to the HTML file at runtime.

Internal annotation instructions are maintained with the rest of the HTML document. The instructions are imbedded as HTML comments, which are interpreted only by WebSphere Transcoding Publisher's annotation engine. If the source or structure of the document were to change, the internal annotations could be synchronized with the changes.

Internal annotation presumes you have control over the HTML content and can add the annotations. If you do not have such control, then you will have to use external annotation.

External annotation

External annotation instructions are kept in an external file, but are the same as internal annotations; they simply reside in a separate file. Using WebSphere Transcoding Publisher administration, the external annotation file is associated with the HTML document's URL against which its instructions are to be applied.

The disadvantage is that this file must be maintained separately from the HTML source. Any changes made to the HTML source require changes to the annotation file to keep it current. The benefits are that you don't have to own the HTML source to support annotation and you can write annotations that can be applied to multiple HTML files.

Note: The complimentary copy of WebSphere Studio HTML Editor shipped with Voice Server allows internal annotations to be built visually based on clipping regions that you mark in the graphic view of the page.

9.3.1 Content clipping

The most common use of annotation is for clipping regions of the original HTML document, leaving only the portions you wish to convert to VoiceXML. The annotation instructions identify the tags, or nodes, that mark the start and end of the clipping regions.

These tag locations are specified in terms of their XPATH. Before the annotations are processed, the HTML document is parsed into a document object model (DOM) where it is represented as a tree, where each node in the tree is a tag in the original HTML. The XPATH of a node is the unique path to that node in the DOM tree.

A common use of annotation is to clip table regions, specifying rows and/or columns to remove. This is an easy method of simplifying data presentation, or even page layout. Here is an example of some simple annotation statements for clipping regions of a document.

Table annotation

Example 9-1 demonstrates a set of instructions that will keep all but the first column and second row of the first table in the document.

Example 9-1 Table annotation

```
<description take-effect="before" target="/descendant::TABLE[1]">
<keep />
<table majoraxis="row">
<column index="1" clipping="remove" />
<column index="*" clipping="keep" />
<row index="*" clipping="keep" />
<row index="2" clipping="remove" />
</table>
</description>
```

Text clipping

Example 9-2 shows a set of instructions that will remove all content after the first table in the document. Content will continue to be clipped until a <description> with a <keep /> instruction is encountered.

Example 9-2 Text clipping

```
<description take-effect="after" target="/descendant::TABLE[1]">
<remove />
</description>
<description take-effect="before" target="/descendant::TABLE[2]">
<keep />
</description>
```

9.3.2 Form simplification

Most forms on Web sites contain several optional fields. These could actually be clipped out of the content, thus reducing the amount of input required from the listener. For other input controls, you might be able to identify what typically is provided as input the majority of the time. You could default the value of that input and eliminate the control all together. The less information required of the listener to complete the task, the better their experience with the interface will be. You could also replace a text input control to limit the possible values to a list.

Content replacement

Sometimes using annotation alone to clip out unnecessary regions of the document before transcoding to VoiceXML may not be enough to produce usable results. Certain elements of the original content may need to be replaced all together with new content designed specifically for the listener.

A good example of this is converting a text input control to a select list to limit the choices the user has as input. This eliminates the problem of having to recognize spelled input. Example 9-3 replaces the CITY text input control with a SELECT control with three options.

Example 9-3 City select

```
<field name="CITY" type="SELECT">
<option value="Sydney" label="Sydney" />
<option value="Melbourne" label="Melbourne" />
<option value="Brisbane" label="Brisbane" />
</field>
```

In the original HTML input form, the user could type in any city. After applying this annotation, the choices for the city are limited to picking from a list of three: Sydney, Melbourne and Brisbane. Alternatively, you can use annotation to replace an entry form with a new one as shown in Example 9-4.

Example 9-4 City annotation

```
<description take-effect="before" target="/descendant::FORM[1]">
<replace>
<form>
<text>NEW prompt for INPUTTEXT3(now a select)</text>
<field name="INPUTTEXT3" type="SELECT">
<option value="No" label="No." />
<option value="Yes" label="Yes." />
<option value="Maybe" label="Don't Care" />
</field>
<field type="SUBMIT" />
</form>
</replace>
</description>
```

You can also replace text between tags in the document with alternative text. Example 9-5 changes the title text of the document.

Example 9-5 Alternative text

```
<description take-effect="after" target="/descendant::TITLE[1]">
<replace>
<text>IBM Stock Quote</text>
</replace>
</description>
```

9.4 HTML to VoiceXML transcoding limitations

The HTML-to-VoiceXML transcoder cannot predict the intention of all elements within an HTML page. There are potential issues that are specific to voice applications that may need to be resolved by the developer of this system, possibly through the use of facilities provided by WebSphere Transcoding Publisher.

Grammars for text input in forms

The HTML-to-VoiceXML transcoder has no way of knowing what type of input is expected for a text box control. It therefore cannot generate a grammar for it. Text input using voice is not impossible, such as for numeric entries, so the HTML-to-VoiceXML transcoder preserves these text prompts in the VoiceXML, leaving it up to the developer to tag the prompts with the appropriate grammar, replace them with a selection list, or remove them all together.

Since the Voice Server requires a grammar, you must specify one for each input control, using either annotation or Java clippers.

Multi-selection controls in forms

Some input controls in HTML allow multiple selections, such as <SELECT> tags with the MULTIPLE attribute set, or convey state, such as checkboxes. Since VoiceXML applications don't allow for more than one choice to be made from a list at a time or for an item to be checked, the MULTIPLE attribute is removed from select controls and checkboxes are omitted. In general, however, all controls of types text, input, select, and option are preserved.

Elements which are difficult to speak

It is inevitable that the voice browser will find words that it does not understand how to pronounce and will resort to spelling it. The most common occurrences include:

- **Acronyms**

  If you are accustomed to hearing an acronym spoken as a word instead of spelled (for example, WYSIWYG stands for "what you see is what you get", but is normally pronounced "wizzi-wig") or spelled instead of spoken, then you might not recognize the acronym when the reverse is done by the voice browser.

  If you have access to the HTML source, then the number of such problems can be reduced by substituting the actual words for the acronym. You can also use annotation or Java clippers to insert <sayas> tags to instruct the Voice Server on how to pronounce certain words or acronyms.

- **Spelled URL links**

  It is not uncommon for the text of an anchor tag to match its associated HREF URL link, so that the user of the page actually clicks the address to be taken there. Besides the spoken address of a URL link being hard to understand, it isn't clear at all what the significance of the link is, especially since this link will be part of a list of links optionally read to you for selection.

Annotation can be used to replace the anchor tag text with a meaningful description, or, if you have access to the source yourself, you can specify your own description.

Unsupported embedded types

Since the voice interface cannot support visual elements, all embedded graphical source is removed, such as images, video, and other forms of multimedia. There is also no way to specify recorded audio files as embedded source within VoiceXML, so audio source is also removed, such as WAV and MID files.

9.5 Creating a WebSphere Transcoding Publisher project

This section describes setting up a simple WebSphere Transcoding Publisher project to use with WebSphere Voice Server.

1. First install WebSphere Transcoding Publisher onto your machine. Run the setup executable file and follow the install instructions.
2. Once completed, click the **Server Setup** icon in the WebSphere Transcoding Publisher folder. The setup window shown in Figure 9-4 on page 441 will appear. A reverse proxy is the simplest configuration. Select this and then click **Next**.

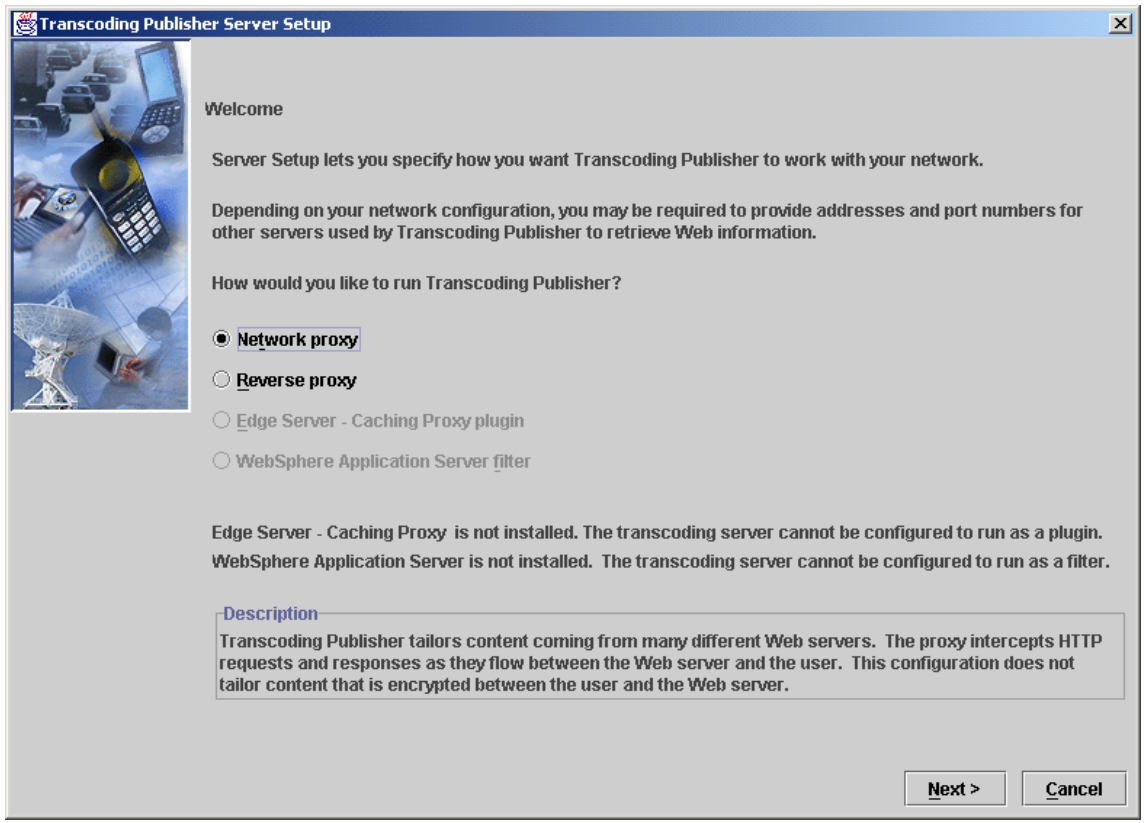


Figure 9-4 Setup window

A reverse proxy is ideal when you only need to access a single Web site for generating VoiceXML. If your applications runs across multiple Web sites, you may need to set up a Network proxy instead.

3. Our environment did not have a firewall. If your site does, enter its information in this window, as shown in Figure 9-5 on page 442. Click **Next** when completed.

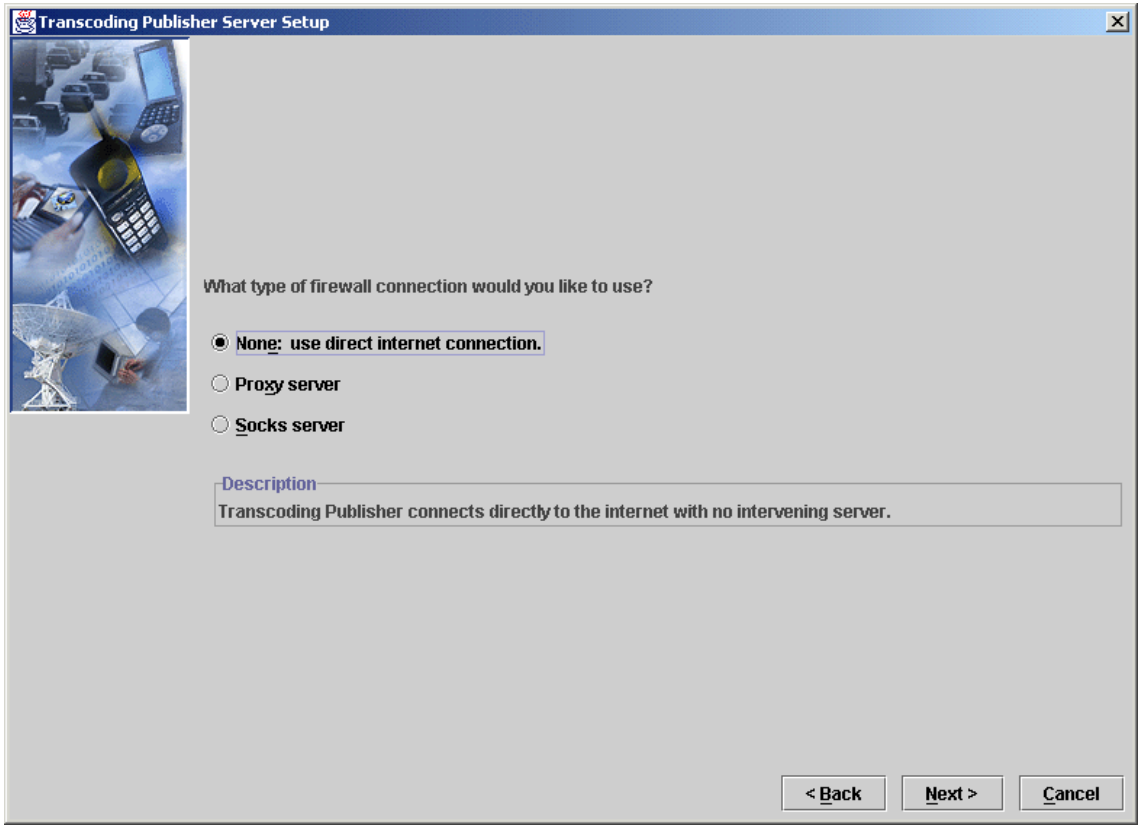


Figure 9-5 Firewall setup

Our configuration was for a single Web server. The options are displayed in Figure 9-6 on page 443. Select this option and click **Next**.

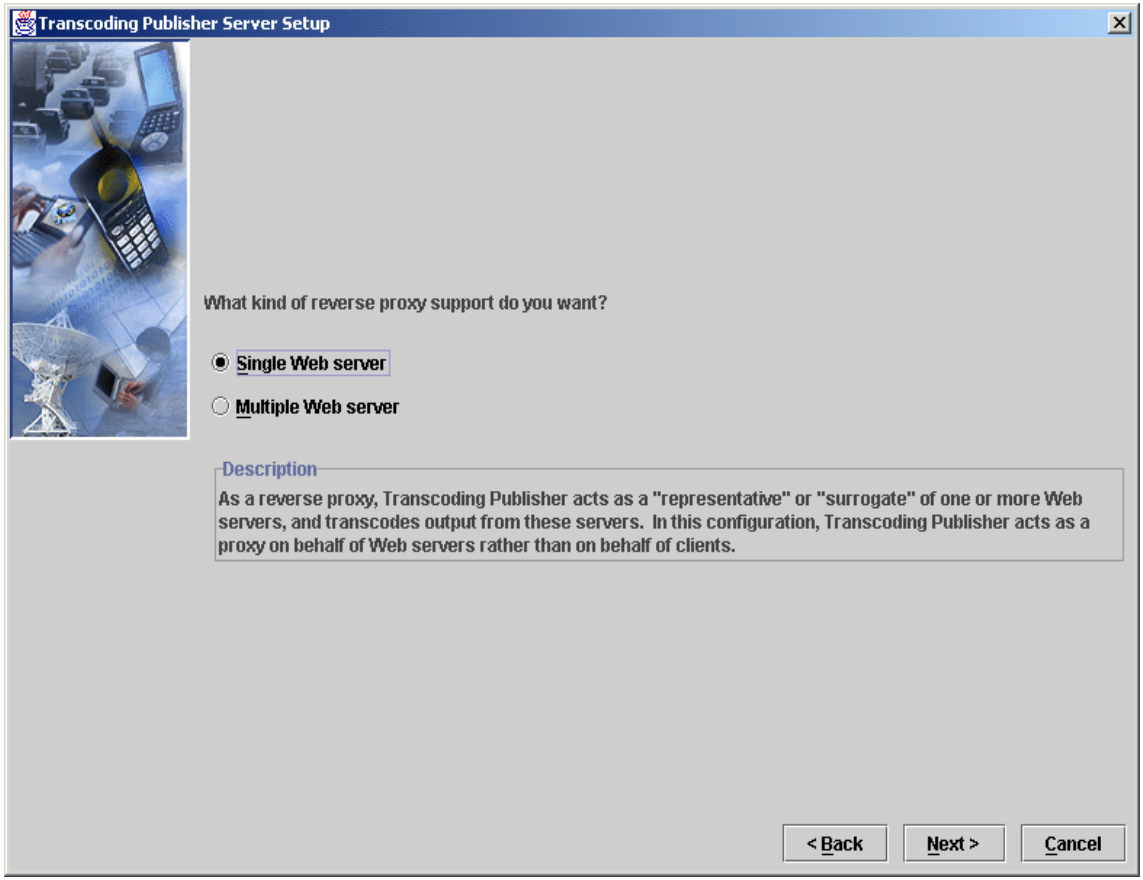


Figure 9-6 Web server configuration

4. Figure 9-7 on page 444 displays the address page for your WebSphere Transcoding Publisher and Web server. These addresses are required. Enter these in and click **Next** to continue.

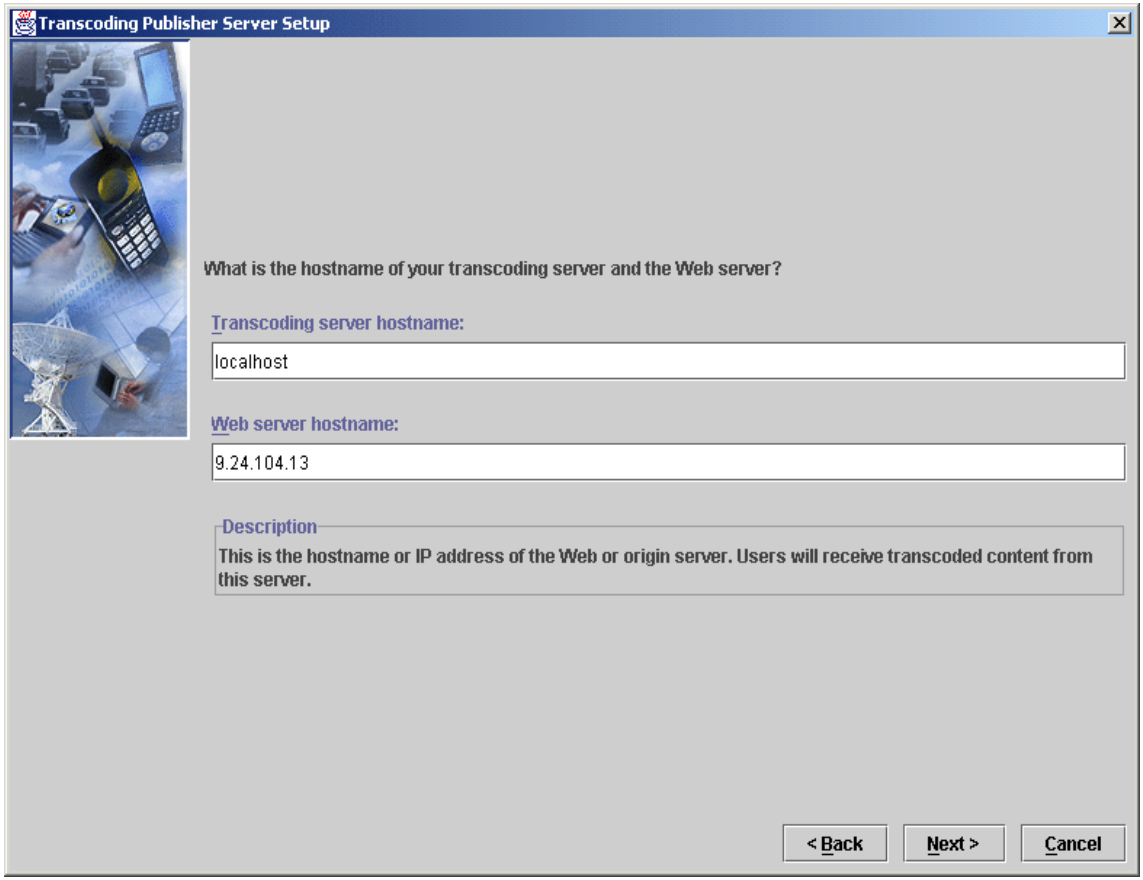


Figure 9-7 Server addresses

5. WebSphere Transcoding Publisher has three port settings, each for a specific network connection. When WebSphere Transcoding Publisher returns requested data from a set port address, the content is filtered by the ports settings. Figure 9-8 on page 445 displays the default values, which are what we used. Enter your values and click **Next**.

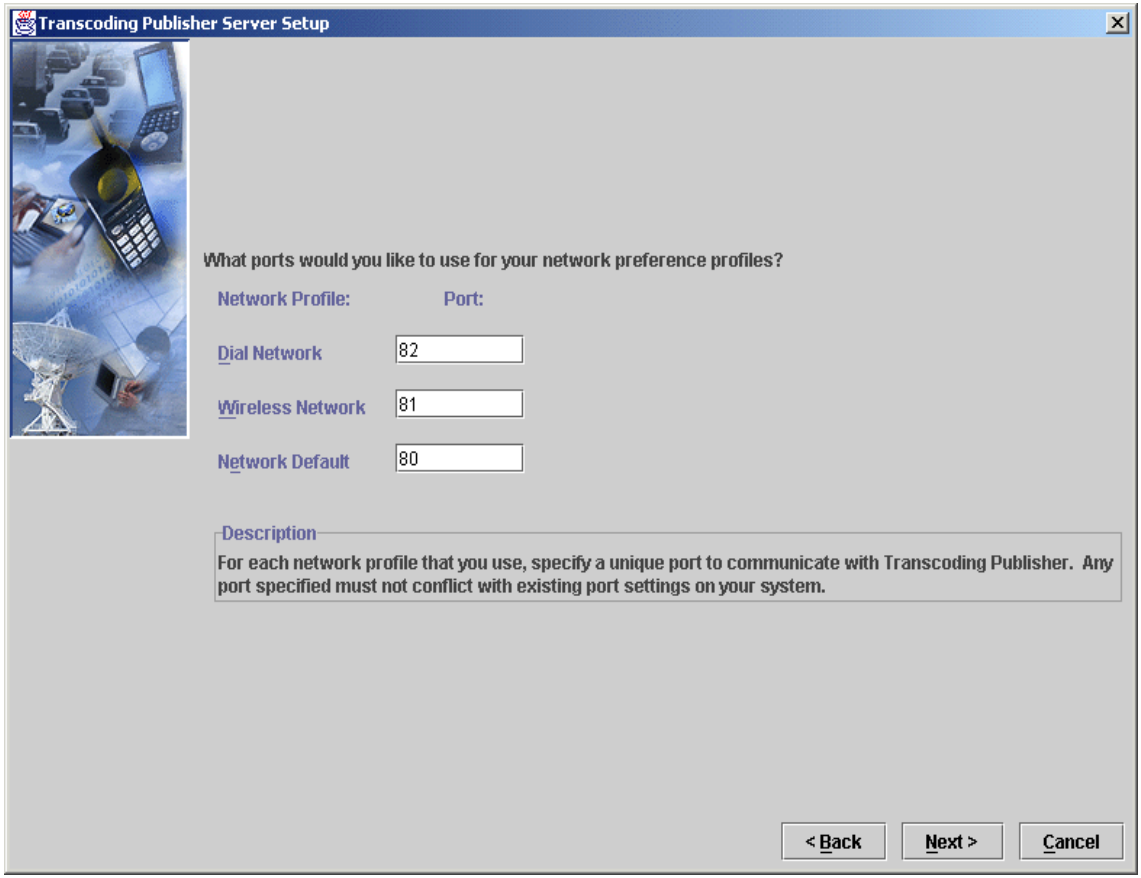


Figure 9-8 Port settings

6. The WebSphere Transcoding Publisher installation is complete, as illustrated in Figure 9-9 on page 446. Click **Finish**.

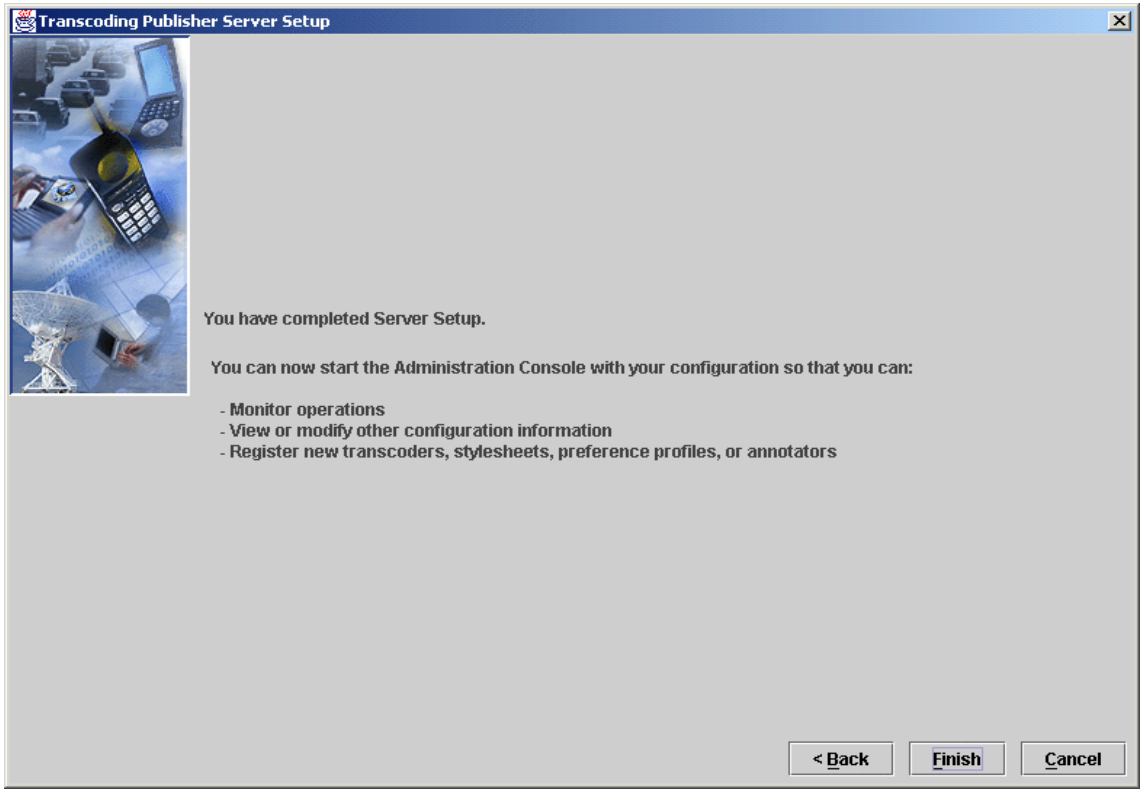


Figure 9-9 Finish window

This completes setting up a project in WebSphere Transcoding Publisher. At this stage, requests made through the WebSphere Transcoding Publisher server will result in ongoing requests being made to the configured Web server and the resulting data will be translated by the default WebSphere Transcoding Publisher rules. By default there are no rules enabled to process into VoiceXML, so it is necessary to configure WebSphere Transcoding Publisher for a VoiceXML client.

9.6 Configuring WebSphere Transcoding Publisher

Once WebSphere Transcoding Publisher has been installed, the Administration Console can be started by clicking the **Administration Console** icon in the IBM Transcoding Publisher folder. You will see a window as shown in Figure 9-10 on page 447.

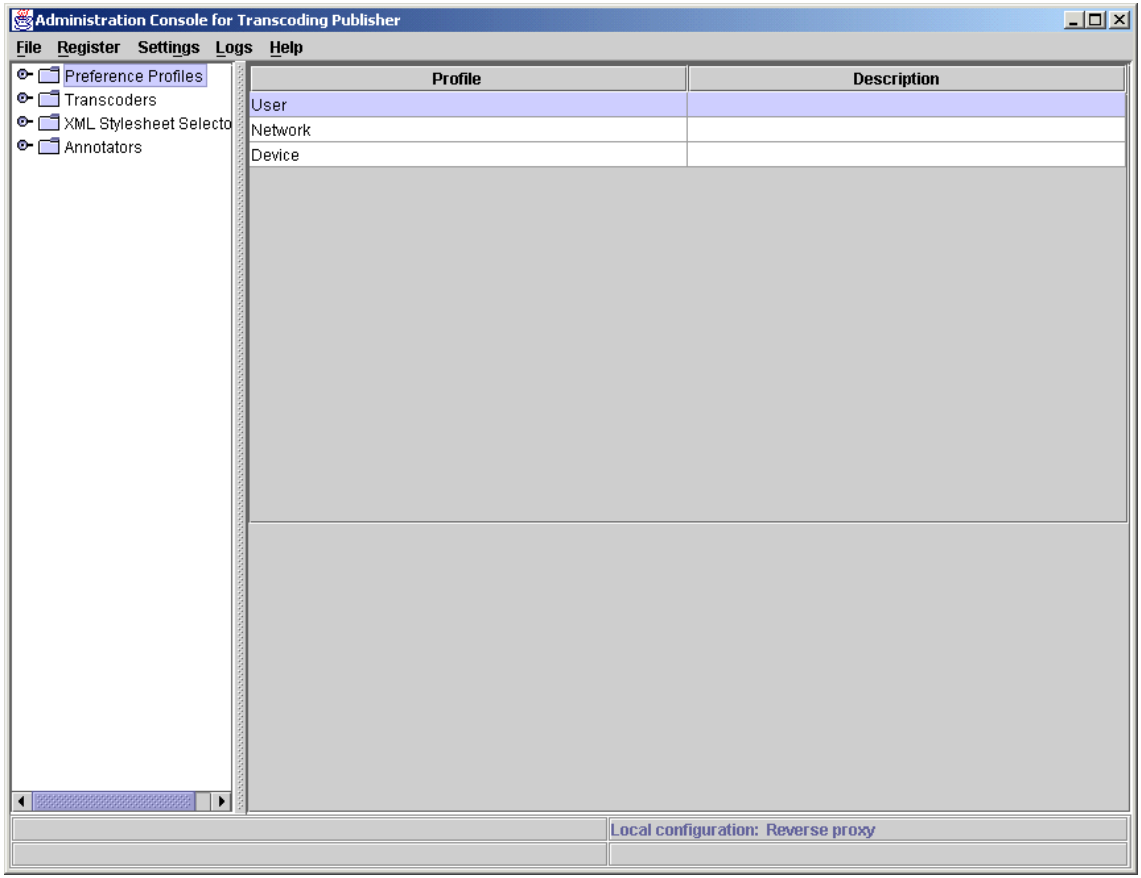


Figure 9-10 Administration Console

To configure WebSphere Transcoding Publisher, do the following:

1. When a device or browser makes a request for data, the actual request has the device type in the header field. From this information WebSphere Transcoding Publisher will then return the requested data in the format that devices understand. So, if it were a mobile phone, WebSphere Transcoding Publisher would return data in WML format. In our case the application is a voice browser and only understands VoiceXML. In Figure 9-11 on page 448, the voice browser profile is illustrated. Under the Advanced section, shown in Figure 9-12 on page 449, the header identifier for a voice browser is shown. Click **OK** to return to the main Administration Console.

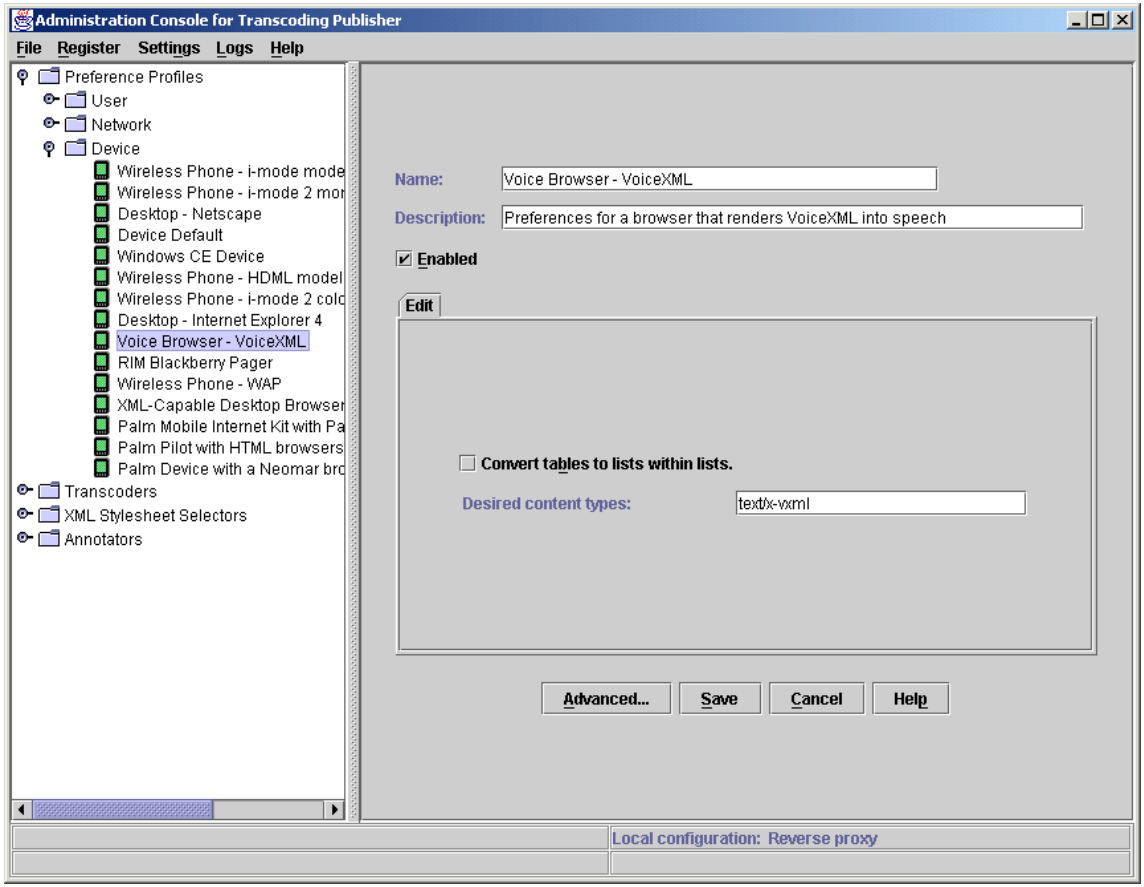


Figure 9-11 Voice browser device

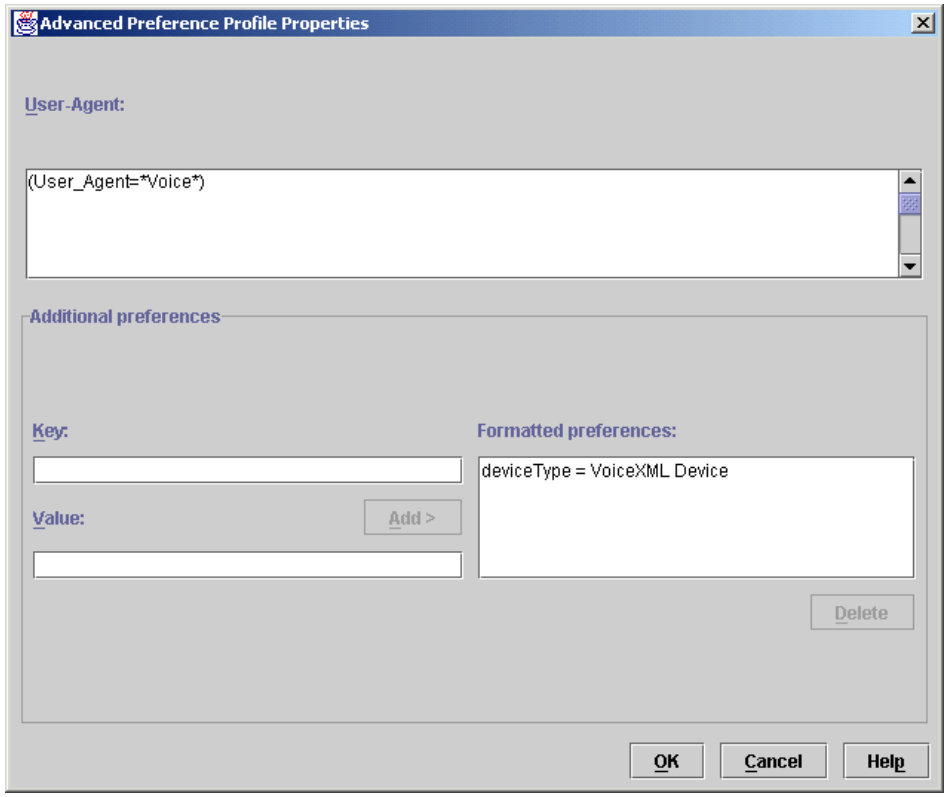


Figure 9-12 Header identifier

2. By default the HTML-to-VoiceXML Transcoder is disabled. This needs to be enabled and the WebSphere Transcoding Publisher server needs to be refreshed to accept the new changes. Click **Transcoders** to expand. Then Expand **ibm**. The HTML-to-VoiceXML Transcoder will have a red cross in a square. This means it is disabled. Select this and enable the Transcoder by clicking the **Enable** option. Click **Save**. Notice how one transcoder change is pending, as illustrated in Figure 9-13 on page 450.

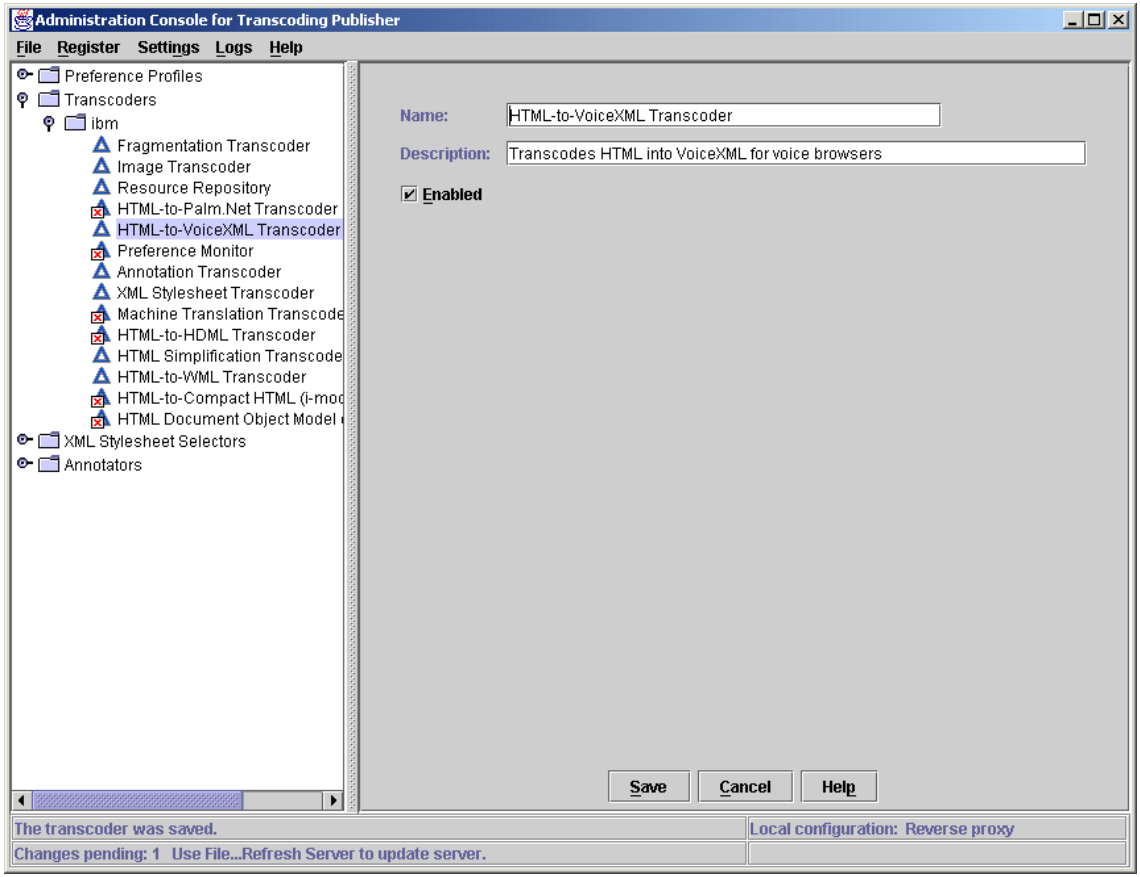


Figure 9-13 Transcoder enabled pending

3. To update the new setting, the server needs to be refreshed. This is done by clicking **File -> Server**, as shown in Figure 9-14 on page 451. Once done, any changes made are now active. WebSphere Transcoding Publisher is now ready to transcode into VoiceXML and HTML requests made by a voice browser.

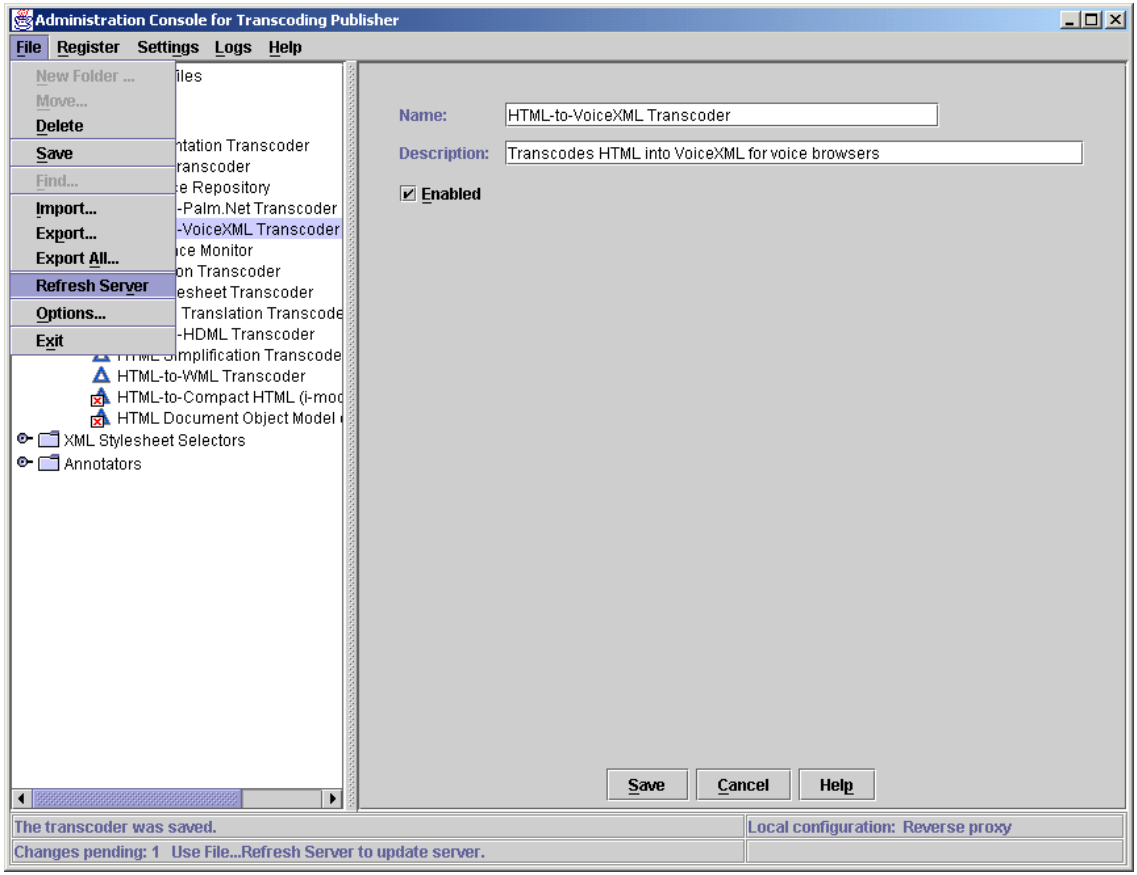


Figure 9-14 Refresh server

4. To complete this configuration you will need to configure your WebSphere Voice Server configuration to point to the root Web page of your WebSphere Transcoding Publisher project. That is, if your real Web site root page is called `http://www.mysite.com/app/index.html` and the WebSphere Transcoding Publisher server is configured to transcode data from `http://www.mysite.com`, then the root VoiceXML page will be `http://mytranscoder/app/index.html`. The return data from this request will be VoiceXML.

For most sites, the raw VoiceXML data that is returned by transcoding a Web site into VoiceXML will not be sufficient for a VoiceXML application. Typically there will be too much additional page data returned, or the data may not be in an ideal format for a VoiceXML application. In order to improve your VoiceXML application, it will be necessary to use the WebSphere Transcoding Publisher annotator to select and translate the sections of the Web page request that are appropriate for your VoiceXML application.

10

Voice Server language component

This chapter describes what the Voice Server language support component is, and where it is used. In addition to the base text-to-speech, it also describes IBM's concatenative text-to-speech components. Finally, it explains the considerations that must be addressed in different environments for a Voice Server implementation.

10.1 What is language support?

Voice Server's language support component provides speech recognition and text-to-speech operations for specific languages. For each language required for Voice Server, its corresponding language component needed to be installed. For example, to have both French and US English, the language component for each must be installed. In the case of US English and UK English, these are considered two different languages, and so their associated language support components must be installed for each.

In each language version installed, the component includes several functions.

10.1.1 IBM text-to-speech engine

The IBM text-to-speech engine allows text to be converted to synthesized speech. The text source may come from prompts within a VoiceXML application or from data that is retrieved from a database.This is known as TTS.

This TTS is referred to as formant TTS. This type of TTS system produces speech entirely through software using linguistic rules and models based on years of analyzing human speech. Speech output is generated completely algorithmically.

10.1.2 IBM speech recognition engine

The VoiceXML browser uses the IBM speech recognition engine to recognize voice input from a user responding to prompts generated by a VoiceXML application. The recognition results are used within the VoiceXML application to fill in form items or select among menu items. The speech recognition engine uses telephony acoustic models for accurate speech recognition over telephone lines.

The speech recognition engine process is as follows:

1. During application development, the developer creates a series of speech recognition grammars defining the words and phrases that can be spoken by the user, and specifying where each grammar should be active within the application.
2. When you start Voice Server, each VoiceXML browser starts an instance of the IBM speech recognition and text-to-speech engines.
3. When the application runs, the speech recognition engine processes the incoming audio signal and compares the sound patterns to the patterns of basic spoken sounds, trying to determine the most probable combination that represents the audio input.

4. The speech recognition engine compares the sounds to the list of words and phrases in the currently active grammar(s). Only words and phrases in the active grammars are considered as possible speech recognition candidates.
5. The speech recognition engine returns the results to the VoiceXML browser.
6. The VoiceXML browser uses the results within the VoiceXML application to fill in form fields or select menu items, thereby managing the window with the user.

10.1.3 Languages supported

Each version of Voice Server supports a slightly different set of languages. The choice of which version of Voice Server will be affected by the languages needed for your implementation.

In the Voice Server 2.0 for Cisco, languages supported are:

- French
- German
- Japanese[5]
- UK English
- US English

In the Voice Server 2.0 for Dialogic, languages supported are:

- French
- German
- Japanese[1]
- Simplified Chinese[6]
- UK English
- US English

In the Voice Server 3.1 for WebSphere Voice Response for Windows, languages supported are:

- Canadian French
- Brazilian Portuguese
- French
- German
- UK English
- US English

[5] Japanese language support can be implemented only on the Japanese-language version of Windows 2000.
[6] Simplified Chinese language support can be implemented only on the Chinese-language version of Windows 2000.

In the Voice Server 3.1 for WebSphere Voice Response for AIX, languages supported are:

- Brazilian Portuguese
- Canadian French
- French
- German
- Italian
- Japanese
- Simplified Chinese
- Spanish
- UK English
- US English

Note: In all environments, the Language Support component must be installed before the Voice Server component.

10.2 Language support implementation

Voice Server can be implemented in several ways, each being unique in the type of telephony environment that site has. This section discusses how it would be installed in each.

Important: You must install the Language Support component before installing any other Voice Server component.

Voice Server verifies which languages have been installed, so it will install the necessary server components. Once Voice Server has been installed, future languages cannot be installed to it.

10.2.1 Cisco environment

Voice Server for use with the Cisco telephony platform requires that language support be installed on every machine that will have voice server installed. If there are four machines in a Cisco distributed solution, each would need to have the language support installed prior to any Voice Server installation.

10.2.2 Dialogic environment

There are three general types of configuration possible for a Dialogic-based Voice Server system. This means you must first choose the configuration you want and then install the Voice Server components as described.

1. A stand-alone system. All Voice Server components are installed and reside on the Intel Dialogic platform, the server computer containing the Dialogic hardware and software.

   The Language Support component is installed on the stand-alone server.

2. A client/server system 1. Here the VoiceXML browsers are installed on one or more speech interface client computers but not on the server computer. For reasons of performance and scalability, this is the recommended configuration for Voice Server.

   The Language Support component is installed only on the speech interface client computers where the VoiceXML browsers are installed.

3. A client/server system 2. Here the VoiceXML browsers are installed on both the server and on one or more speech interface client computers.

   The Language Support component must reside on all computers housing the VoiceXML browsers.

10.3 Installing Voice Server Language Support component

The following describes how to install the language support component. The process is the same on all of the Windows-based platforms, but some of the windows may vary slightly, especially when installing a different language.

10.3.1 Installing process

In this example, we installed US English on a Dialogic platform. The installation process is basically the same for the different versions of WebSphere Voice Server, whether it is Version 2.0 or 3.1.

In our case we had executable files that allowed us to install the languages. However, the Language Support component will usually be provided on a CD.

Step 1: Installing

Either run the application or insert the CD. Click **Next**.

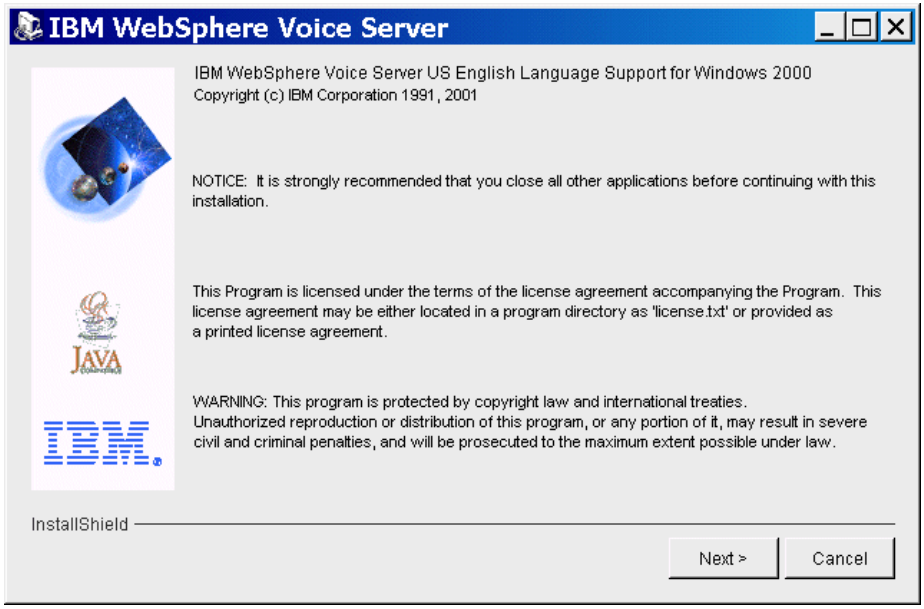


Figure 10-1 Installing Voice Server language support

Step 2: Licensing agreement

The license agreement appears. You must read and agree to this to proceed. Click **Next**.

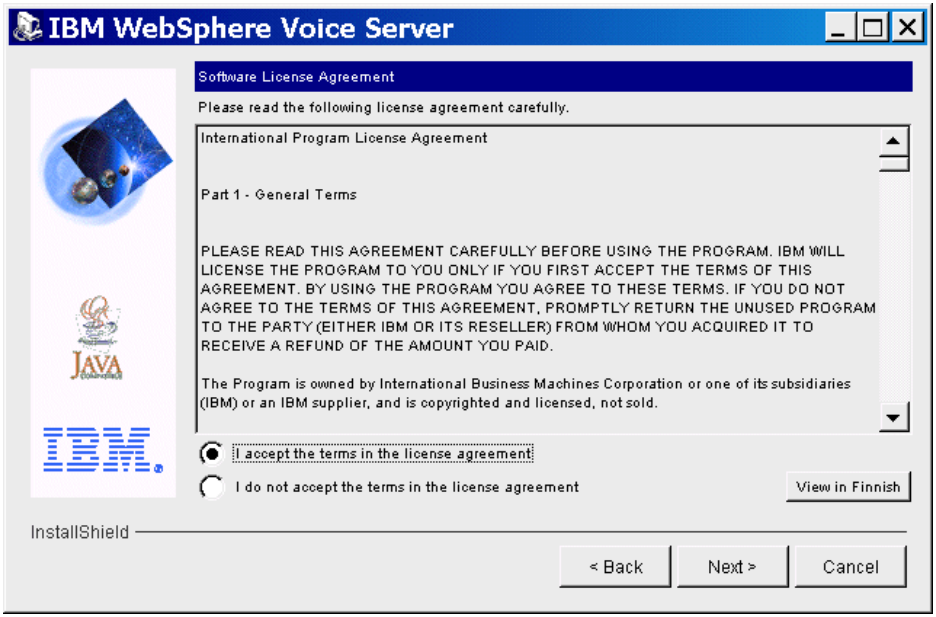


Figure 10-2 Licensing agreement

Step 3: README update

The README file (Figure 10-3 on page 460) contains the latest install information. Read this before moving on. Click **Next**.

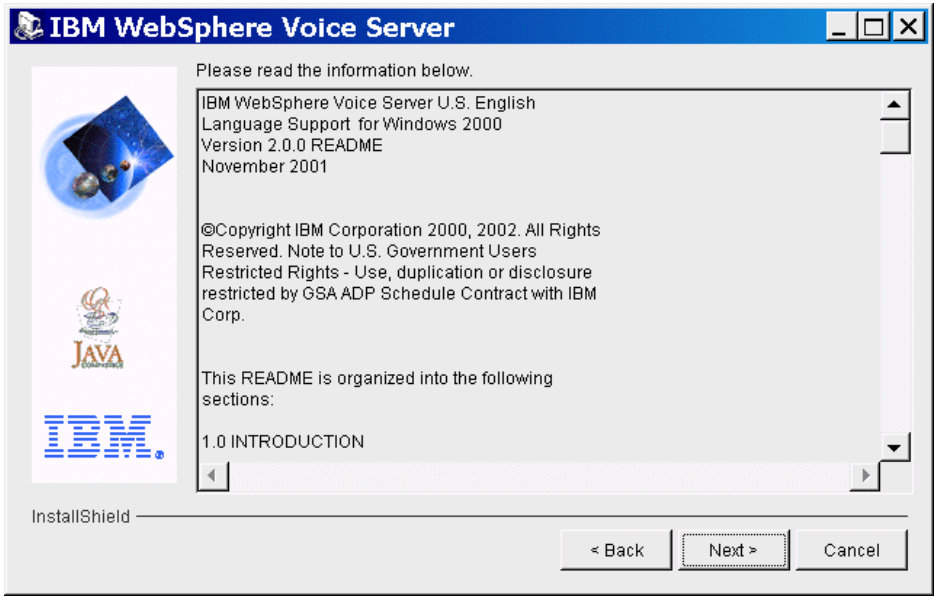


Figure 10-3 README window

Step 4: Language confirmation

Before installing, the language selected is displayed. Figure 10-4 on page 461 shows that we selected and installed US English. Click **Next** to install.

Figure 10-4 Language confirmation

5. Step 5: Completion

Once completed, click **Next**. You are now ready to install the Voice Server component.

Figure 10-5 Completion window

Step 6: Other languages

Repeat the installation process for the other languages you require.

When Voice Server is installed, a Supported languages window appears. It displays the languages its particular version supports, as well as the installed languages. Figure 10-6 on page 463 shows the Cisco environment window, Figure 10-7 on page 463 shows the Dialogic environment window, and Figure 10-8 on page 464 shows the WebSphere Voice Response for Windows enivornment.

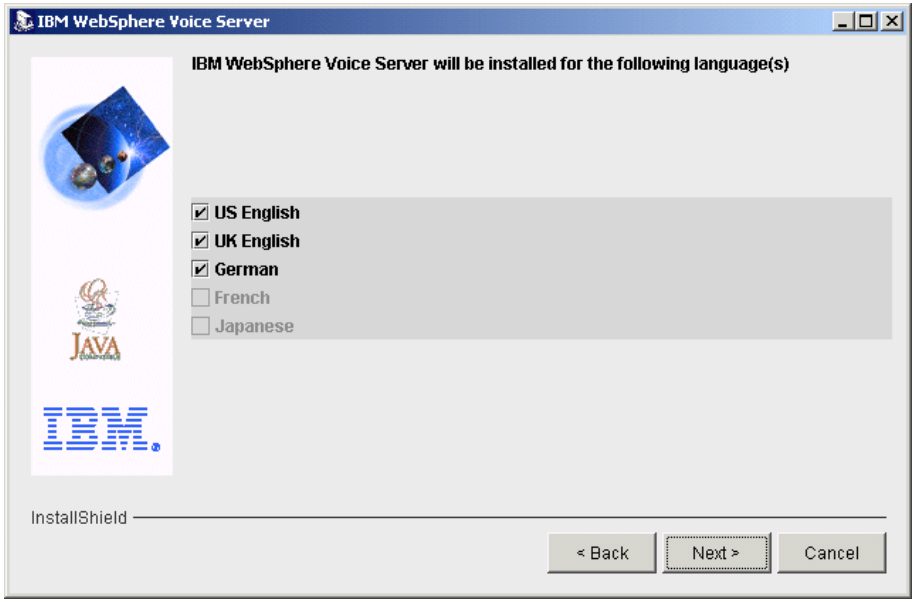


Figure 10-6 Cisco installed languages window

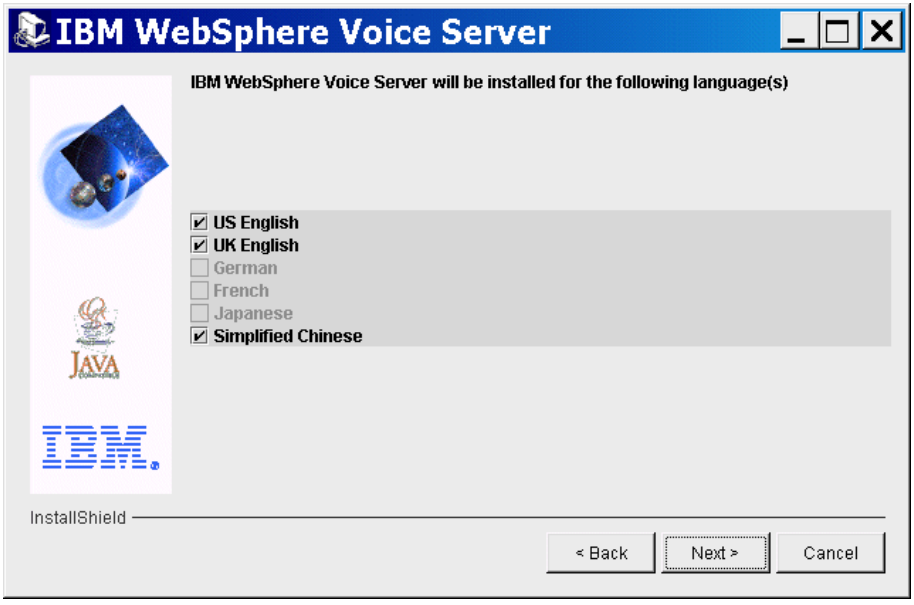


Figure 10-7 Dialogic installed languages window

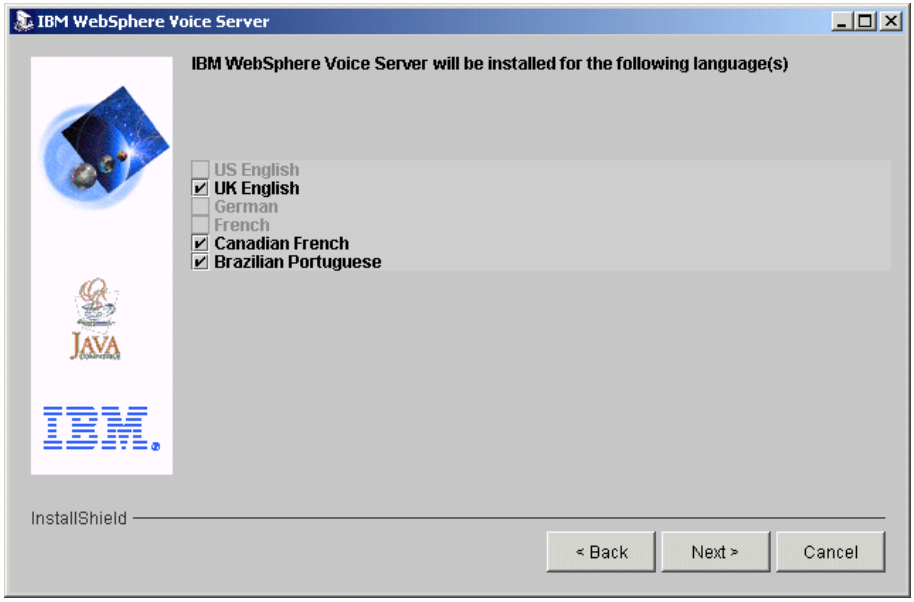


Figure 10-8 WebSphere Voice Response for Windows installed languages window

10.4 Different text-to-speech versions

So far we have spoken about the basic level of text-to-speech that is part of the language support installation file. In this section, we will discuss the different versions of text-to-speech and how development created the different versions so that the playback function sounds more natural to the listener.

The different versions of TTS can be split into three levels. Each usually requires additional software to be installed, but more importantly, more system resources are needed. The advantage is that the TTS sounds more natural to the listener.

Formant TTS: This type of TTS system produces speech entirely through software using linguistic rules and models based on years of analyzing human speech. Speech output is generated completely algorithmically, instead of by concatenating waveforms.

Concatenative TTS (CTTS): This type of TTS produces speech from recordings of units of actual human speech. These units (phonemes, syllables) are then combined (concatenated) according to linguistic rules formulated from analyzed text. The output will be human-sounding speech, but would contain transitions created due to the synthesis. Compared to formant TTS, this type of TTS requires large amount of memory and storage (about 100 to 200 MB).

Phrase Splicing TTS (PSTTS): This type of TTS produces speech from recordings of units of actual human speech. These units (words, or phrases) are then combined (concatenated) according to linguistic rules formulated from analyzed text. The output can be very natural, human-sounding speech. Compared to concatenative TTS, phrase splicing requires an enormous amount of memory and storage for storing the pre-recorded splices (about 500-700 MB); thus, applications that use this type of TTS are usually limited to one voice and to a very specific application domain, for example weather information, automotive directions/navigation, etc. Phrase splicing will fall back to concatenative when a specific word or phrase is not found in the prerecorded splices.

10.5 Concatenative TTS language

IBM has CTTS in several different languages. As with the basic language support, the language set differs slightly for each platform.

For Voice Server 2.0 for Cisco, CTTS languages supported are:

- French
- German
- Japanese[7]
- UK English
- US English

For Voice Server 2.0 for Dialogic, CTTS languages supported are:

- French
- German
- Japanese[1]
- Simplified Chinese[8]
- UK English
- US English

For Voice Server 3.1 for WebSphere Voice Response for Windows, CTTS languages supported are:

- Canadian French
- Brazilian Portuguese
- French
- German
- UK English
- US English

[7] Japanese language support can be implemented only on the Japanese-language version of Windows 2000.

[8] Simplified Chinese language support can be implemented only on the Chinese-language version of Windows 2000.

For Voice Server 3.1 for WebSphere Voice Response for AIX, CTTS languages supported are:

- Brazilian Portuguese
- Canadian French
- French
- German
- Italian
- Japanese
- Simplified Chinese
- Spanish
- UK English
- US English

10.5.1 Installing Voice Server concatenative language component

The CTTS language can only be installed after two prerequisites are met. First, the base language of the CTTS code needs to be present. Secondly, the WebSphere Voice Server code must also be present. Only then will the CTTS application allow itself to be installed. For example, to install US CTTS English, the corresponding US base language and the WebSphere Voice Server code for that particular platform application need to be present.

For multiple language systems requiring CTTS, each language will need to have its own CTTS installed. Otherwise, WebSphere Voice Server will use the formant TTS.

The following describes how to install the CTTS language support component. The process is the same on all of the Windows-based platforms, but some of the windows may vary slightly, especially when installing a different language.

10.5.2 Installing process

In this example, we installed UK CTTS English on a Voice Server for WebSphere Voice Response on Windows 3.1. The installation process is basically the same for the different versions of WebSphere Voice Server, whether it is Version 2.0 or 3.1.

In our case we had executable files that allowed us to install the CTTS language.

Step 1: Installing

Either run the application or insert the CD. The installer window will be presented, as shown in Figure 10-9 on page 467. Click **Next**.

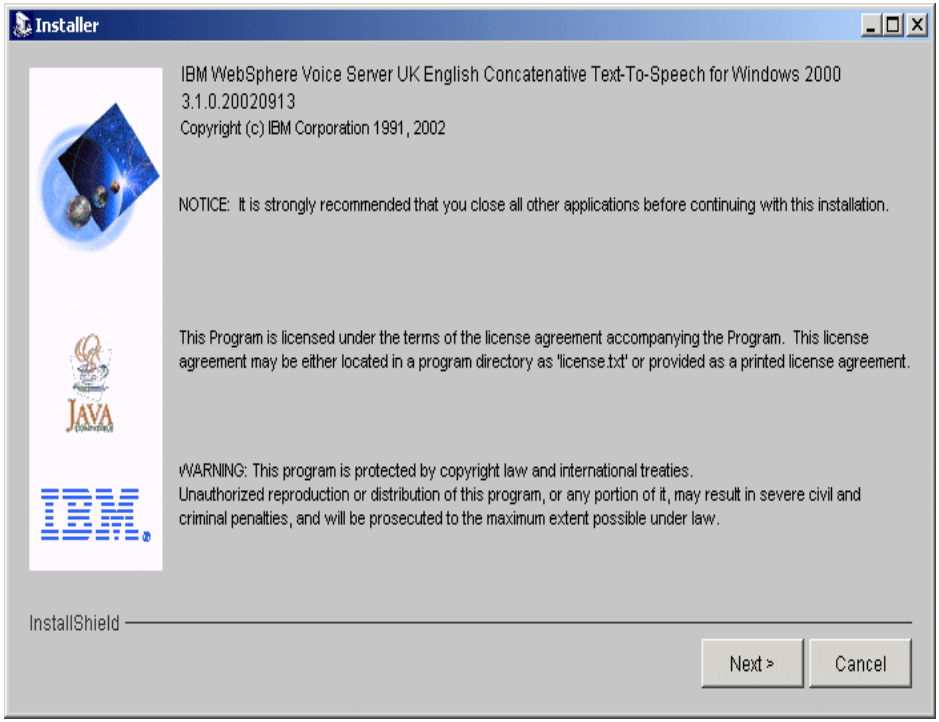


Figure 10-9 CTTS installer

Step 2: Licensing agreement

The license agreement appears. You must read and agree to this to proceed. Click **Next**.

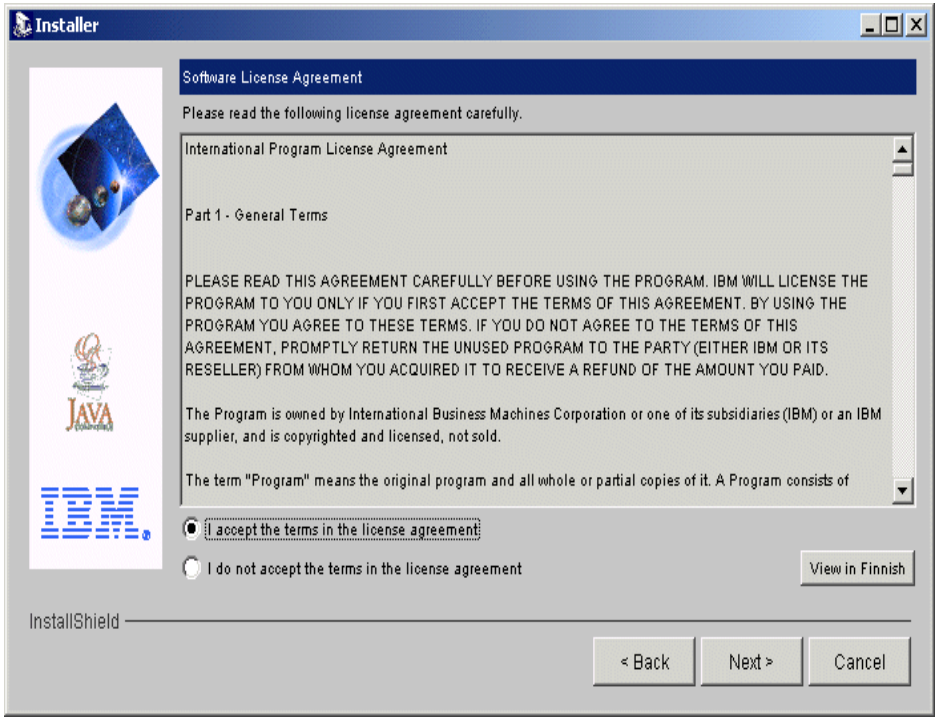


Figure 10-10 License agreement

Step 3: Prerequisite check

The CTTS installer will test for the necessary prerequisites before it will continue. If one does not pass, the installation will not continue. Correct the problem and try again. If all the prerequisites are in place, click **Next,** as shown in Figure 10-11 on page 469.

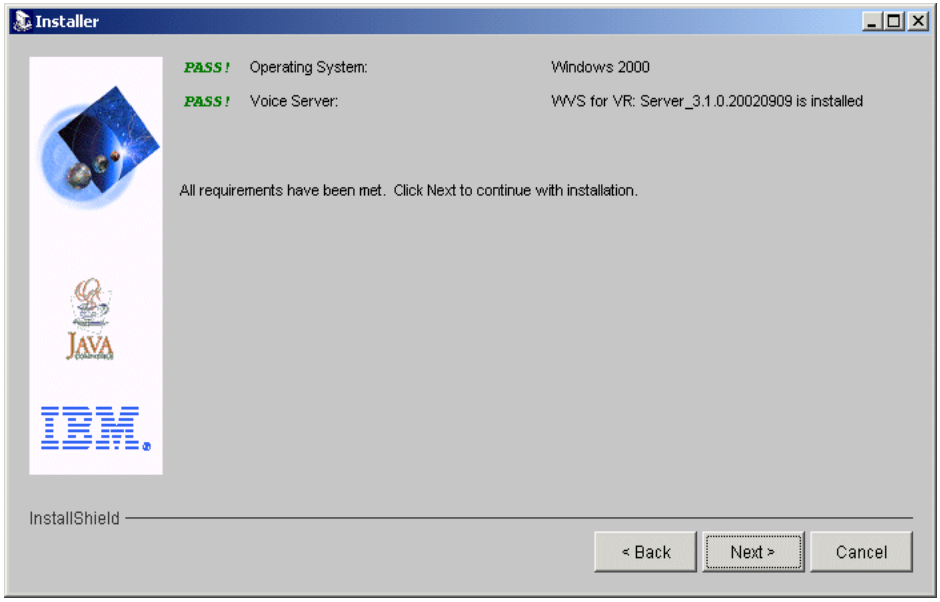


Figure 10-11 Prerequisite check

Step 4: Code installation

The CTTS installer program will display a final confirmation window, as shown in Figure 10-12 on page 470 . Click **Next** to complete the installation.

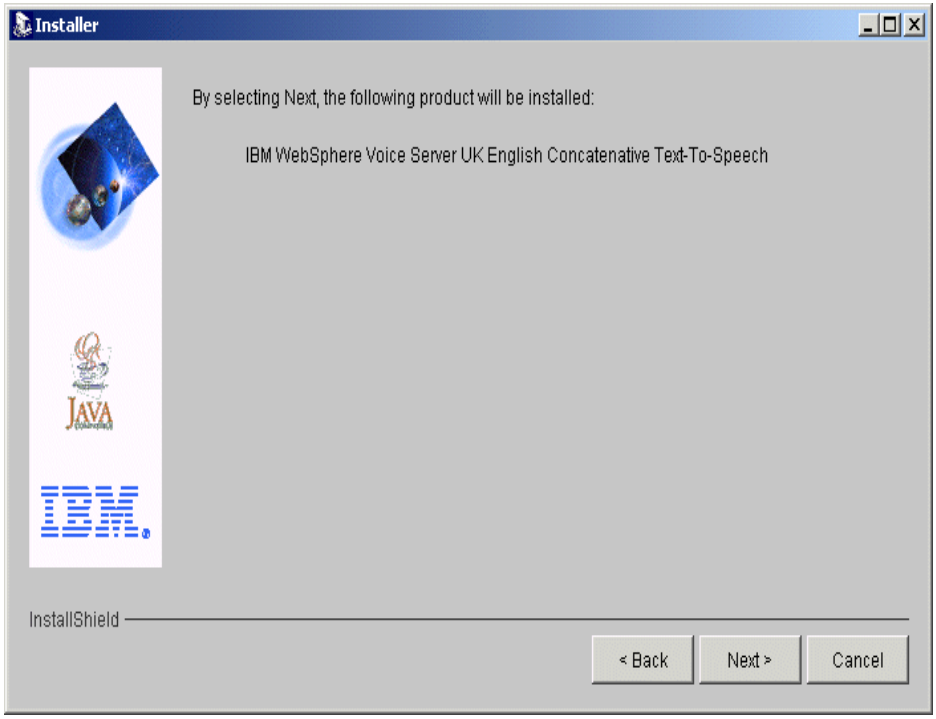


Figure 10-12 Install confirmation

Step 5: Completion

Figure 10-13 on page 471 shows the completion of the UK CTTS language installation. Click **Finish.** At this point you should reboot your machine before continuing.

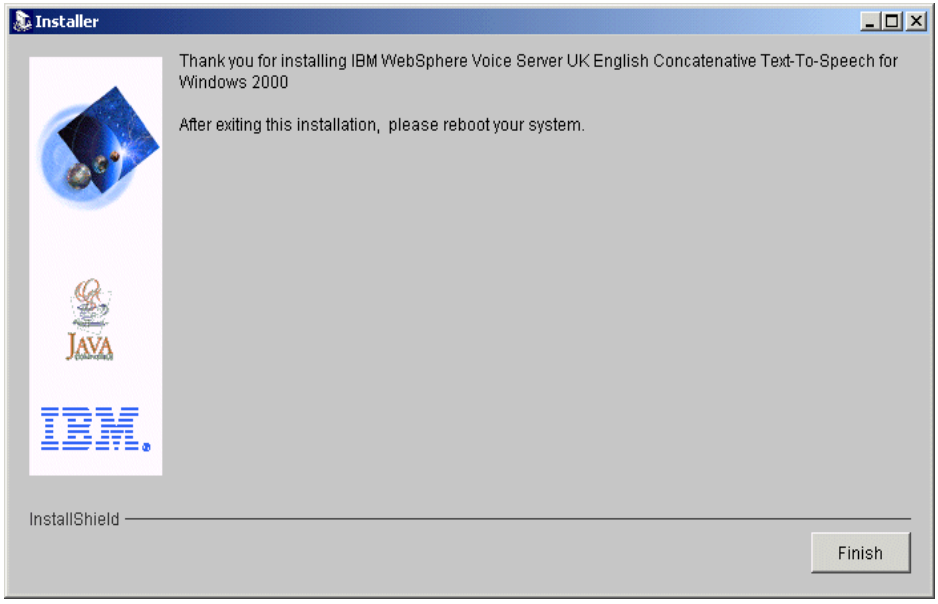


Figure 10-13 Completion and reboot message

Step 6: Testing

There are no parameters that need to be set or modified to enable the use of CTTS. Whenever WebSphere Voice Server has an application that requires the text-to-speech function, it will invoke the CTTS version. An example would be any application that uses the prompt text field. Try it and hear the difference.

The system resources required by CTTS languages are much greater than those of TTS. Table 10-1 displays the memory requirements for each individual CTTS language installed. This information will assist in determining the WebSphere Voice Server machines requirements. This information is based on the CTTS languages for WebSphere Voice Server fow WebSphere Voice Response on Windows 3.1. In other versions it may vary slightly.

Table 10-1 CTTS memory requirements

	Hard drive space	RAM male voice	RAM female voice	Each CTTS channel
Each CTTS	450MB	70MB	145MB	8MB

10.6 Phrase splicing

Phrase splicing is the next step up from CTTS. The TTS uses actual recordings of human speech and then combines these according to linguistic rules. The rules are formulated from analyzed text.

The output sounds very natural, because it utilizes these actual human recordings. However, there is a requirement for large amount of storage space, and also an enormous amount of memory. This is used by the CTTS engine to produce the spoken words.

Phrase splicing is an offering from IBM, rather than an individual product that is simply installed onto an existing WebSphere Voice Server solution. IBM will work with a customer to designed a phrase splicing system specific to that environment. An example maybe a navigation system or information line. The benefit here is that it enhances the end user's experience.

To learn more about this offering, please speak to your IBM customer representative.

11

WebSphere Voice Server hardware environments

This chapter describes the different hardware environments WebSphere Voice Server was installed and test in. Tested environments included stand-alone systems to distributed ones. The machines used varied from Intel-based desktops and servers to AIX systems. The purpose was to show how to create a WebSphere Voice Server solution in a real production environment.

11.1 WebSphere Voice Server in an IBM @server BladeCenter distribution

To reduce the workload on the telephony server, a distributed system is recommended. We tested a fully distributed configuration, meaning one server was used for the telephony environment, and then x number of servers were used to host the WebSphere Voice Server.

Our WebSphere Voice Server implementation used WebSphere Voice Server for Dialogic. This meant one server contained the Dialogic telephony card and the necessary WebSphere Voice Server server component. The other distributed machines had the WebSphere Voice Server client software installed. The number of clients can vary, but all have one thing in common: they communicate with the telephony server to indicate they can accept and process inbound calls.

11.1.1 The IBM @server BladeCenter

The distributed system used the IBM @server BladeCenter server solution. The ITSO lab had the 8677-1XX BladeCenter system unit available for testing. This server can accommodate up to 14 separate individual PCs within its case. Each server has its own processor, memory, hard drive and network card. With the IBM @server BladeCenter the physical space taken up by 14 machines is drastically reduced, saving on environmental considerations and power consumption.

The ITSO @server BladeCenter had five 8678-21X HS20 servers. Each server had the following specifications:

- One Intel Xeon processor
- 512 MB RAM
- 40 GB IDE hard drive
- Windows 2000 Advanced Server with Service Pack 3

Figure 11-1 on page 475 shows the actual IBM @server BladeCenter machine we used. The five servers are located on the left-hand side of the unit.

Note: IBM @server BladeCenter cannot accommodate any of the full-length Dialogic telephony PCI cards.

Figure 11-1 BladeCenter with five HS20 servers

11.1.2 Server

The telephony server used was an IBM xSeries 342 machine. We had it configured with one Dialogic D/120JCT-LS 12 port analog card. The server specifications were:

- One Intel Pentium III 933 processor
- 1 GB RAM
- 2 x18 GB SCSI hard drives

Note: The IBM @server BladeCenter unit cannot be used as the telephony server, since it cannot accommodate full-length PCI cards. This means all of the Dialogic cards will not fit into the server chassis.

11.1.3 Software

In this section, we discuss the software used for our configuration.

Server

The server had WebSphere Voice Server 2.0 for Dialogic installed. We selected the server option and no client software. All TTS and speech recognition engines were run from the clients.

Clients

For the five clients, each had a language installed on it. We had the following languages configured:

- Server 1 - US English
- Server 2 - UK English
- Server 3 - French.
- Server 4 - German.
- Server 5 - UK English and German.

For each client, the base language must be installed before the WebSphere Voice Server client software is. If future languages are needed on a client, WebSphere Voice Server must be uninstalled first.

The most important parameter that must be set on all clients is the CALLMANAGER_HOST. This is located in the sysmgmt.properties file. The WebSphere Voice Server client must have the WebSphere Voice Server telephony server IP address defined here. Failure to do this will mean the client never communicates with the WebSphere Voice Server server.

There are several benefits with this configuration that a customer can take advantage of:

- The WebSphere Voice Server telephony server can have its system resources clearly identified, that is, the number of channels will be defined by the type of cards used.
- The WebSphere Voice Server function is offloaded to client machines allowing the WebSphere Voice Server telephony server to handle just the telephone interface.
- If more WebSphere Voice Server clients are needed, they can be easily added into the environment without disrupting the WebSphere Voice Server telephony server.
- Additional languages can be quickly introduced.

We successfully tested the last benefit by adding a Chinese language client.

Note: In this configuration, WebSphere Voice Server uses a round-robin system for inbound calls. This means all voice browsers on each machine accept one call before the process is repeated again.

DNIS or caller ID could potentially be used to direct inbound calls to specific WebSphere Voice Server clients. This function was not available for use to test in the ITSO lab.

11.2 WebSphere Voice Server in a Voice over IP environment

WebSphere Voice Server for Cisco utilizes the VoIP protocol standard. Normally the WebSphere Voice Server server would be configured to work with a Cisco 2600 router that has a telephony interface connection. When a phone call is made, the 2600 will convert the call to VoIP and then redirect the voice packets to the WebSphere Voice Server server.

We decided to test this implementation in a different way. Rather than use a phone on a desk or a mobile phone, we looked at how a system could be used in a mobile environment, where the user has a Personal Digital Assistant(PDA).

A PDA can be used for keeping inventory, spreadsheets, e-mail and documents, as well as for connecting to wireless networks. Add to its phone capability, the PDA becomes a real portable productivity tool.

In a real-life solution, the user may be in a warehouse where there is a wireless network. They are mobile people, moving around, taking notes, filling in spreadsheets, working with inventory, and making calls, using the same wireless LAN they use to write e-mail.

11.2.1 Hardware environment

In our environment, we used a Windows CE device called a Compact I-paq. The actual model was a 3850 with Windows CE 3.0. Attached to this was a PC card sleeve. This allowed us to use a wireless LAN card for connectivity.

The WebSphere Voice Server server was installed on an IBM T21 ThinkPad laptop. We used UK English with the addition of CTTS language support.

Normally a wireless LAN would have multiple access points for the mobile user to gain access, perhaps also running DHCP. In our test, we connected the laptop

to the PDA directly, using static IP address. The settings listed in Figure 11-1 are what we used to connect the two devices together.

Table 11-1 Wireless network settings

Machine	Operating System	IP address	SSID	Encryption	Operating mode	Channel
PDA	Windows CE	111.111.111.11	voice	disabled	peer-to-peer	1
WebSphere Voice Server server	Windows 2000	111.111.111.112	voice	disabled	ad-hoc	1

The wireless LAN must be configured correctly for each device or it will not work. Ensure the right drivers and settings are used. Before installing anything further, test that there is connectivity. A simple but effective test is pinging each other.

11.2.2 Software environment

Once the hardware is configured, the software can be installed.

Laptop - Server

The laptop server had WebSphere Voice Server for Cisco 2.0 installed. We used UK English language support. In addition we installed CTTS UK English to the system.

PDA - Client

The PDA required a H.323 compliant application to communicate with WebSphere Voice Server. There are several freely available on the Internet. We used a number of these to test WebSphere Voice Server over the wireless LAN. Figure 11-2 on page 479 shows the H.323 application running on the PDA.

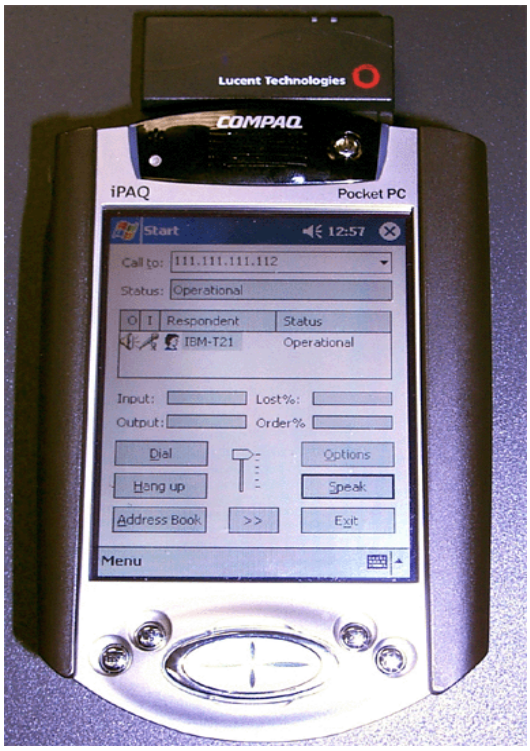


Figure 11-2 H.323 compliant application running on PDA

Note: To reduce the effect of background noise, we used PDA H.323 applications that had a mute function.

11.3 WebSphere Voice Server for WebSphere Voice Response on Windows Version 3.1

For WebSphere Voice Server on WebSphere Voice Response for Windows 3.1, we used several machines for testing. Our main test system was an IBM Netfinity 5100 Server, 8658-51Y, pictured in Figure 11-3 on page 480. Its hardware specifications were:

- One Intel Pentium III 933 processor
- 1 GB of RAM
- Two 18 GB SCSI hard drives

Figure 11-3 5100 Netfinity server

This system was tested with multiple Dialogic cards configurations, both in a stand-alone server and a distributed environment. For the WebSphere Voice Response server configuration, we used:

- Analog environment:
  - Single D/41JCT-LS
  - Single D/120JCT-LS
  - D/41JCT-LS and D/120JCT-LS
  - D/41JCT-LS, D/120JCT-LS and D/320JCTU
- Digital T1 environment:
  - D/240JCT-LS
  - D/480JCT-LS
  - D/240JCT-LS and D/480JCT-LS
  - D/240JCT-LS and D/320JCTU

Our second test machine was an IBM Netfinity 5500, 8660-4RU, pictured in Figure 11-4 on page 481. The hardware specifications were:

- Pentium II 400 processor
- 1 GB of RAM
- Six 9.1GB SCSI hard drives in RAID 5 configuration

This server was used to test WebSphere Voice Server for WebSphere Voice Response on Windows 3.1 in an analog environment. We tested multiple card support and echo cancellation functions. In addition the server was also used for the Voice Server SDK and Voice Toolkit.

Figure 11-4 NF550 server

11.4 WebSphere Voice Server on WebSphere Voice Response on AIX 3.1

For the AIX environment, we use an IBM RS/6000 server. This was a 44P Model 170 (7044-170). The specifications were:

- One CPU POWER3-II 333 MHz
- 512 GB of RAM

This server was used to install WebSphere Voice Server on WebSphere Voice Response for AIX 3.1. We used this in several configurations:

- Stand-alone server
- Distributed system.
- Distributed system with non-English WebSphere Voice Server client machines on AIX.
- Distributed system with WebSphere Voice Server clients running on Intel PCs, with Windows 2000 Server as their operating system.

Figure 11-5 RS/6000 44P Model 170

Note: The mixed operating system configuration is not officially supported. However, we successfully configured a working environment. Refer to Chapter 4, “WebSphere Voice Server with WebSphere Voice Response for AIX V3.1” on page 123 for more details.

12

WebSphere Portal Technology for Voice

This chapter introduces the WebSphere Portal Technology for Voice, and discusses how it enhances WebSphere Portal availability to the end user. Rather than reach information through a traditional browser on a PC, users can access it using a normal telephone as the interface.

Note: Excerpts for this chapter were extracted from the *IBM WebSphere Portal Technology for Voice* document and were used to set up our Portal for Voice environment. The complete document is available to you when you unpack the Portal for Voice wpt4va02.exe file. Then go the following directory to access:

c:\ destination folder docs\en\infocenter\wpt4va.pdf

Where destination folder would be, for example, voiceportal.

12.1 WebSphere Portal Technology for Voice overview

WebSphere Portal Technology for Voice enables mobile employees to reach information they need, by use of a phone. Think of a field sales professional getting remote access to Personal Information Management (PIM) data, information about customers, inventory, and delivery times. Employees are able to confirm delivery times, at any time after office hours, and then check their own calendars to make follow-up appointments. This can be achieved using Portal for Voice.

Portal for Voice enables you to listen to the content of a portal by giving voice commands. The portal content can be company application information provided by portlets that are voice-enabled, or PIM data provided by Voice for Note portlet, which is included with this product.

12.1.1 Portal content

Portal for Voice enables you to listen to the content of a portal by giving voice commands.

The portal content can be company application information provided by portlets that is voice-enabled, or PIM data provided by a Voice for Notes portlet, which is included in this product. You can get access to your portlet by both PC browser and telephone.

For information on voice-enabled portlets, see 12.2, “Developing applications” on page 485.

12.1.2 Component connections

Portal for Voice is a solution that consists of components that enable you to get access to your portal and to the Voice for Notes portlet, by giving voice commands.

With your telephone, you will dial the number given to you by your administrator and connect to the WebSphere Voice Server. You may be asked for your ID and password, then be connected to your portal, where the Voice Aggregator will navigate you and dispatch an appropriate voice-enabled portlet. The voice-enabled portlet can be the one you worked with on your PC at your office. Your portal may be configured by your administrator, and you can also customize it by yourself by using the browser on your personal computer.

When you get access to your Notes mail, calendar, or address book, the Notes portlet will get access to the Domino Server through Notes Client installed in the Portal for Voice.

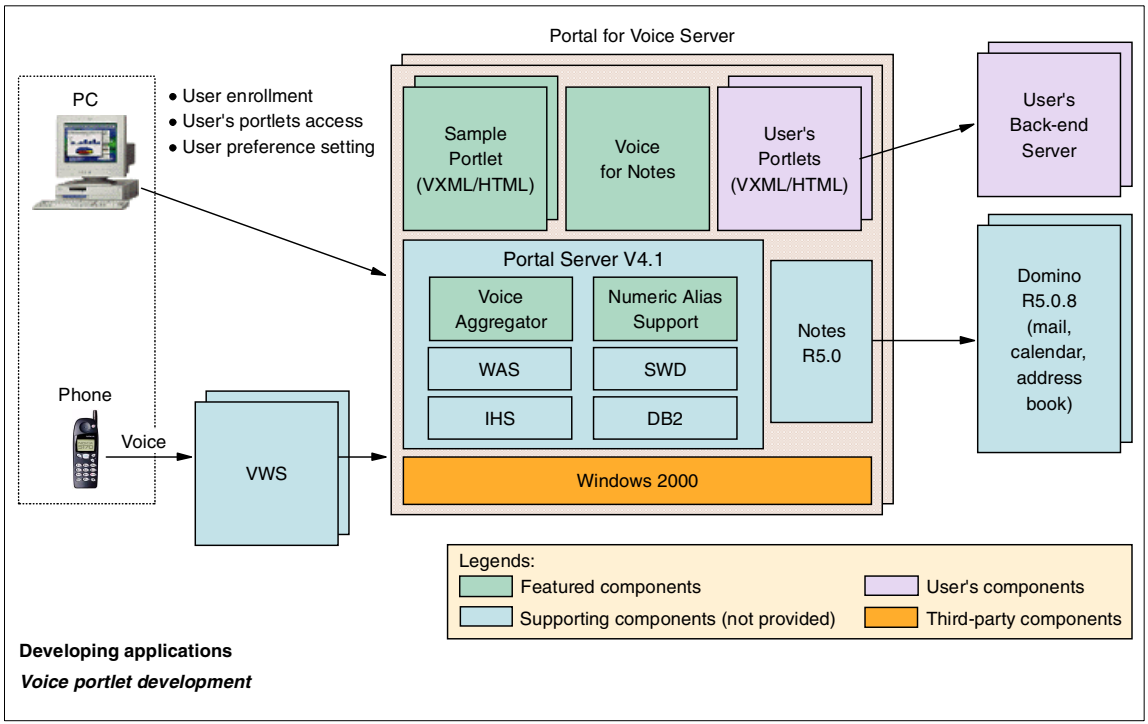


Figure 12-1 WebSphere Portal Technology for Voice environment

12.2 Developing applications

In this section, we discuss the development of portlet applications.

12.2.1 Voice portlet development

A portlet is a special type of servlet that allow you to easily plug in and run the portlet on the portal server. The portlet container relies on the J2EE architecture implemented by WebSphere Application Server, and the process for voice portlet development is similar to J2EE Web applications and are deployed like servlets.

12.2.2 Setting up an environment for portlet development

You need to set up an environment in which to write, compile, and test voice portlets. For this purpose, the machine you use needs to have the following software components:

- WebSphere Portal Technology for Voice

- WebSphere Portal Offering 4.1: Application Server 4.0 and DB/2 UDB 7.2 are included in this package.
- A Java development environment: for example, IBM WebSphere Studio Application Developer 4.0.
- An editor for the VoiceXML and grammar files: for example, IBM WebSphere Voice Toolkit 2.0.
- Everyplace Toolkit 4.1.2: this provides tools for developing portlets.
- WebSphere Voice Server Software Developers Kit (SDK): this provides tools for testing VoiceXML applications on a PC.

12.2.3 Generating markup using JSPs

For the typical voice portlet, you need to create JSPs to generate VoiceXML. You can separate the portlet markup from the main functionality of the portlet by using JSPs and the taglib function of JSPs.

12.2.4 Creating deployment descriptors

The WAR file for a portlet application must contain two descriptor documents:

- Web application deployment descriptor, web.xml
- Portlet deployment descriptor, portlet.xml

12.2.5 Packaging and deploying a voice portlet

Package your portlet application as a WAR file. If you are using IBM WebSphere Studio Application Developer, IBM WebSphere Studio Site Developer, or IBM WebSphere Voice Toolkit, you can use the export function of the tool to create a WAR file.

After you package your voice portlet as a WAR file, you need to install your Portal for Voice application on the portal server. To do this, you use the Portal administration function.

12.2.6 Using the Everyplace Toolkit

The Everyplace Toolkit is an IBM environment for developing portlets. Version 4.1.2 includes a sample voice portlet in addition to the GUI sample portlets. This toolkit improves the productivity of portlet development. *Developing a Voice Portlet* describes in detail the steps to develop a voice portlet by using this Toolkit. This document is found in the Voice for Portal package, in the \docs\en\Toolkit directory.

Installing Voice Aggregator on WebSphere Portal

If you remotely debug a voice portlet by using Everyplace Toolkit, you need to install Voice Aggregator on WebSphere Portal which is installed on WebSphere Application Server Advanced Single Server Edition. To install the Voice Aggregator, take the following steps:

1. Copy the following ZIP file in Portal for Voice package to your local directory:

   \install\en\voicebasic.zip

2. Create the voice directory as follows:

   <WebSphere>\PortalServer\install\voice

3. Extract the ZIP file into the voice directory.
4. Stop WebSphere Application Server.
5. Extract app.zip file in the voice directory to the following directory:

   <WebSphere>\PortalServer\app

6. Extract appserver.zip file in the voice directory to the following directory:

   <WebSphere>\AppServer

7. Start WebSphere Application Server (using portal config file) and ensure WebSphere Portal is started.
8. Run updatePortal.bat as follows:

   ```
   updatePortal hostname
   ```

   Where `hostname` is the host name of the WebSphere Portal machine.

9. Stop WebSphere Application Server.

Note that the Portal for Voice installer does not support the installation of Voice Aggregator on WebSphere Portal, which is installed on WebSphere Application Server Advanced Single Server Edition. To check if the Voice Aggregator is correctly installed and configured, take the steps described in 12.13, “Verifying installation” on page 499.

12.3 Components

In this section, we discuss the components for the Portal for Voice solution.

12.3.1 Overview

This section provides a detailed description of the components in this package. There are three types of components:

- Featured components: the primary services and components that make up the Portal for Voice solution.
- Supporting components: components that provide underlying support to the featured components. These components are not included in this package and must be purchased separately.
- Third-party components: components made up of third-party software packages that are required to support some featured components. These components are not included in the package, but must be purchased separately.

12.3.2 Featured components

The following components are featured in this solution:

Voice for Notes

Voice for Notes enables access to a Lotus Notes mail database with voice. Users can listen to, reply to, or forward Notes mail with voice. Users can utilize contacts in their personal Notes address book for filtering mail to listen to or specifying a target address to forward to.

Users can also have access to personal Notes calendar to check or add their calendar entries.

For the purposes of this chapter, we have not installed/tested the Voice for Notes. If you would like more information on installing this component, please refer to the following documentation:

- InfoCenter: /docs/language/InfoCenter/index.html
- README: /readme_language.html

Voice Aggregator

Voice Aggregator is an additional support package for Portal Server to support voice markup called VoiceXML, and to enable VoiceXML rendering for portlets. The Voice Aggregator handles user authentication by use of VoiceXML and activation/deactivation of portlets that users access to with voice. Switching of active portlet can be done with voice by saying the name of the portlet.

Numeric Alias Support

Because a normal phone can only enter numerical values, a voice user needs a second user ID and password that is all numeric. The Numeric Alias Support component adds a pair of numeric user ID and password attributes to Portal Server users so that the Portal Server can accept a numerical user ID and password for voice user authentication. This component is not necessary when you allow users to enter numerical values only for the original user ID and password of Portal Server and use them for voice user authentication.

12.3.3 Supporting components

The following are support components required for the feature components to work. These components are not included in the Portal for Voice packaging.

WebSphere Portal for Multiplatforms Version 4.1

WebSphere Portal enables a company to build its own custom portal Web site to serve the needs of employees, business partners, and customers. Users can sign on to the portal and receive personalized Web pages providing access to the information, people, and applications they need. This personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases the use of the Web site.

This release of Portal for Voice requires the WebSphere Portal Enable Version 4.1.2 offering, which is called WebSphere Portal or Portal Server for short in this document.

DB2 Universal Database Version 7.2 Fixpack 5

DB2 Universal Database is a Web-enabled relational database management system. It supports many levels of complexity in database environments. DB2 is used by WebSphere Portal and stores portal-specific data.

IBM HTTP Server Version 1.3.19.1

An IBM enhanced Web server based on the Apache Web server. IBM HTTP Server supports both the Secure Sockets Layer (SSL) Version 2 and SSL Version 3 protocols for secure connections. It also includes a cache accelerator for improved performance when serving static Web pages.

SecureWay Directory Version 3.2.2

This is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon. It is based on a client/server model that provides client access to an LDAP server. SecureWay Directory provides an easy way to maintain directory information in a central location for storage, updating, retrieval, and exchange. WebSphere Portal uses LDAP to store user-specific information.

WebSphere Application Server Advanced Edition Version 4.0.2
Enables Web transactions and interactions with a robust deployment environment for e-business applications. It provides a portable, Java-based Web application deployment platform focused on supporting and executing servlets, JavaBeans, JavaServer Pages (JSP) files, and enterprise beans.

Lotus Domino Version 5.08 or higher

Domino offers the industry's most comprehensive support for Internet messaging standards, with Internet addressing, SMTP routing and MIME content support all native. It also provides full support for E/SMTP, S/MIME, SSL, POP3, IMAP4, LDAP, HTTP, HTML, SNMP, etc. Domino supports WebSphere Portal portlets. Voice for Notes uses Lotus Notes client as a library to get access to Lotus Domino Server.

Lotus Notes Version 5.08 or higher

Lotus Notes client is used as a library by Voice for Notes to get access to Lotus Domino Server.

WebSphere Voice Server

The IBM WebSphere Voice Server enables developers to quickly develop and deploy voice-enabled e-business solutions. It enables companies to utilize their existing Web infrastructures for the delivery of voice-enabled Internet applications to wireline and wireless devices. It does all this by applying industry-standard technology such as VoiceXML and Java. WebSphere Voice Server has four main parts:

- Speech recognition engine recognizes callers' speech and converts into text.
- Text-to-speech engine receives text from an application, and converts it into a speech audio stream for playback over a telephone.
- Voice application development tools used to develop and test speech applications using speech recognition and/or text-to-speech.
- Telephony platform connector runtime platform for speech applications that connects the voice audio streams from the telephony network to engines for speech.

12.4 Third-party components

In this section, we discuss the third-party components.

12.4.1 Required

The following third-party components are required, depending on your implementation, to enable some functions. These components are not provided.

- Operating system:
  - Windows 2000 Server with Service Pack 2
  - Windows 2000 Advanced Server with Service Pack 2
- Web browser support:
  - Microsoft Internet Explorer Versions 5.0, 5.5, and 6.0
  - Mozilla 5.0
  - Netscape Versions 6.1 and 6.2
  - Opera 5.0

  Other browsers may be supported. For more information, refer to the README file.
- LDAP:
  - SecureWay Directory Version 3.2.2
  - iPlanet Directory Server Version 5.0
  - Lotus Domino Enterprise Server 5.0.5
  - Microsoft Active Directory 2000

Note: If you install the Numeric Alias Support component, only SecureWay Directory is supported.

12.5 CD contents

In this section, we list the contents of the CD.

12.5.1 Portal for Voice information

Documentation for Portal for Voice is provided in the following files:

InfoCenter: /docs/language/InfoCenter/index.html

README: /readme_language.html

12.5.2 Contents

- Portal for Voice Installer
- Numeric Alias Support
- Voice for Notes
- Voice Aggregator and a sample voice portlet
- InfoCenter and any other documents

12.6 Additional information

Web site and support site

Search for WebSphere Portal Technology for Voice at IBM alphaWorks site (http://www.alphaworks.ibm.com) to find the information you need.

IBM Everyplace

Everyplace Toolkit is available at:

http://www-3.ibm.com/pvc/products/mobile_apps/everyplace_toolkit.shtml

IBM HTTP Server

Web site: http://www.ibm.com/software/webservers/httpservers/

Web library:
http://www.ibm.com/software/webservers/httpservers/library.html

Lotus Domino

Web site:
http://www.lotus.com/home.nsf/welcome/dominoapplicationserver

SecureWay Directory

Web site: http://www.ibm.com/software/network/directory/

Web library: http://www.ibm.com/software/network/directory/library/

WebSphere Application Server

Web site: http://www.ibm.com/software/webservers/appserv/

Web library site:
http://www.ibm.com/software/webservers/appserv/library.html

WebSphere Personalization

Web site: http://www.ibm.com/software/webservers/ personalization/

WebSphere Portal

Web site: http://www.ibm.com/software/webservers/portal/

Web library site:
http://www.ibm.com/software/webservers/portal/library.html

Redbooks

The following Redbooks are available from the IBM Redbook site:

http://www.redbooks.ibm.com

- *Using LDAP for Directory Integration A Look at IBM SecureWay Directory, Active Directory and Domino*, SG24-6163-00
- *IBM WebSphere V4.0 Advanced Edition Handbook,* SG24-6176-00
- *Access Integration Pattern Using IBM WebSphere Portal Server*, SG24-6267-00
- *IBM WebSphere Portal V4.1 Handbook Volume 1,* SG24-6883
- *IBM WebSphere Portal V4.1 Handbook Volume 2,* SG24-6920
- *IBM WebSphere Portal V4.1 Handbook Volume 3,* SG24-6921

12.7 Installation planning

This section provides information on planning the installation of Portal for Voice. Before installing the components of Portal for Voice, it is important that you become familiar with the requirements, the supported installation environments, and Portal for Voice Installer, the installation program that installs the components. You should also review the installation planning worksheet, which helps you collect the information required by Portal for Voice Installer.

12.8 Installation scenarios

This section presents installation scenarios--that is, examples of configurations suitable for users with different needs. Choose the one that most closely resembles the environment you are setting up.

The first example configuration is for a development environment in which you develop voice applications for Portal for Voice. In this configuration, you need to install Portal for Voice on an installation of Portal Server for which Development has been selected as the install type. You also need to install WebSphere Voice Server SDK on a machine other than the Portal for Voice. For details, see 12.10.4, “Software” on page 498. You must not set a numeric alias for user ID and password. For details, see 12.14.2, “Numeric alias for user ID and password” on page 500.

The second example configuration is for a production environment in which real users get access to Portal for Voice. In this configuration, you need to install Portal for Voice on an installation of Portal Server for which Typical has been selected as the install type, and Database and LDAP Directory mode for the authentication mode. You also need to install WebSphere Voice Server on a machine other than the Portal for Voice. For details, see 12.10.4, “Software” on page 498.

12.9 Installation planning worksheet

This planning sheet describes examples of the information you will need when you install Portal for Voice. Before you begin to install, print this worksheet and collect as much information as possible for the components you want to install.

You will need this information when you want to uninstall, so keep this worksheet for future reference.

Table 12-1 Feature worksheet

Features	Description	Write your choice here:
Voice basic feature (Aggregator and sample portlet)	This feature is mandatory. It includes Voice Aggregator.	You need to install this feature.
Numeric Alias Support	If you install Portal for Voice in the production environment, and you have not created a numeric user ID and password in Portal Server, select this feature.You can use it in setting a numeric alias for user ID and password.For details, see 12.14.2, “Numeric alias for user ID and password” on page 500. If you select this feature, Portal for Voice searches only numeric alias for user authentication. If you do not install it, Portal for Voice looks for user ID created in Portal Server. In that case, you need to create a numeric user ID and password. If you are installing Portal for Voice in the development environment, do not select this feature.	
Voice for Notes	If you plan to get access to Notes mail DB with Portal for Voice, select this feature.	

Table 12-2 Parameter worksheet

Parameter	Defaults	Write your values here:
Directory where Portal Server is installed	C:\WebSphere\PortalServer	
Directory where WebSphere Application Server is installed	C:\WebSphere\AppServer	

Parameter	Defaults	Write your values here:
Node Name in WebSphere Application Server	*Hostname* in which Portal for Voice Installer is running.	
Administrator's ID of Portal Server	(wpsadmin)	
Administrator's password of Portal Server	(blank)	
Host name or IP address of LDAP Server where Portal Server gets access	*Hostname* in which Portal for Voice Installer is running. *Hostname* should be a fully qualified domain name.	
Port number of LDAP Server that Portal Server gets access	389	
Administrator's ID of LDAP Server for WebSphere Application Serve.	cn= Enter the administrator's ID of LDAP after "cn=." For example, cn=adminusr.	
Administrator's password of LDAP Server for WebSphere Application Server.	(blank)	
IP address of Voice Server you want to give the access authorization to Portal for Voice	IP address in which Portal for Voice Installer is running. You should change it to the IP address of your Voice Server. You can set up to 10 Voice Servers in the Portal for Voice Installer. If you need to configure more than 10 Voice Servers, add them to the property file of Portal for Voice. See 12.14.4, "Customizing Portal for Voice" on page 502.	
Notes ID file of a super user	C:\Lotus\Notes\user.id. Change it to the path name of the Notes ID file of a super user.	

12.10 Requirements

This section provides information about the requirements for installing the components of Portal for Voice.

- Hardware
- Disk space
- Operating systems
- Software

12.10.1 Hardware

The following is the recommended hardware:

- Windows processor:

  An Intel Pentium III 1 GHz processor or better

- Memory (in addition to what is required for the operating system):

  512 MB minimum if used in a proof-of-concept environment. 1GB is recommended.

- Hard drive:
  - 1 GB or more for installing an application. More may be needed as the application operates.
  - 1 GB or more for storing application data. Storage space depends on the amount of data you have.
- CD-ROM drive on network
- Mouse or pointing device and keyboard
- Support for a communications adapter
- Network software: TCP/IP network software installed
- SVGA or better display resolution

12.10.2 Disk space

Each component may require a different amount of disk space. The amount of disk space required depends on the number of users and amount of application data that you expect to have on your system.

12.10.3 Operating systems

The components of Portal for Voice run on the following operating systems. Exceptions, if any, are listed under each operating system.

Windows 2000

- Windows 2000 Server with Service Pack 2
- Windows 2000 Advanced Server with Service Pack 2

To obtain the latest version, go to http://www.microsoft.com.

12.10.4 Software

You must have the following products installed:

- WebSphere Portal Enable Version 4.1.2 offering

Note: If you are installing WebSphere Portal for the production environment, you must select **Typical** for the install type, and select **Database and LDAP** for the authentication mode. If you are planning to install the Numeric Alias Support feature, you must install SecureWay Directory and not other LDAP Servers. If you are installing WebSphere Portal for the development environment, you must select **Development** for the install type.

- Lotus Notes client R5.08 or higher (for Voice for Notes)

You must have the following products installed on a machine other than the Portal for Voice:

- For the production environment:
  - WebSphere Voice Server 2.0 for Cisco/Dialogic (The voice browser must be replaced with a new one that comes with this package. See \fixes\wvs20\readme.txt.)
  - WebSphere Voice Server 3.1 for WebSphere Voice Response (AIX/Windows)
- For the development environment: WebSphere Voice Server SDK 3.1
- Lotus Domino Server R5.08 or higher (for Voice for Notes)

12.11 Portal for Voice installation

The following is the procedure for installing Portal for Voice.

12.11.1 Launch Portal for Voice Installer

Perform the following steps to install Portal for Voice:

1. Print out the installation planning worksheet, and fill it out.

2. Start LDAP Server, WebSphere Application Server, and Portal Server.
3. Insert the CD into the CD-ROM drive.
4. Open a command prompt, and change directory to the CD drive.
5. To run Portal for Voice Installer, type `install.bat` and press Enter.
6. Portal for Voice Installer starts. Click **Next** to continue.
7. The IBM Program License Agreement window opens. Click **Accept**.
8. You will be prompted for the features you will install:
   - Portal for Voice
   - Numeric Alias Support
   - Voice for Notes
   - Voice Aggregator

   For our example, we selected **Portal for Voice** and Numeric **Alias Support**.
9. Type the name of the portal node server. We typed `venus.itso.raleigh.ibm.com`. Click **Next**.
10. Type the portal admin ID and password. Click **Next**.
11. Type in the LDAP server parameters for the WebSphere Application Server as well as the LDAP user ID and password. Click **Next**.
12. Click **Finish** and reboot when the installation has completed.

12.12 What to do after installation

After installation, you must verify that the installation has been completed successfully. To do so, check the log that Portal for Voice Installer generates. The log file can be found in the following directory:

\WebSphere\PortalServer\log\ibmvainstlog.txt

Then, you must configure many of the components before you can use them. For additional configuration information, refer to the following related information.

12.13 Verifying installation

This section explains the verification process after installation.

Portal for Voice provide a sample voice portlet, HelloWorld.war, that you can use in checking whether Portal for Voice is correctly installed and configured. This sample portlet is located in the <WebSphere>\PortalServer\install\voice\HelloWorld.war directory.

Install the sample portlet and place it on one of voice-enabled pages where VoiceXML markup (VXML) is marked. Then dial one of the phone numbers you set up for getting access to Portal for Voice, or run Voice Server SDK specifying the following URL:

```
http://<hostname of Portal for Voice>/wps/portal/.scr/Login
```

After logging in with voice or DTMF tones, navigate through the place, page, and service (portlet) menu structure by voice or DTMF tones to reach the HelloWorld portlet. If you can listen to the HelloWorld portlet successfully, the Portal for Voice is correctly installed and configured.

12.14 Administering

In this section, we discuss administering Portal for Voice.

12.14.1 Creating a new user in Portal Server

If you have not yet created users for Portal for Voice or Voice for Notes in Portal Server, you must create them. For details, see the InfoCenter of Portal Server. For example, if you have installed the the Voice for Notes, the users of Voice for Notes must be associated with users in the Domino Server. To do so, the administrator must tell the users to set the related data, such as the name of the Notes server and the Notes user short name and the Internet password in the Preferences window. To learn more on how to do this read the "How to set your preferences" in wpt4va.pdf.

12.14.2 Numeric alias for user ID and password

Because an ordinary telephone can enter only numerical values, a voice user needs a second user ID and password that are all numeric. The administrator has to tell each user to set a numeric user ID and password in the edit profile page of the Portal Server before using the product.

To enable voice access for the portal, do the following:

1. In the Edit profile page, specify a language, a numeric ID, and a numeric password.
2. Define the portal pages and place that support voice.
3. Place the services on the page.

Note: Some of the steps--for example, defining voice pages and selecting portlets that can be used with voice--may have already been done by the system administrator.

Note: In the development environment, you cannot use a numeric alias for a user ID and password. If you need access control, create a numeric user ID and password on the Portal Server.

You can change the maximum and minimum length of the numeric alias for a user ID and password by changing the following file:

x:\WebSphere\AppServer\lib\app\config\puma.properties

Table 12-3 Setting parameters for numeric alias

Property	Default value	Description
puma.NUID.min	3	Minimum length of numeric alias for user ID
puma.NUID.max	60	Maximum length of numeric alias for user ID
puma.NPASSWORD.min	5	Minimum length of numeric alias for password
puma.NPASSWORD.max	60	Maximum length of numeric alias for password

12.14.3 Editing your profile

Before any use of voice service is possible, you need to specify another user ID and password that is specific to voice. Because only numbers can be entered at a standard phone, each of these must consist entirely of numbers. This pair of numbers is called the numeric alias.

To do this, do as follows:

1. Log in to the portal.
2. On the portal navigation bar, click **Edit my profile** icon. Your profile is displayed.
3. Specify your numeric user ID and password.
4. When you are finished, click **Continue**. The confirmation message is displayed.
5. Click **Continue** to proceed with the changes, or click **Cancel** to discard the changes.

Following is a sample of the contents of the puma.properties file:

```
# Numeric user ID and password
puma.NUID.min = 3
```

```
puma.NUID.max = 60
puma.NPASSWORD.min = 5
puma.NPASSWORD.max = 60
:
```

To make the changes effective, you need to restart Portal Server.

12.14.4 Customizing Portal for Voice

You can set the properties shown in the following table in the x:\WebSphere\AppServer\lib\app\config\voiceaccess.properties file.

Table 12-4 Configuring Portal for Voice to access WebSphere Voice Server

Property	Value	Description
voice.hostCheck	“true” or “false”	Set this property to specify whether Portal for Voice is to accept a request from any Voice Server. ► true: reject a request from a Voice Server that is not specified in this property file. Specify the IP address of the Voice Server by using the property “hostx”. ► false: accept a request from any Voice Server.
hostx (x: number)	IP address	Specify the IP address of a Voice Server to which you allow access to Portal for Voice. Do not use host names.

The following is a sample of the contents of the voiceaccess.properties file:

```
# Portal for Voice
voice.hostCheck=true
# Permitted Voice Server list
host1=192.68.10.1
host2=192.68.10.2
host3=192.68.10.3
host4=192.68.10.4
:
```

To make the changes effective, you need to restart Portal Server.

Note: When the voice.hostCheck parameter is set to true, you should specify the hostx parameters to permit authentication requests from the Voice Server.

12.14.5 Customizing WebSphere Voice Server for Cisco or Dialogic

You must set the following properties in file C:\Program files\VoiceServer\cfg\sysmgmt.properties:

```
INBOUND_URLn=http://<hostname of Portal for Voice>/wps/portal/.scr/Login
```

For more information, see “Chapter 5, “Cisco telephony environment” on page 193 and Chapter 6, “Dialogic environment” on page 217.

12.14.6 Customizing WebSphere Voice Server for WebSphere Voice Response

You must set the following properties in the file default.cff:

```
AppName=xxxxxx
Enabled=yes
Parameter=URI, http://<hostname of Portal for Voice>/wps/portal/.scr/Login
AppClass=com.ibm.speech.VXML.DTVoicelet
```

12.15 Sample portlet

Portal for Voice contains a sample voice portlet called HelloWorld. You can get access to the portlet by using a Web browser or telephone. It is a very simple portlet for use in verifying the installation of Portal for Voice. You can find this portlet at:

<WebSphere_root>\PortalServer\install\voice\HelloWorld.war

Everyplace Toolkit 4.1.2 contains a sample voice portlet, called Inventory Management. The document "Writing a voice portlet" contains a description of this sample portlet.

12.16 Troubleshooting

This topic discusses tools and methods for troubleshooting.

12.16.1 Installation problems

- Make sure that all the requirements are met. See 12.10, “Requirements” on page 497. Check the log file of Portal for Voice Installer. It is in \WebSphere\PortalServer\log\ibmvainstlog.txt.

- Make sure that the parameters you enter on the installation planning worksheet are correct, and that you have entered them in Portal for Voice Installer. See 12.9, “Installation planning worksheet” on page 494.
- Make sure that Portal Server is installed under the same directory and at the same level as the directory of WebSphere Application Server.
- Make sure that LDAP server, WebSphere Application Server, and WebSphere PortalServer are up and running before starting Portal for Voice Installer.

An attempt to call Portal for Voice gets no answer or busy tone

Check that WebSphere Voice Server is working. Contact the administrator of WebSphere Voice Server or see the *Administrator's Guide for WebSphere Voice Server*.

Authentication failure

- Make sure that your numeric user ID and password are authorized to get access to Portal for Voice. If you use an alphanumeric user ID and password for Portal Server, you need to create a numeric alias. See 12.14.2, “Numeric alias for user ID and password” on page 500.
- Make sure that LDAP server is up and running. In the current version, only SecureWay Directory is supported. See requirements of 12.10.4, “Software” on page 498.
- Make sure that all mandatory parameters have been set as input to Portal for Voice Installer. See 12.9, “Installation planning worksheet” on page 494.
- Make sure that the parameters in voiceaccess.properties are set correctly. See 12.14.4, “Customizing Portal for Voice” on page 502.

Inability to get access to Portal for Voice

- Make sure that WebSphere Voice Server and Portal for Voice can communicate over TCP/IP.
- Make sure that WebSphere Application Server and Portal Server are up and running.

A

VXML application

This appendix contains the sample application for the IBM WebSphere Airlines.

A.1 IBM WebSphere Airlines

```
<?xml version="1.0" encoding="iso-8859-1"?>     <!DOCTYPE VXML PUBLIC "VXML"
"">

<VXML version="1.0"><ibmlexicon><word spelling="Brisbane"
pronunciation="&#712;br&#618;z.b&#616;n"></word><word spelling="sydney"
pronunciation="&#712;s&#618;d.ni"></word><word spelling="Springs"
pronunciation="spr&#618;&#331;z"></word><word spelling="Canberra"
pronunciation="w&#616;&#712;notto&#650;"></word><word spelling="Start"
pronunciation="st&#593;rt"></word><word spelling="economy"
pronunciation="&#616;&#712;k&#652;.n&#601;.mi"></word><word spelling="Kempny"
pronunciation="&#712;vur.&#616;s"></word><word spelling="jiong"
sounds-like="Jhong"/><word spelling="dadhich"
pronunciation="&#712;d&#230;dh&#618;&#679;"/><word spelling="dietrich"
pronunciation="&#712;di.tr&#616;k"/><word spelling="Poggioli"
pronunciation="&#716;p&#593;.&#676;i&#712;o&#650;.li"/></ibmlexicon>
<meta name="Content-Type" content="text/x-VXML"/>
<var name="User" expr="'No user defined'" />
<var name="AwardBooking" expr="'No booking defined'" />
<var name="PassengersNo" expr="'No number defined'" />
<var name="Traveller" expr="'Yes'" />
<var name="DestinationCity" expr="'No destination'" />
<var name="DepartureCity" expr="'No departure city'" />
<var name="DepartureDate" expr="'No departure date'" />
<var name="DepartureTime" expr="'No departure time'" />
<var name="ArrivalTime" expr="'No arrival time'" />
<var name="CurrentFlight" expr="1" />
<var name="Flight1" expr="'No flight number. '" />
<var name="Flight2" expr="'No flight number. '" />
<var name="Flight3" expr="'No flight number. '" />
<var name="Flight4" expr="'No flight number. '" />
<var name="Flight5" expr="'No flight number. '" />
<var name="StringDate" expr="'No date'" />
<var name="TravelClass" expr="'Economy'"/>
<var name="SelectedFlight" expr="'No flight selected'"/>


<link next="#TravelClass">
   <grammar>Any</grammar>
</link>


<link next="#registration">
   <grammar>Start over</grammar>
</link>

<link next="#good_bye">
```

```
   <grammar>[Good] [bye] bye | exit </grammar>
</link>

<form id="welcome">
   <block name="Hello">
      <audio src="viavoice.au">
      </audio>
      <audio src="Welcome.au">
      Welcome to IBM WebSphere Airlines Frequent Flyer Award Bookings.
      </audio>
      <audio src="Anytime.au">
      At anytime you can say start over, ask for help or exit.
      </audio>

      <goto next="#registration"/>
   </block>
</form>
<form id="registration">
   <field name="frequent_flyer" type="digits">
      <audio src="FFNumber.au">
         What is your IBM WebSphere Airlines frequent flyer number?
      </audio>
      <help><audio src="HelpDigits.au">
         You can say any sequence of digits.
         </audio>
      </help>
   </field>


<field name="frequent_flyer_confirm" type="boolean">
      <prompt>You frequent flyer nunmber is <value expr="frequent_flyer" />.
Is this correct?</prompt>
   <filled>
        <if cond="frequent_flyer_confirm">
      <goto nextitem="name" />
      <else />
          <clear namelist="frequent_flyer" />
          <clear namelist="frequent_flyer_confirm" />
          <goto nextitem="frequent_flyer" />
      </if>
      <clear/>
      </filled>
      </field>



<field name="name" >
      <audio src="Name.au">
         What is your last name?
```

```
      </audio>
      <grammar>
         Dadhich | Dietrich | Kempny | Hu |  Jiong | Poggioli | Credle
      </grammar>
   </field>


   <filled>
         <assign name="User" expr="name" />
   </filled>

<field name="name_confirm" type="boolean">
      <prompt>You stated your name is<value expr="name" />.  Is this
correct?</prompt>
   <filled>
        <if cond="name_confirm">
      <goto nextitem="PIN" />
      <else />
          <clear namelist="name" />
          <clear namelist="name_confirm" />
          <goto nextitem="name" />
      </if>
      <clear/>
      </filled>
      </field>


<field name="PIN"  type="digits">
      <audio src="PIN.au" >
         What is your PIN number?
      </audio>
      <help><audio src="HelpDigits.au">
         You can say any sequence of digits.
         </audio>
      </help>

   </field>

   <filled>
      <if cond="User == 'Kempny'">
         Welcome Mister Guy Kempny
      <elseif cond="User == 'Dadhich'" />
         Welcome Misses Suchita Dadhich
      <elseif cond="User == 'Hu'" />
         Welcome Mister Dave Jiong Hu
      <elseif cond="User == 'Poggioli'" />
         Welcome Mister Justin Poggioli
      <elseif cond="User == 'Dietrich'" />
         Welcome Misses Ekaterina Dietrich
```

```
      <elseif cond="User == 'Credle'" />
         Welcome Mister Rufus Credle
      </if>
      <audio src="PtsAvailable.au">
      You currently have 25,000 frequent flyer points available, to make an
Award booking.
      </audio>
      <goto next="#itinerary"/>
   </filled>
</form>

<form id="itinerary">
   <field name="build_itinerary" type="boolean">
      <prompt bargein="true">
         <audio src="Itinerary.au">
         Would you like to create an itinerary?
         </audio>
      </prompt>
      <help><audio src="HelpItinerary.au">
         You can say yes or no.
         </audio>
      </help>

   </field>

   <filled>
      <if cond="build_itinerary">
         <goto next="#passengers"/>
      <else/>
         <goto next="#good_bye"/>
      </if>
   </filled>
</form>

<form id="passengers">
   <field name="passengers" type="number">
      <prompt>
         <audio src="Passengers.au">
         How many passengers will be travelling on this booking?
         </audio>
      </prompt>
      <filled>
         <assign name="PassengersNo" expr="passengers" />
         <if cond="passengers == '1' ">
            <goto next="#booking" />
         </if>
      </filled>
      <help><audio src="HelpPassengers.au">
         Please say the number of passengers travelling on this booking.
```

```
            </audio>
         </help>

      </field>

      <field name="travelling" type="boolean">
         <prompt>
            <audio src="Traveller.au">
            Are you one of the travelling passengers?
            </audio>
         </prompt>

         <filled>
            <assign name="Traveller" expr="travelling" />
            <goto next="#booking" />
         </filled>
      </field>
   </form>


   <form id="booking">
      <field name="booking">
         <prompt>
            <audio src="Booking.au">
            What type of Award booking would you like to make?
            </audio>
         </prompt>
         <help>
            <audio src="BookingHelp.au">
            Available award bookings are:
            Single Destination Return,
            Single Destination One-Way, or
            Multi-Destination
            </audio>
         </help>
         <grammar>
            Single Destination Return | Single Destination One-Way |
   Multi-Destination
         </grammar>
         <filled>
            <if cond="booking == 'Single Destination One-Way'">
               <goto next="#SDR" />
            <else />
               <goto next="#SDOW"/>
            </if>
         </filled>
      </field>
   </form>
```

```
<form id="SDR">
   <field name="departure">
      <prompt>
         <audio src="DepartureCity.au">
         What is the departure city?
         </audio>
      </prompt>
      <grammar>
          Brisbane | Sydney | Alice Springs | Canberra | Darwin | Perth |
Melbourne
      </grammar>

      <help><audio src="HelpCities.au">
         Valid cities are: Brisbane, Sydney, Melbourne, Alice Springs,
Canberra, Darwin and Perth
         </audio>
      </help>

      <filled>
         <assign name="DepartureCity" expr="departure" />
      </filled>
   </field>

   <field name="destination">
      <prompt>
         <audio src="DestinationCity.au">
         What is your destination city?
         </audio>
      </prompt>
      <grammar>
         Brisbane | sydney | Alice Springs | Canberra | Darwin | Perth |
Melbourne
      </grammar>

      <help><audio src="HelpCities.au">
         Valid cities are: Brisbane, sydney, Melbourne, Alice Springs,
Canberra, Darwin and Perth
         </audio>
      </help>

      <filled>
         <assign name="DestinationCity" expr="destination" />
      </filled>
   </field>

   <field name="date" type="date">
      <prompt count="1">
         <audio src="TravelDate.au">
```

```
   On which date would you like to travel?
   </audio>
</prompt>
<prompt count="2">
   <audio src="TravelDate2.au">
   When would you like to travel?
   </audio>
</prompt>

<prompt count="3">
   <audio src="TravelDate3.au">
      Travel Date?
   </audio>
</prompt>

<help><audio src="DateHelp.au">
   Please say the date when you would like to travel.
   </audio>
   <audio src="DateEx.au">
   For example, you can say August thirteenth, 2001
   </audio>
</help>

<filled>
   <var name="year" expr="date.substring(0,4)" />
   <var name="month" expr="date.substring(4,6)" />
   <var name="day" expr="date.substring(6,8)" />
   <if cond="year == '????'">
      <assign name="year" expr="2001" />
   </if>
   <if cond="month == '01'">
      <assign name="month" expr="'January'" />
   <elseif cond="month == '02'" />
      <assign name="month" expr="'February'" />
   <elseif cond="month == '03'" />
      <assign name="month" expr="'March'" />
   <elseif cond="month == '04'" />
      <assign name="month" expr="'April'" />
   <elseif cond="month == '05'" />
      <assign name="month" expr="'May'" />
   <elseif cond="month == '06'" />
      <assign name="month" expr="'June'" />
   <elseif cond="month == '07'" />
      <assign name="month" expr="'July'" />
   <elseif cond="month == '08'" />
      <assign name="month" expr="'August'" />
   <elseif cond="month == '09'" />
      <assign name="month" expr="'September'" />
   <elseif cond="month == '10'" />
```

```
            <assign name="month" expr="'October'" />
         <elseif cond="month == '11'" />
            <assign name="month" expr="'November'" />
         <elseif cond="month == '12'" />
            <assign name="month" expr="'December'" />
         </if>


         <if cond="day == '??'">
            <prompt>
               <audio src="SayDay.au">
               You have to say also the day of the month.
               </audio>
            </prompt>
            <clear namelist="date"/>
         <elseif cond="day == '01'" />
            <assign name="day" expr="'first'" />
         <elseif cond="day == '02'" />
            <assign name="day" expr="'second'" />
         <elseif cond="day == '03'" />
            <assign name="day" expr="'third'" />
         <elseif cond="day == '04'" />
            <assign name="day" expr="'four'" />
         <elseif cond="day == '05'" />
            <assign name="day" expr="'five'" />
         <elseif cond="day == '06'" />
            <assign name="day" expr="'six'" />
         <elseif cond="day == '07'" />
            <assign name="day" expr="'seven'" />
         <elseif cond="day == '08'" />
            <assign name="day" expr="'eight'" />
         <elseif cond="day == '09'" />
            <assign name="day" expr="'nine'" />
         </if>


         <assign name="StringDate" expr="month + ' ' + day + ' ' + year" />
         <assign name="DepartureDate" expr="StringDate" />
         <assign name="DepartureDate" expr="date" />
      </filled>
   </field>

   <field name="time" type="time">
      <prompt>
         <audio src="PreferredTime.au">
         What is your preferred travel time?
         </audio>
      </prompt>
      <filled>
```

```
         <assign name="DepartureTime" expr="time" />
         <goto next="#TravelClass"/>
      </filled>
   </field>
</form>

<form id="TravelClass">
   <field name="travel_class">
      <prompt>
         <audio src="TravelClass.au">
         Would you like to travel by economy or business class?
         </audio>
      </prompt>
      <help>
         <audio src="HelpClass.au">
         The seats that you can choose from are either economy or business.
         </audio>
      </help>
      <grammar> economy [class] | business [class] </grammar>
      <filled>
         <assign name="TravelClass" expr="travel_class" />
      </filled>
   </field>


   <filled>
      <assign name="Flight1" expr="'Flight number one: departing at six twenty
five a.m. and arriving at nine fifteen a.m.'" />
      <assign name="Flight2" expr="'Flight number two: departing at eight
thirty five a.m. and arriving at one p.m.'" />
      <assign name="Flight3" expr="'Flight number three: departing at eleven
fourty five a.m. and arriving at three p.m'" />
      <assign name="Flight4" expr="'Flight number four: departing at two p.m.
and arriving at six oh five p.m'" />
      <assign name="Flight5" expr="'Flight number five: departing at four ten
p.m and arriving at nine twenty seven p.m.'" />
      <goto next="#flights" />
   </filled>
</form>


  <script src="browseselectionlist.es">
    var objBrowselelist = new BrowseSelectionList();
  </script>

  <form id="flights">
    <subdialog name="browselelist" src="browseselectionlist.VXML">
      <param name="paramSubdialogObj" expr="objBrowselelist"/>
```

```
      <param name="paramPromptQuestion" expr="'Here are the flights with
available ' + TravelClass + ' class seats, departing from ' +
      DepartureCity + ' to ' + DestinationCity + ' on ' + StringDate + '.
Please select a flight from the following '+
      'list, using the commands Select, Next, and Previous.'"/>

      <param name="paramItemList" expr="Flight1 + '.,' + Flight2  + '.,' +
Flight3  + '.,'  + '.,' +   Flight4  + '.,' +  Flight5"/>

      <filled>
   <assign name="SelectedFlight" expr="browselelist.returnSelectedItem"/>
        <prompt>
           You have selected <value expr="browselelist.returnSelectedItem"/>
        </prompt>
   <goto next="#confirm" />
      </filled>
    </subdialog>
  </form>

<form id="SDOW">
   <block>
      <audio src="NotAvailable.au">
      This option is currently not available.
      Please try another booking.
      </audio>
      <goto next="#booking"/>
   </block>
</form>

<form id="confirm">
   <field name="confirm" type="boolean">
      <prompt>
      <audio src="BookFlight.au">
      Would you like to book this flight?
      </audio>
      </prompt>

      <filled>
      <if cond="confirm">
         <prompt>
         You have booked the following flight:
         <value expr="SelectedFlight"/>.
         Your confirmation number is A, 3, 1, 3, 5, 6, 8, 7.
         </prompt>
         <goto next="#book_another" />
      <else />
         <goto next="#flights" />
      </if>
      </filled>
```

```
   </field>
</form>

<form id="book_another">
   <field name="answer" type="boolean">
      <prompt>
         <audio src="BookAnother.au">
         Do you want to book another flight?
         </audio>
      </prompt>
   </field>
   <filled>
      <if cond="answer">
         <goto next="#passengers" />
      <else />
         <goto next="#good_bye" />
      </if>
   </filled>
</form>

<form id="good_bye">
   <block>
      <audio src="ThankYou.au">
      Thank you for using IBM WebSphere Airlines Frequent Flyer Award
Bookings.
      </audio>
   </block>
</form>

</VXML>
```

B

Voice Server tool

In this chapter we introduce some tools that will aid in the installation and configuration of Voice Server.

B.1 Dialtone determination

When configuring Voice Server, the dialtone frequency values need to be provided by your telephone specialist. However, there are several tools that can be used to collect this information for you.

B.2 Dialogic software

There are several tools that are installed when the Dialogic software is installed. These are found in the Samples folder, within the Dialogic folder.Multithread voice

Multithread voice allows the capture of sound being heard on the phone line. The sound is saved into a WAV file, which then is processed by a third-party audio application.

Step 1: Open application

Mutlithread voice is installed as part of the Dialogic software. It is normally located in the Samples folder that is within the main Dialogic folder. Click it to open.

Step 2: Open file

To capture the audio, a file must be opened. Click **File -> Open** as shown in Figure B-1.

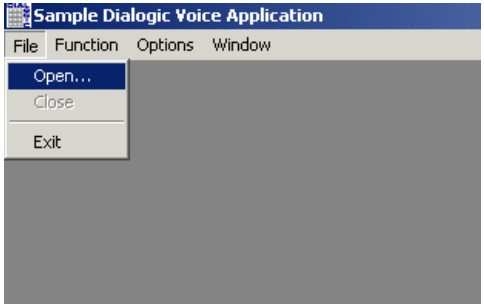


Figure B-1 Opening a file

Step 3: Channel selection

You will be presented with a channel selection window. The values here will vary depending on the type of Dialogic card you have installed. You must select a valid channel that has the telephone connection. In our case the Dialogic card was a D/41JCT**.** We selected line 1 on Channel 1, as shown in Figure B-2 on

page 519. This means that the phone line we are testing will be in the channel 1 socket of the card.

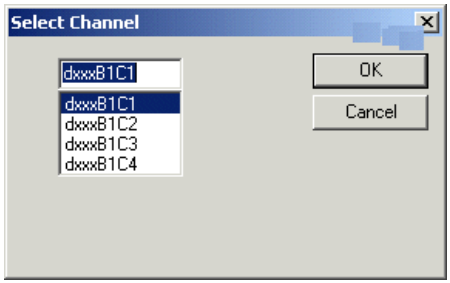


Figure B-2 channel selection

When a channel has been opened successfully, a window will appear saying that it is opened. This can be seen in Figure B-3.

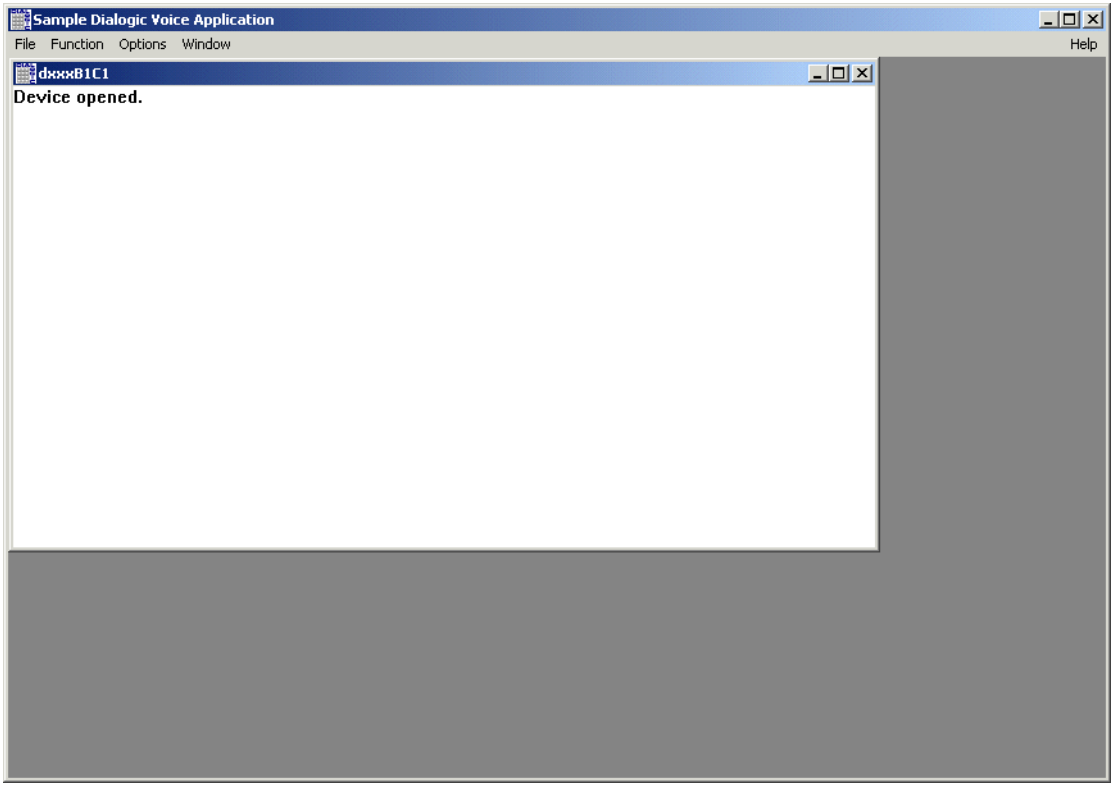


Figure B-3 Opened channel

Step 4: Offhook

Normally the channel will be onhook, meaning the phone is not dialling. To capture the dialtone signal we need to select offhook. Do so now. The channel opened window will say `Setting Offhook.... Done!`

Step 5: Capture WAV file

To capture the sound, a WAV file must be recorded. Click **Function -> Record WAV** as shown in Figure B-4.

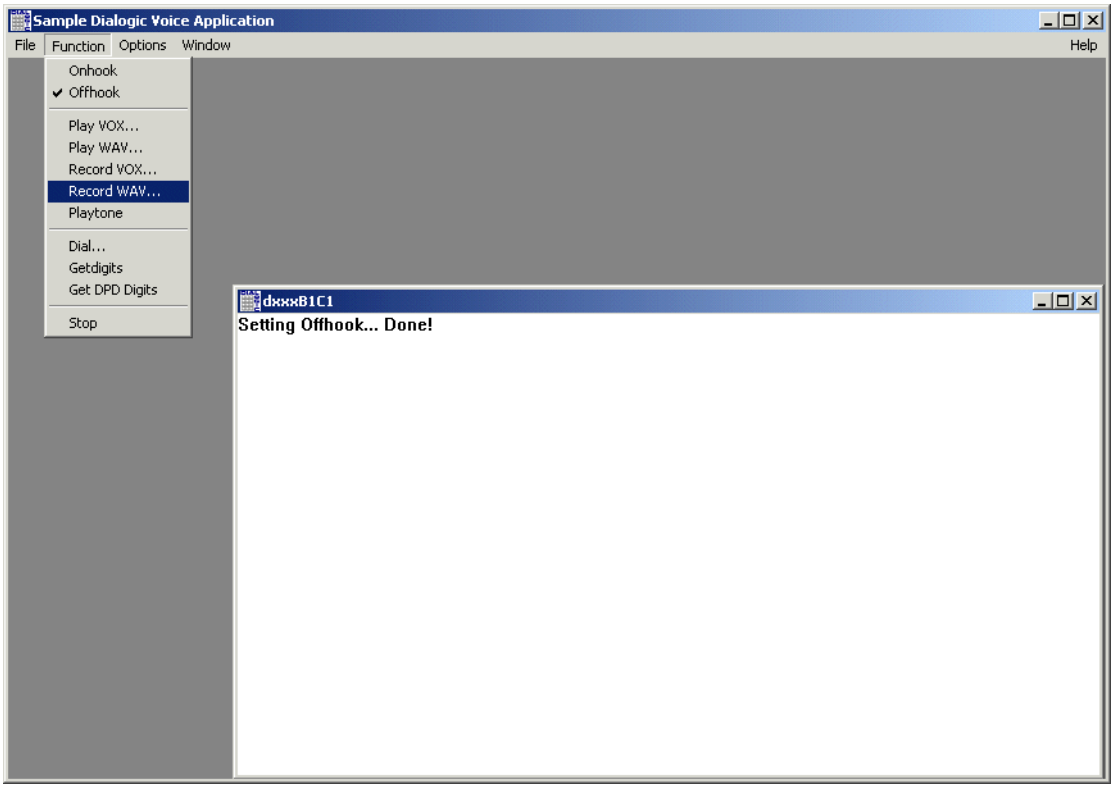


Figure B-4 Record WAV file

A file name and location must be typed and selected before any recording happens. In our example we called the file test.wav and saved it in the voice folder in the Dialogic software, as shown in Figure B-5 on page 521.

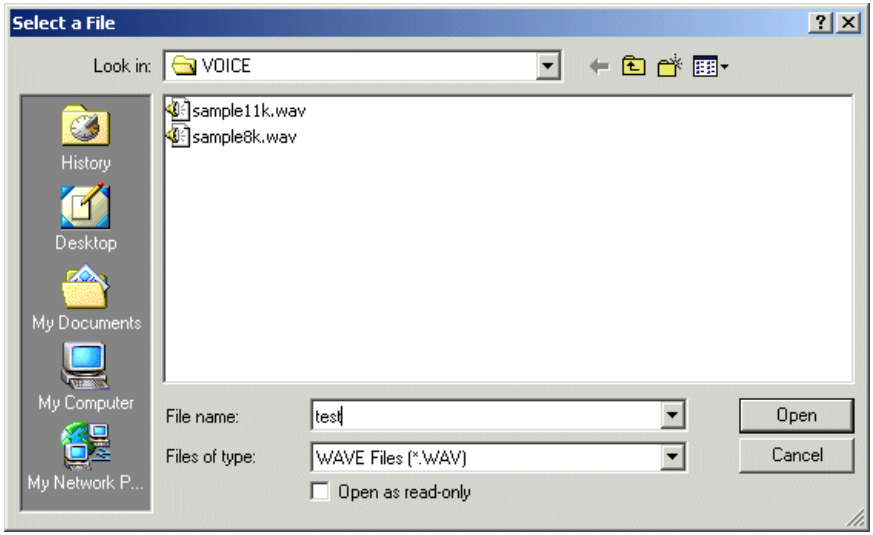


Figure B-5 Naming WAV file and specifying location

The channel window will display a message saying the WAV file is being saved, similar to Figure B-6 on page 522.

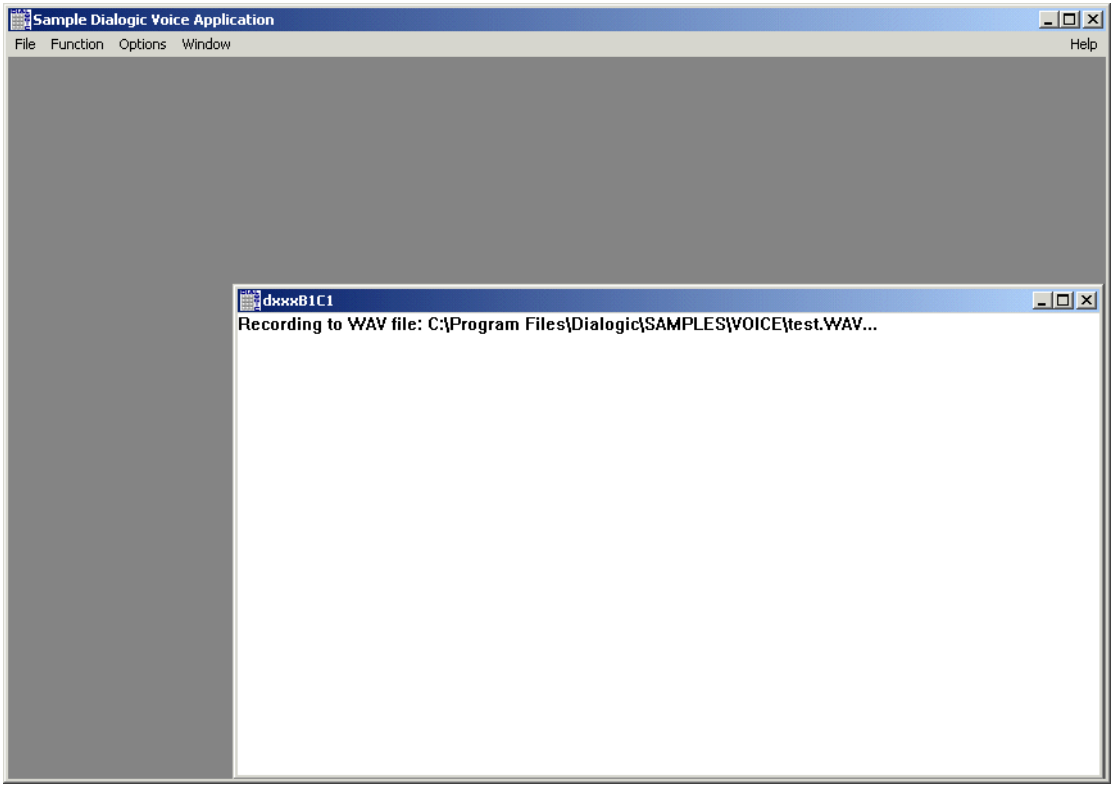


Figure B-6 Saving WAV file

After several seconds, the recording can be stopped by selecting **Function -> Stop**, as illustrated in Figure B-7 on page 523.

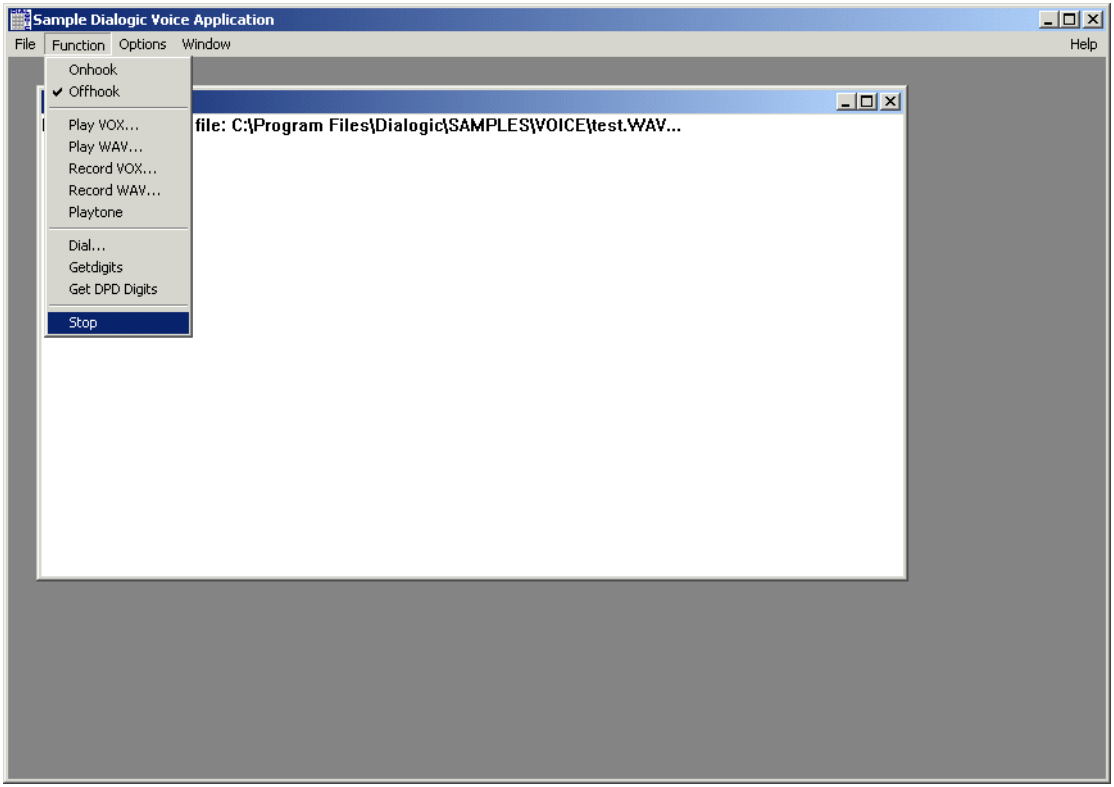


Figure B-7 Stopping WAV file recording

When the file is closed a message indicating this will appear in the channel window, as in Figure B-8 on page 524.

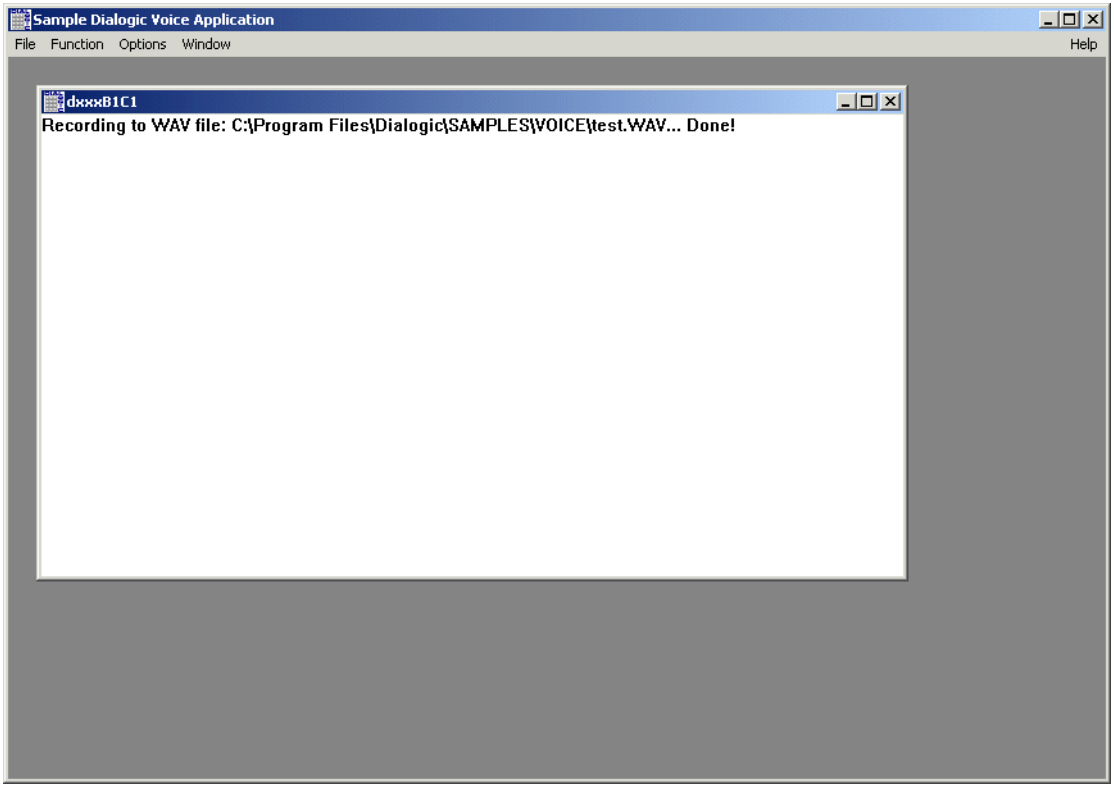


Figure B-8 Message that WAV recording has stopped

To complete the process, select **Function -> Onhook**.

Note: Capturing the audio of the dialtone needs to be done in a timely fashion. Once off the hook, the dialtone only lasts for so many seconds. After this you will only record a busy signal. Try to capture the dialtone as soon as you place the channel in the Offhook setting.

B.2.1 Cooledit 2000

Once the dialtone has been successfully recorded, it has to be analyzed to obtain the dialtone frequencies and range. This is done using an audio editor of some sort. We used a trial version of CoolEdit 2000, which is freely available on the Internet. The product can be downloaded from:

http://www.syntrillium.com

Step 1: Opening file

Once Cooledit 2000 has been installed successfully, open it, as shown in Figure B-9.

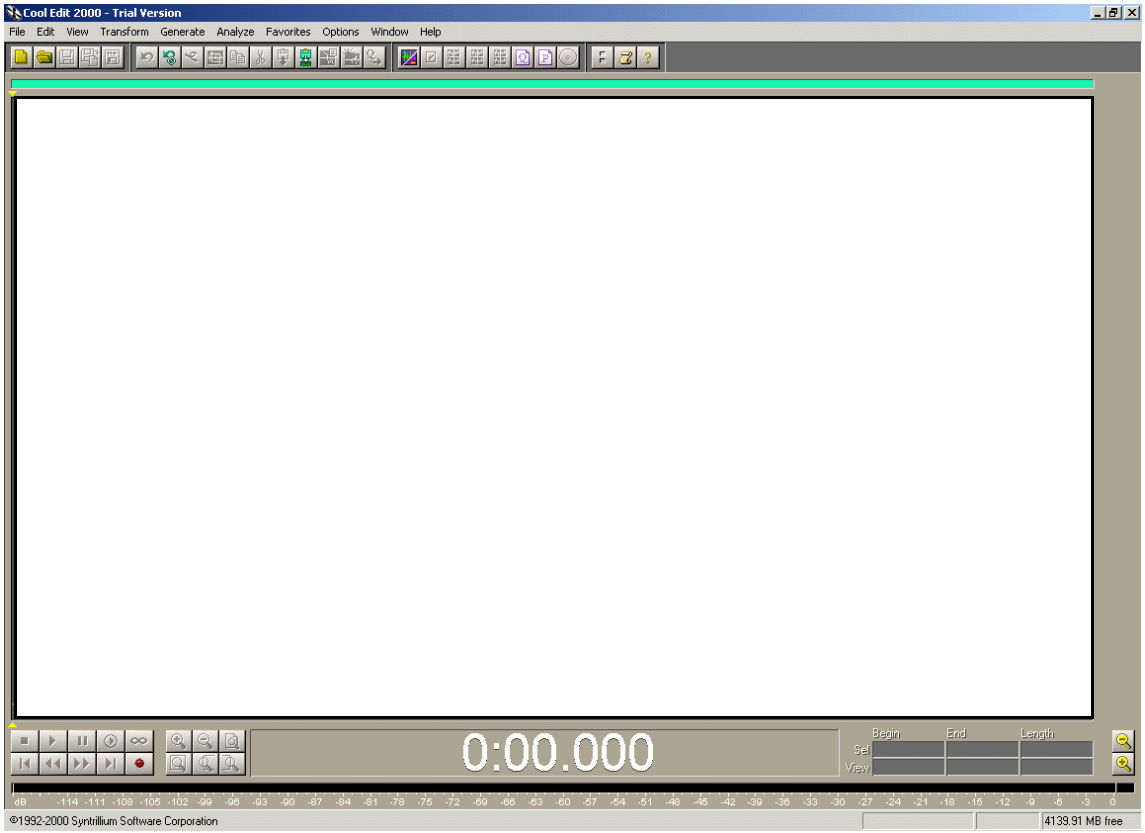


Figure B-9 CoolEdit 2000 opened

Open the WAV file you created. In our case this was called test.wav. Since it was saved in the voice subfolder under the Dialogic folder, we need to use the browse function to find it. Figure B-10 on page 526 illustrates this.

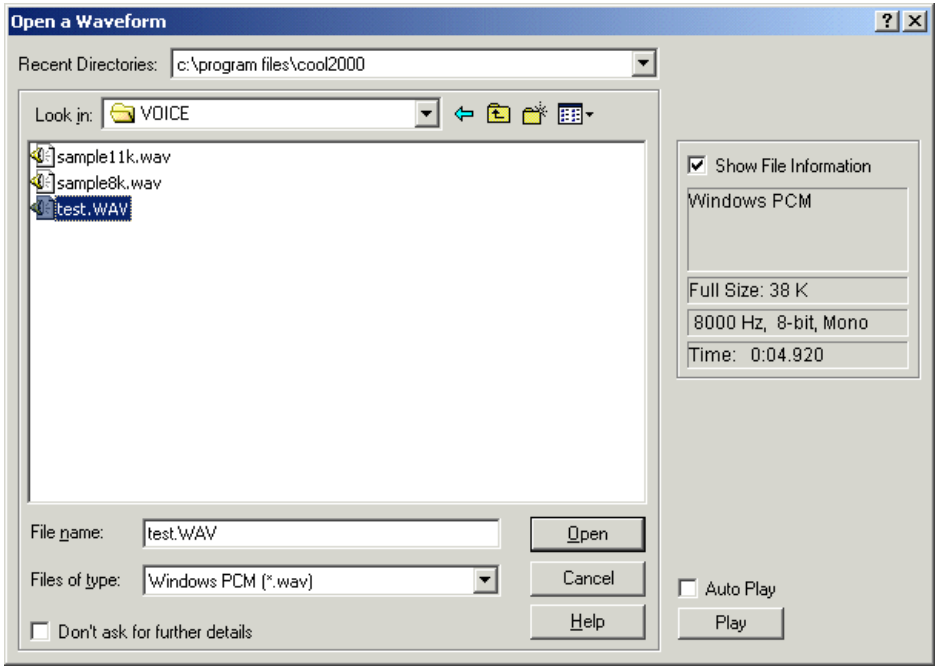


Figure B-10 test.wav file location

The file will look something like Figure B-11 on page 527. You can play the file to confirm the dialtone was successfully recorded. In our case you can see the dialtone is the wider section. Near the end it reduces in size. This is where the engaged tone came into effect.

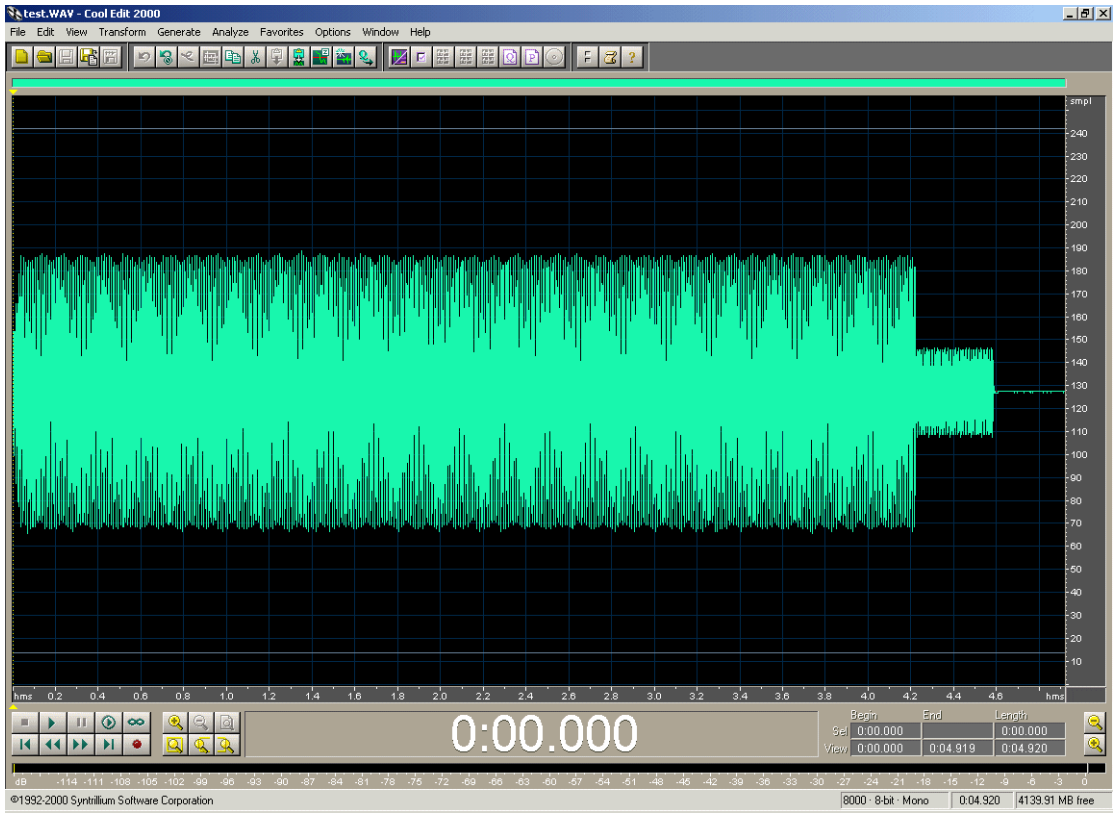


Figure B-11 Dialtone recording

Step 2: Analyzing the dialtone

Highlight a section of the dialtone sound. Once done, select **Analyze -> Frequency Analysis**, as shown in Figure B-11.

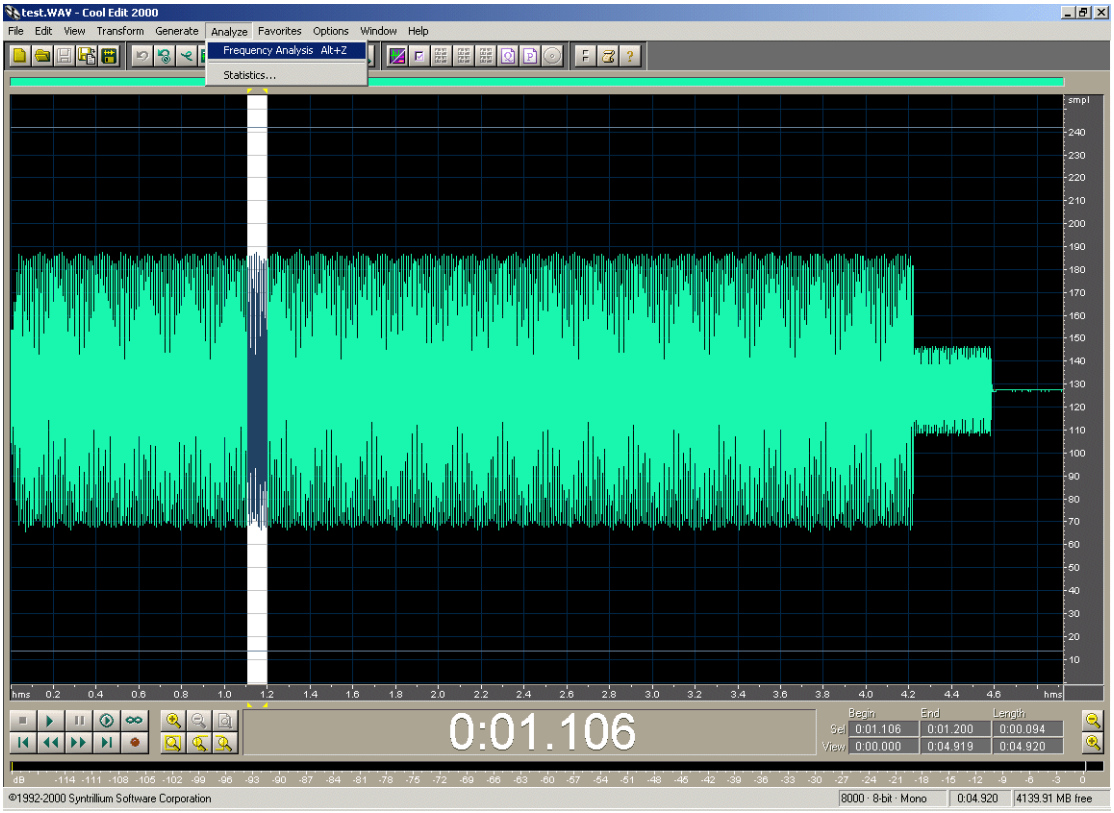


Figure B-12 Analysis process

Note: Highlight a small section only, because this will give you a more accurate frequency range.

Step 4: Working with an analyzed section

The analysis will show a graph representation of the dialtone. There will be two identifiable peaks. These represents the two different tones that make up the dialtone sound. Figure B-13 on page 529 has the dialtone graph.

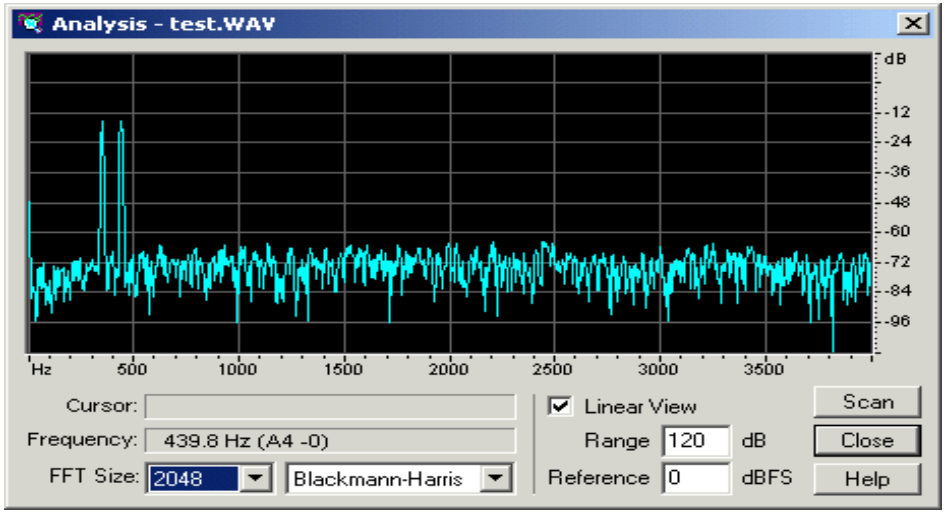


Figure B-13 Dialtone graph

To obtain the dialtone frequency, use the bottom or horizontal scale to determine the point where the peaks are at their highest. To obtain a better value for these, the FFT size can be reduced. In our case, we lowered the value from 2048 to 1024. At the same time we un-ticked the Linear View option. This is shown in Figure B-14.

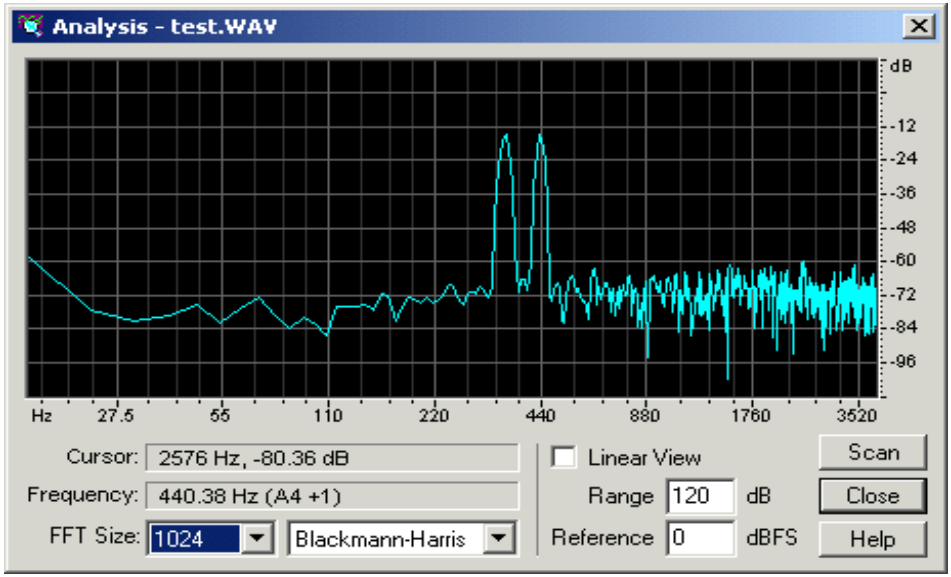


Figure B-14 Non-linear view with FFT of 1024

We determined the first frequency was around 359 and its range of 54. Its range was calculated by subtracting the peak point of -16 from the its low point of -70. Using the same procedure, we determined the second dialtone frequency to be 440 and its range to be 54.

Note: Using this process to calculate the dilatone frequencies is a good indicator. However, finding out the true values from the PBX supplier is always considered the best approach.

B.2.2 PBXpert

PBXpert is another tool included with the Dialogic software. It enables you analyze the difference tones your telephone environment generates. This tool automatically tests the phone line for the difference signals and then presents them in a spreadsheet. It takes out much of the guesswork when compared to the Multithread Voice application.

Our testing environment was done using the D/41JCT-LS 4 port analog card. We had two phone lines plugged into the telephony card, using standard RJ-11 type connectors. The phone lines were connected to a PBX system.

Step 1: Open application

PBXpert is installed as part of the Dialogic software. It is normally located in the main Dialogic folder. When the application is opened, the PBXpert wizard will appear, as shown in Figure B-15 on page 531. Click **Next**.

Figure B-15 Start of PBXpert wizard

Step 2: Wizard setup

The wizard required information about the telephony environment before it we can run the application testing functions. Figure B-16 on page 532 illustrates the PBX information. In our case we entered dummy information. Click **Next**.

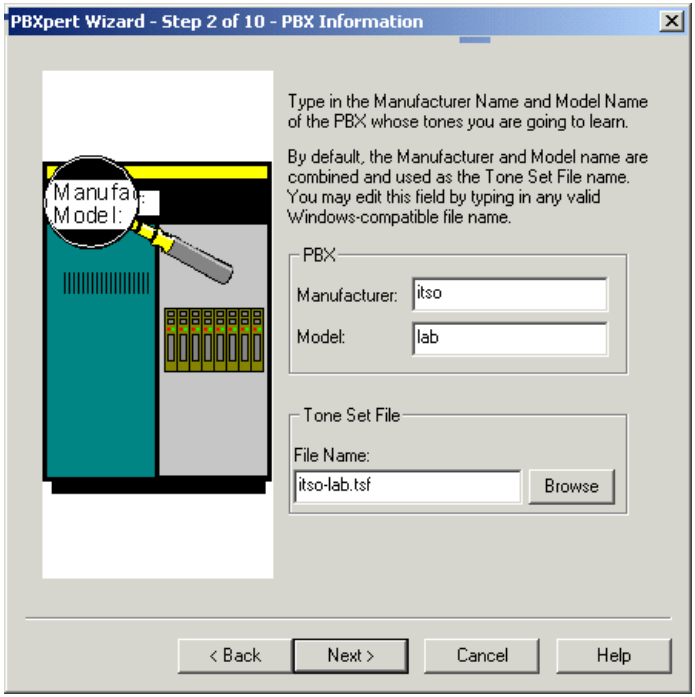


Figure B-16 PBX information

The TAPI information window will appear, as illustrated in Figure B-17 on page 533. We left the default option here. Click **Next**.

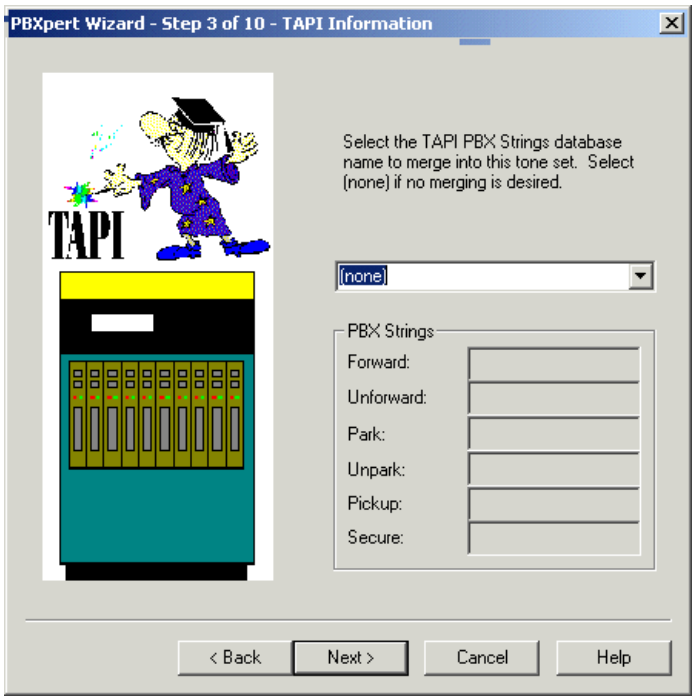


Figure B-17 TAPI information

The wizard asks for the type of Dialogic card in the PC. As ours was the D/41JCT, we selected **#1** to match the D/41JCT. In our case this was the default setting, as shown in Figure B-18 on page 534. Click **Next**.

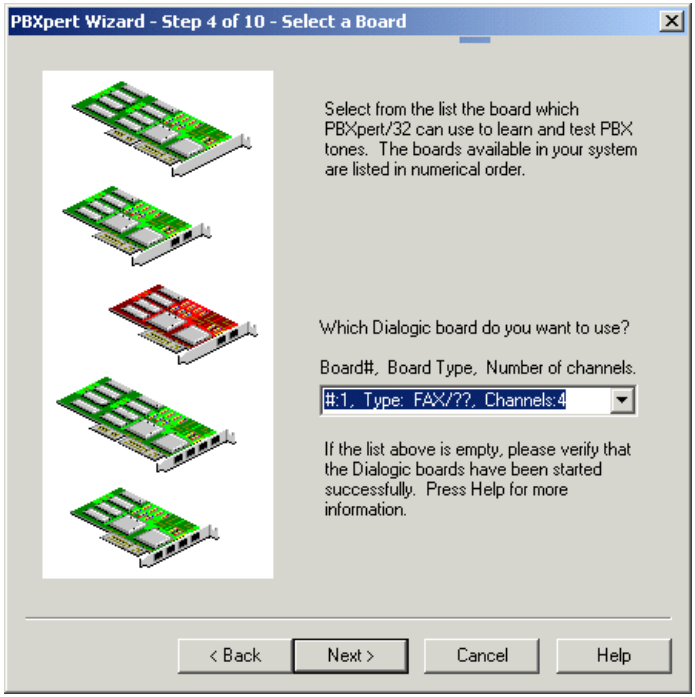


Figure B-18 Board selection

To successfully test the different tone frequencies, at least two phone lines must be available for the application to use. In Figure B-19 on page 535, the first phone line settings are entered. In our lab the phone number for line 1 was 34536 and it was physically connected to channel 1. Click **Next.**

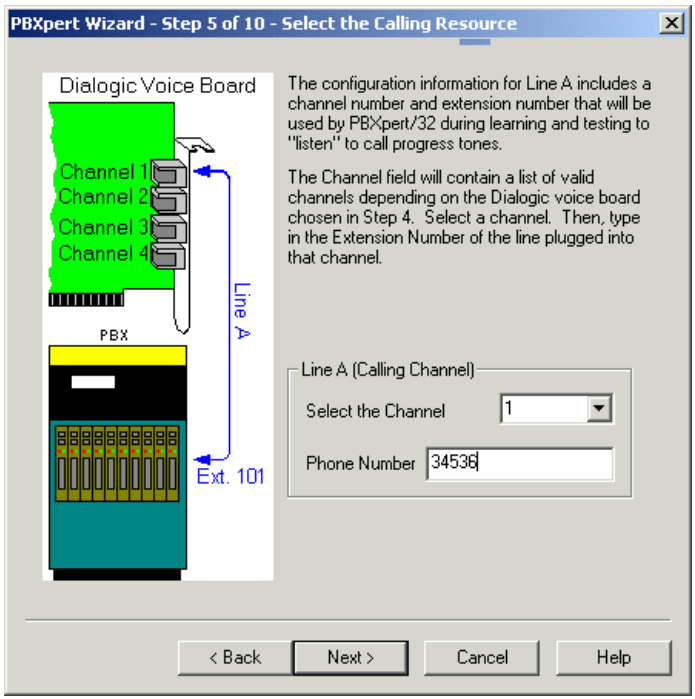


Figure B-19 Phone line 1 settings

Figure B-20 on page 536 displays the settings for phone line 2. In our lab the phone line number was 34524 and it was physically connected to channel 2. Click **Next**.

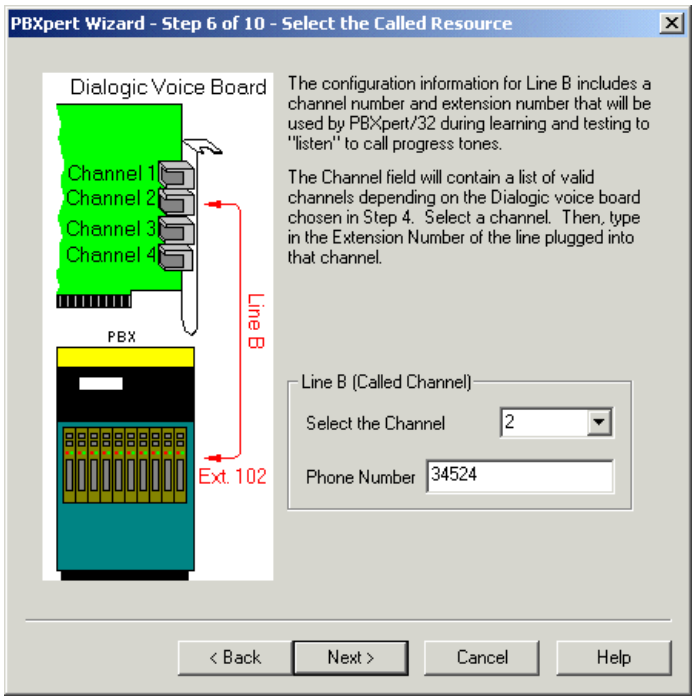


Figure B-20 Phone line 2 settings

The wizard now has enough information to proceed. In our case we de-selected the Run wizard auto-test, as shown in Figure B-21 on page 537. Click **Next**.

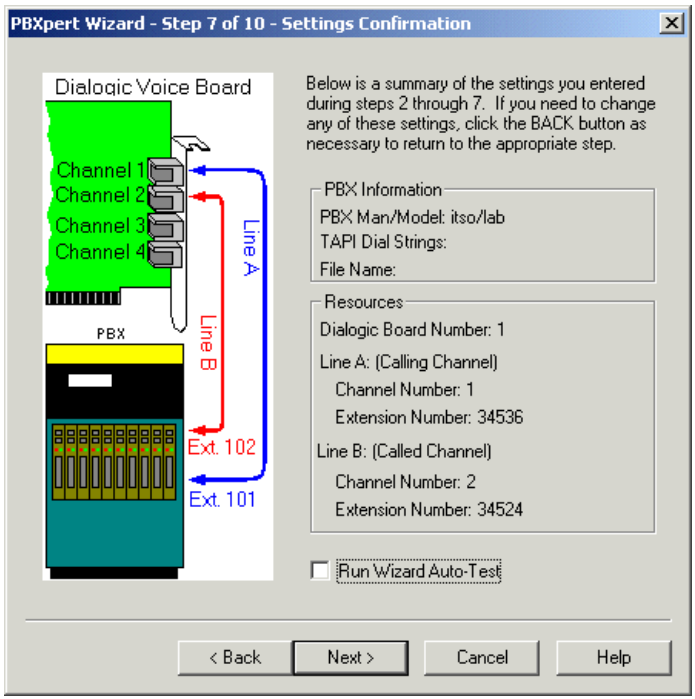


Figure B-21 Wizard summary

Step 3: Learning tones

PBXpert is now ready to test for the telephone system. In Figure B-22 on page 538 the PBXpert application advises the test will start as soon as you click **Next**. Click **Next** to start the test.

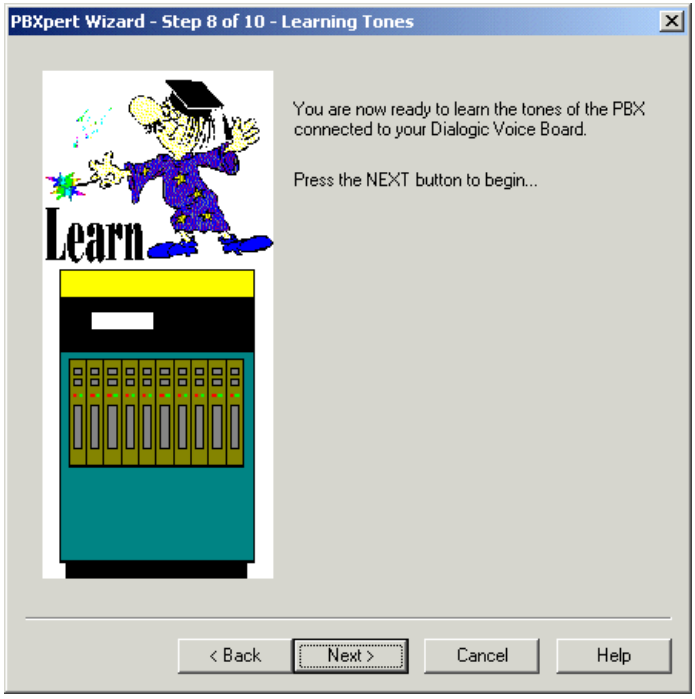


Figure B-22 Learning tones

PBXpert will cycle through the different stages of a phone call, to analyze the different tones. In Figure B-23 the dialtone frequencies are displayed.

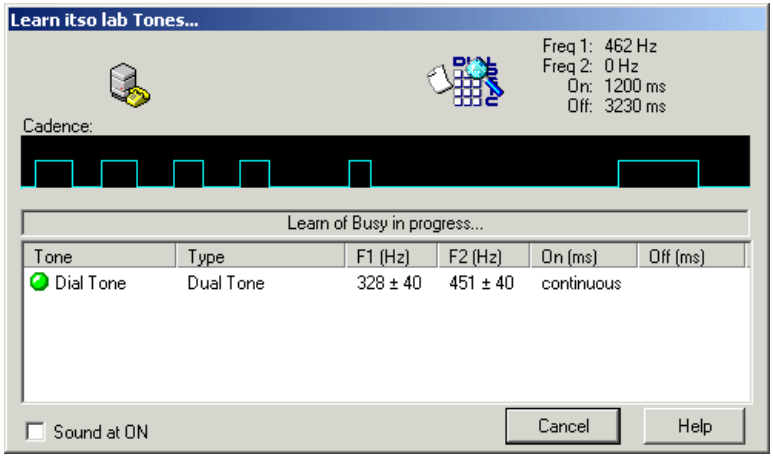


Figure B-23 Dialtone analysis

Figure B-24 shows a summary of all the tones registered and analyzed during the phone call. Click **Keep Data**. This will store the frequencies onto a table.

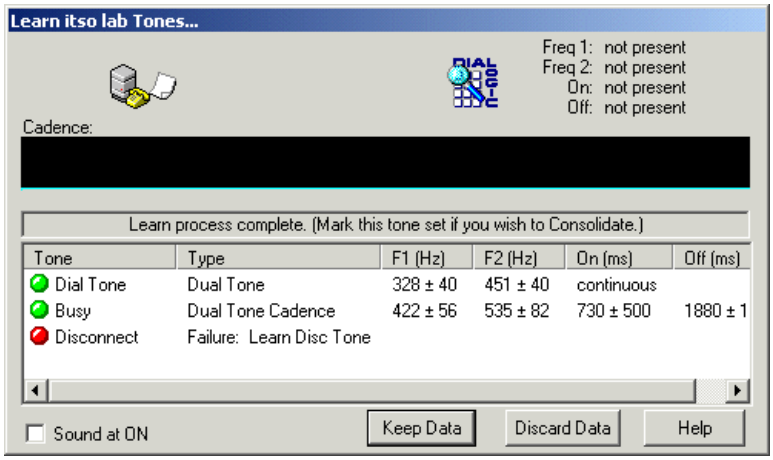


Figure B-24 Tone analysis summary

Step 4: Reviewing tones

PBXpert will prompt you to test the tones as shown in Figure B-25 on page 540. We are interested in the frequencies, so click **Cancel**, to stop any further testing of the phone system.

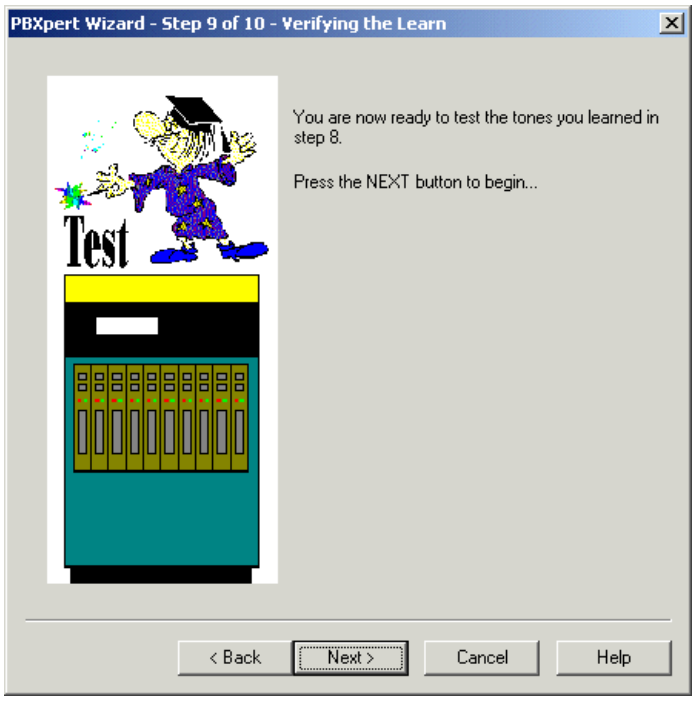


Figure B-25 Testing tones option

The summary table of the telephone tone frequencies will appear. Our lab environment settings are shown in Figure B-26 on page 541. We then used these dialtone frequencies and ranges in the VVTDefaults configuration file.

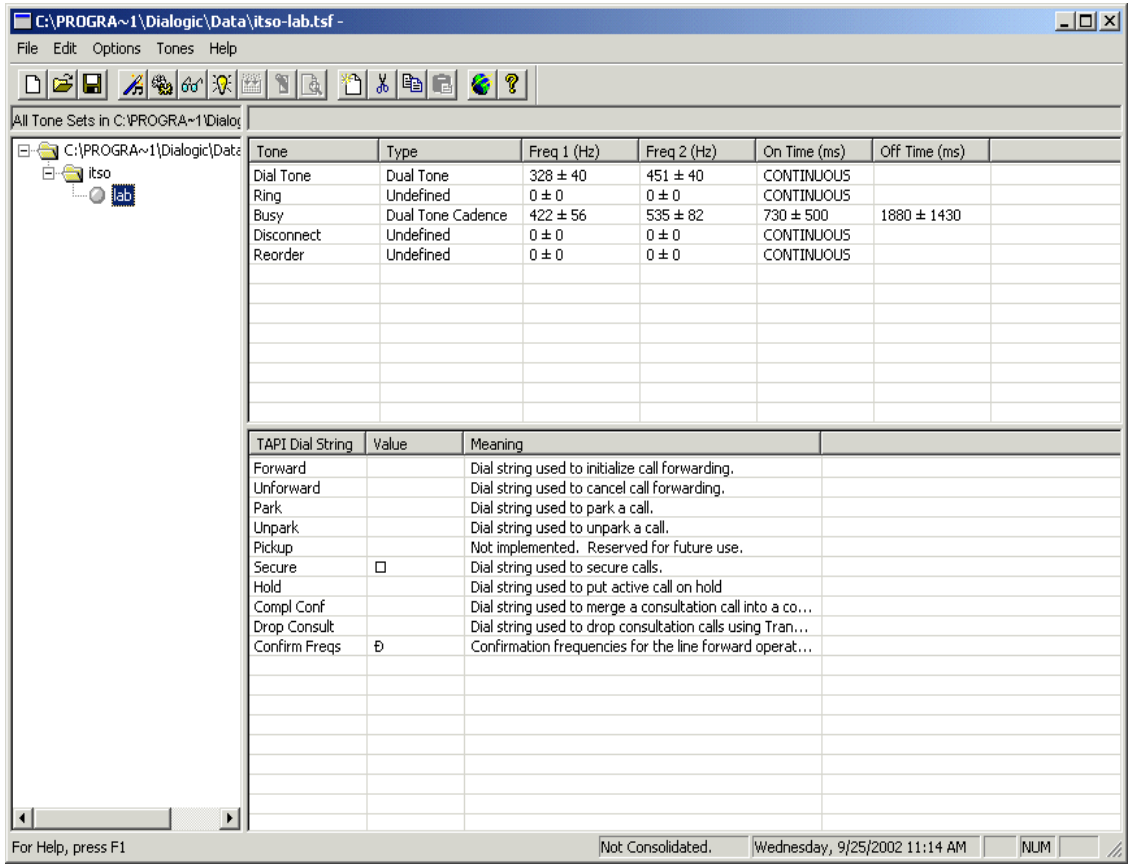


Figure B-26 PBXpert summary table

C

Integration of Windows 2000 and AIX

In this appendix we describe how to integrate Windows 2000 with AIX using SAMBA.

12.17 Samba instruction

We use Samba to share the AIX folder with Windows 2000. Samba is a freeware of AIX. It acts as a daemon in AIX and lets AIX support the protocol NETBUEI, which is used by the Windows network browser.

You can get it from http://www.samba.org. Before downloading it, please read the license carefully.

Important: This software is not included in WebSphere Voice Server 3.1 or WebSphere Voice Response 3.1. If any problems of this software happens, IBM will not provide any support for it.

12.17.1 Installation of Samba

Perform the following instructions to install Samba:

1. Download the newest version of Samba.
2. Install Samba using **smitty install_latest**.
3. With the installation, a /usr/local directory is automatically created. There are two subfolders under it.
   - lib includes the configuration file smb.conf and codepages of Samba.
   - sbin or bin (depending on the version of Samba) includes the smbd exec file.

12.17.2 Configuration of Samba

Edit the configuration file smb.conf of Samba as follows:

```
vi /usr/local/lib/smb.conf
```

Example 12-1 smb.conf

```
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not many any basic syntactic errors.
```

```
#
#======================= Global Settings =====================================
[global]

# workgroup = NT-Domain-Name or Workgroup-Name
; workgroup = SUPPORT
; debug level = 10
; syslog = 0
   workgroup = MYGROUP

# server string is the equivalent of the NT Description field
   server string = Samba Server

# This option is important for security. It allows you to restrict
# connections to machines which are on your local network. The
# following example restricts access to two C class networks and
# the "loopback" interface. For more examples of the syntax see
# the smb.conf man page
;   hosts allow = 192.168.1. 192.168.2. 127.

# If you want to automatically load your printer list rather
# than setting them up individually then you'll need this
   load printers = yes

# you may wish to override the location of the printcap file
;   printcap name = /etc/printcap

# on SystemV system setting printcap name to lpstat should allow
# you to automatically obtain a printer list from the SystemV spool
# system
;   printcap name = lpstat

# It should not be necessary to specify the print system type unless
# it is non-standard. Currently supported print systems include:
# bsd, sysv, plp, lprng, aix, hpux, qnx
;   printing = bsd
printing = aix

# Uncomment this if you want a guest account, you must add this to /etc/passwd
# otherwise the user "nobody" is used
;  guest account = pcguest
guest account = root

# this tells Samba to use a separate log file for each machine
# that connects
   log file = /var/samba/log/log.%m

# Put a capping on the size of the log files (in Kb).
   max log size = 50
```

```
# Security mode. Most people will want user level security. See
# security_level.txt for details.
;   security = user
security = share
# Use password server option only with security = server
;   password server = <NT-Server-Name>

# Password Level allows matching of _n_ characters of the password for
# all combinations of upper and lower case.
;  password level = 8

# You may wish to use password encryption. Please read
# ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba documentation.
# Do not enable this option unless you have read those documents
; encrypt passwords = yes
encrypt passwords = no

# If the following this parameter is set to "yes" (it defaults to "no") and
# an smbpasswd file exists containing all the valid users of a Samba system
# but no encrypted passwords (ie. the Lanman hash and NT hash entries in the
# file are set to "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"), then as users log in
# with plaintext passwords that are matched against their UNIX password
# entries, their plaintext passwords will be hashed and entered into the
# smbpasswd file.
#
update encrypted = yes

# Unix users can map to different SMB User names
;  username map = /etc/smbusers
;  username map = /var/samba/users.map

# Using the following line enables you to customise your configuration
# on a per machine basis. The %m gets replaced with the netbios name
# of the machine that is connecting
;   include = /usr/local/lib/smb.conf.%m

# Most people will find that this option gives better performance.
# See speed.txt and the manual pages for details
   socket options = TCP_NODELAY

# Configure Samba to use multiple interfaces
# If you have multiple network interfaces then you must list them
# here. See the man page for details.
;   interfaces = 192.168.12.2/24 192.168.13.2/24

# Configure remote browse list synchronisation here
#  request announcement to, or browse list sync from:
#   a specific host or from / to a whole subnet (see below)
```

```
;   remote browse sync = 192.168.3.25 192.168.5.255
# Cause this host to announce itself to local subnets here
;   remote announce = 192.168.1.255 192.168.2.44

# Browser Control Options:
# set local master to no if you don't want Samba to become a master
# browser on your network. Otherwise the normal election rules apply
;   local master = no

# OS Level determines the precedence of this server in master browser
# elections. The default value should be reasonable
;   os level = 33

# Domain Master specifies Samba to be the Domain Master Browser. This
# allows Samba to collate browse lists between subnets. Don't use this
# if you already have a Windows NT domain controller doing this job
;   domain master = yes

# Preferred Master causes Samba to force a local browser election on startup
# and gives it a slightly higher chance of winning the election
;   preferred master = yes

# Use only if you have an NT server on your network that has been
# configured at install time to be a primary domain controller.
;   domain controller = <NT-Domain-Controller-SMBName>

# Enable this if you want Samba to be a domain logon server for
# Windows95 workstations.
;   domain logons = yes

# if you enable domain logons then you may want a per-machine or
# per user logon script
# run a specific logon batch file per workstation (machine)
;   logon script = %m.bat
# run a specific logon batch file per username
;   logon script = %U.bat

# Where to store roving profiles (only for Win95 and WinNT)
#        %L substitutes for this servers netbios name, %U is username
#        You must uncomment the [Profiles] share below
;   logon path = \\%L\Profiles\%U

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable it's WINS Server
;   wins support = yes

# WINS Server - Tells the NMBD components of Samba to be a WINS Client
#  Note: Samba can be either a WINS Server, or a WINS Client, but NOT both
;   wins server = w.x.y.z
```

```
# WINS Proxy - Tells Samba to answer name resolution queries on
# behalf of a non WINS capable client, for this to work there must be
# at least oneWINS Server on the network. The default is NO.
;   wins proxy = yes

# DNS Proxy - tells Samba whether or not to try to resolve NetBIOS names
# via DNS nslookups. The built-in default for versions 1.9.17 is yes,
# this has been changed in version 1.9.18 to no.
   dns proxy = no

# Case Preservation can be handy - system default is _no_
# NOTE: These can be set on a per share basis
;  preserve case = no
;  short preserve case = no
# Default case is normally upper case for all DOS files
;  default case = lower
# Be very careful with case sensitivity - it can break things!
;  case sensitive = no

#============================ Share Definitions ==============================
[homes]
   comment = Home Directories
   browseable = no
   writable = yes

# Un-comment the following and create the netlogon directory for Domain Logons
; [netlogon]
;   comment = Network Logon Service
;   path = /usr/local/samba/lib/netlogon
;   guest ok = yes
;   writable = no
;   share modes = no


[grammars]
path = /var/wingrammar
guest ok=yes
writable = yes
browseable = yes

# Un-comment the following to provide a specific roving profile share
# the default is to use the user's home directory
;[Profiles]
;    path = /usr/local/samba/profiles
;    browseable = no
;    guest ok = yes
```

```
# NOTE: If you have a BSD-style print system there is no need to
# specifically define each individual printer
[printers]
   comment = All Printers
;   path = /usr/spool/samba
   path = /tmp
   browseable = yes
# Set public = yes to allow user 'guest account' to print
   guest ok = yes
   writable = no
   printable = yes

# This one is useful for people to share files
[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   public = yes

# A publicly accessible directory, but read only, except for people in
# the "staff" group
;[public]
;   comment = Public Stuff
;   path = /home/samba
;   public = yes
;   writable = yes
;   printable = no
;   write list = @staff

# Other examples.
#
# A private printer, usable only by fred. Spool data will be placed in fred's
# home directory. Note that fred must have write access to the spool directory,
# wherever it is.
;[fredsprn]
;   comment = Fred's Printer
;   valid users = fred
;   path = /homes/fred
;   printer = freds_printer
;   public = no
;   writable = no
;   printable = yes

# A private directory, usable only by fred. Note that fred requires write
# access to the directory.
;[fredsdir]
;   comment = Fred's Service
;   path = /usr/somewhere/private
;   valid users = fred
```

```
;   public = no
;   writable = yes
;   printable = no

# a service which has a different directory for each machine that connects
# this allows you to tailor configurations to incoming machines. You could
# also use the %U option to tailor it by user name.
# The %m gets replaced with the machine name that is connecting.
;[pchome]
;  comment = PC Directories
;  path = /usr/pc/%m
;  public = no
;  writable = yes

# A publicly accessible directory, read/write to all users. Note that all files
# created in the directory by users will be owned by the default user, so
# any user with access can delete any other user's files. Obviously this
# directory must be writable by the default user. Another user could of course
# be specified, in which case all files would be owned by that user instead.
;[public]
;   path = /usr/somewhere/else/public
;   public = yes
;   only guest = yes
;   writable = yes
;   printable = no

# The following two entries demonstrate how to share a directory so that two
# users can place files there that will be owned by the specific users. In this
# setup, the directory should be writable by both users and should have the
# sticky bit set on it to prevent abuse. Obviously this could be extended to
# as many users as required.
;[myshare]
;   comment = Mary's and Fred's stuff
;   path = /usr/somewhere/shared
;   valid users = mary fred
;   public = no
;   writable = yes
;   printable = no
;   create mask = 0765
```

It maps the AIX folder /var/wingrammar as the root folder /grammars. Any host can access it as root and automatically have write and read permissions.

Important: For security considerations, you must change this configuration. (For detaied information, please refer to http://www.samba.org).

12.17.3 Testing of the installation

To test the installation, do the following:

1. Type the command to launch the Samba server:

   ```
   /usr/local/sbin/smbd
   ```

2. You can access \\9.27.111.193\grammars from Windows 2000.
3. Map it as j:\ in Windows 2000.
4. Create the folder j:\grammars.

Now the connection is established.

D

Default.cff with English and German support

In this appendix we provide the example of default.cff, which can support speech recognition engines and text-to-speech engines of both English and German versions.

Example 12-2 Default.cff with English and German support

```
#--------------------------------------------------------------------------
-
# File Name   : default.sample.cff
# Description : sample configuration file for using IBM WebSphere Voice
Response
#               with DirectTalk Technology for AIX with IBM WebSphere Voice
Server
#               for AIX.
#
# The keywords in this file are documented in the books
# "Application Development using Java and VoiceXML" and
# "WebSphere Voice Server for AIX Use with IBM WebSphere Voice Response for AIX
# "with DirectTalk technology Administrators Guide"
#
# This file defines the sample Weather.VXML application, which is written in
# VoiceXML. You can use this application to verify that you have installed
# and set up WebSphere Voice Server and the WebSphere Voice Response
text-to-speech
# or speech recognition client successfully.
# The application allows you to select a language by pressing a DTMF key
# and then presents different options that test:
# 1. No text-to-speech or speech recognition
# 2. Text-to-speech only
# 3. Speech recognition only
# 4. Both text-to-speech and speech recognition
#--------------------------------------------------------------------------
-
# 5724-C35
# (C) Copyright IBM Corp. 1998, 2002
#--------------------------------------------------------------------------
-
#--------------------------------------------------------------------------
-
# This is the definition of the plugin used for playing cached prerecorded
# audio in VoiceXML applications. DO NOT REMOVE THIS DEFINITION.
#--------------------------------------------------------------------------
-
TTSService=VXML
  PlugInClass=com.ibm.speech.VXML.DTPlayURLPlugIn
  TTSType=VXMLTTS
;
#--------------------------------------------------------------------------
-
# This is the definition of the German text-to-speech service,
# using WebSphere Voice Server text-to-speech.
#
# EngineName=    refers to the engine type as defined in the VVTdefaults file
```

```
#                on the text-to-speech server.
# EngineFree=    setting this to true means that the TTS engine will be freed
#                after each use, rather than at the end of the call.
#                (Optional, default is false).
#-------------------------------------------------------------------------------
-
TTSService=WVS_TTSde_DE
 PluginClass=com.ibm.telephony.directtalk.TTSVVAIX
 TTSType=TTS
 InitSessionString=EngineName=tts, EngineFree=true
;
#-------------------------------------------------------------------------------
# This is the definition of the U.S. English text-to-speech service,
# using WebSphere Voice Server text-to-speech.
#
# EngineName=    refers to the engine type as defined in the VVTdefaults file
#                on the text-to-speech server.
# EngineFree=    setting this to true means that the TTS engine will be freed
#                after each use, rather than at the end of the call.
#                (Optional, default is false).
#-------------------------------------------------------------------------------
-
TTSService=WVS_TTSen_US
 PluginClass=com.ibm.telephony.directtalk.TTSVVAIX
 TTSType=TTS
 InitSessionString=EngineName=tts
;
#-------------------------------------------------------------------------------
-
# This is the definition of the German speech recognition service,
# using WebSphere Voice Server speech recognition.
# InitTechnologyString parameters:
# LocalBaseDir=   Specifies the client-side pathname of your shared file
system.
# ServerBaseDir=  Specifies the server-side pathname of your shared file
system.
# InitSessionString parameters:
# EngineName=
#   Refers to the engine type as defined in the VVTdefaults file on the speech
#   recognition server.
# Profiles=
#   Specifies the context profile to use. Specifies the filename of a context
#   profile in the default directory (/usr/lpp/dirTalk/db/viavoice/profiles/),
#   without the .vvc extension. This parameter applies to Java applications
only.
# EngineFree=
#   Setting this to true means that the Reco. engine will be freed after each
use,
#   rather than at the end of the call. (Optional, default is false)
```

```
# BargeInMode=
#   Selects the mechanism for interrupting voice prompts. Use 'BARGEIN' for
hardware
#   speech detection, 'BARGEIN_SPEECH_TECH' to have WVS do speech detection, or
#   'BARGEIN_RECO_COMPLETE' to have WVS do speech detection and only interrupt
the
#   prompt once a recognition result is obtained. (Optional, default is
BARGEIN_SPEECH_TECH)
# GruntLevel=
#   Sets the WVR for AIX system variable SV218 when using BargeInMode=BARGEIN.
#   (Optional, default is -23)
# GruntMinOnTime=
#   Sets the WVR for AIX system variable SV219 when using BargeInMode=BARGEIN.
#   (Optional, default is 100)
# GruntMinOffTime=
#   Sets the WVR for AIX system variable SV220 when using BargineMode=BARGEIN.
#   (Optional, default is 0)
# VXMLLogProblemWords=
#   Turns on reporting of problem words. Problem words are words for which the
recognition
#   engine is unable to automatically create baseforms, particularly for
Asia-Pacific
#   languages. Set to 'true' to enable. Problem words will be written in UTF-8
to
#   /var/dirTalk/DTBE/dtj_logs/ProblemWords.txt. This setting is intended for
use during
#   application development and its use on a production system is not advised
#   (Optional, default is false)
#-------------------------------------------------------------------------------
-
RecoService=WVS_Recode_DE
 PlugInClass=com.ibm.telephony.directtalk.RecoVVAIX
 InitSessionString=EngineName=asrde_DE, Profiles=myprofile, EngineFree=true,
BargeInMode=BARGEIN, GruntLevel=-10, GruntMinOnTime=60, GruntMinOffTime=5
 InitTechnologyString=LocalBaseDir=/var/vvt, ServerBaseDir=/var/vvt
 RecoType=Recode_DE
;
#-------------------------------------------------------------------------------
-
# This is the definition of the U.K. English speech recognition service,
# using WebSphere Voice Server speech recognition.
# InitTechnologyString parameters:
# LocalBaseDir=   specifies the client-side pathname of your shared file
system.
# ServerBaseDir=  specifies the server-side pathname of your shared file
system.
# InitSessionString parameters:
# EngineName=
#   Refers to the engine type as defined in the VVTdefaults file on the speech
```

```
#   recognition server.
# Profiles=
#   Specifies the context profile to use. Specifies the filename of a context
#   profile in the default directory (/usr/lpp/dirTalk/db/viavoice/profiles/),
#   without the .vvc extension. This parameter applies to Java applications
only.
# EngineFree=
#   Setting this to true means that the Reco. engine will be freed after each
use,
#   rather than at the end of the call. (Optional, default is false)
# BargeInMode=
#   Selects the mechanism for interrupting voice prompts. Use 'BARGEIN' for
hardware
#   speech detection, 'BARGEIN_SPEECH_TECH' to have WVS do speech detection, or
#   'BARGEIN_RECO_COMPLETE' to have WVS do speech detection and only interrupt
the
#   prompt once a recognition result is obtained. (Optional, default is
BARGEIN_SPEECH_TECH)
# GruntLevel=
#   Sets the WVR for AIX system variable SV218 when using BargeInMode=BARGEIN.
#   (Optional, default is -23)
# GruntMinOnTime=
#   Sets the WVR for AIX system variable SV219 when using BargeInMode=BARGEIN.
#   (Optional, default is 100)
# GruntMinOffTime=
#   Sets the WVR for AIX system variable SV220 when using BargineMode=BARGEIN.
#   (Optional, default is 0)
# VXMLLogProblemWords=
#   Turns on reporting of problem words. Problem words are words for which the
recognition
#   engine is unable to automatically create baseforms, particularly for
Asia-Pacific
#   languages. Set to 'true' to enable. Problem words will be written in UTF-8
to
#   /var/dirTalk/DTBE/dtj_logs/ProblemWords.txt. This setting is intended for
use during
#   application development and its use on a production system is not advised
#   (Optional, default is false)
#-------------------------------------------------------------------------------
-
RecoService=WVS_Recoen_US
 PlugInClass=com.ibm.telephony.directtalk.RecoVVAIX
 InitSessionString=EngineName=asren_US, Profiles=myprofile
 InitTechnologyString=LocalBaseDir=/var/vvt, ServerBaseDir=/var/vvt
 RecoType=Recoen_US
;

#-------------------------------------------------------------------------------
-
```

```
# This is the application definition for the sample Weather application.
# The VoiceXML file is fetched from the local filesystem.
#-------------------------------------------------------------------------------
-
AppName=weather
  Enabled=yes
  Parameter=URI, file:///var/dirTalk/DTBE/samples/Weather.VXML
  AppClass=com.ibm.speech.VXML.DTVoicelet
;
#-------------------------------------------------------------------------------
-
# This is the application definition for the Menu sample Java application.
#-------------------------------------------------------------------------------
-
AppName=menu
  Enabled=yes
  AppClass=com.ibm.telephony.directtalk.samples.DTDemoV
;

#-------------------------------------------------------------------------------
-
# This is a group definition. Applications listed here are started when
# the node is started, provided that the group is specified in the Group=
# keyword in the NodeName definition.
# The weather application is commented out because it does not need to be
# started, provided that it is the NodeDefApp= application for the node.
#-------------------------------------------------------------------------------
-
#GroupName=group1
#  Enabled=yes
#  Application=weather
#  Application=menu
#;
#-------------------------------------------------------------------------------
-
# This is the definition of the voice response node on LocalHost.
# VRNode=          specifies that this is a voice response node.
# NodeDefLocale=   specifies the locale which will be used for the weather
#                  application and any other applications that do not specify
#                  their own locale.
#                  Change this to the locale you want to use.
# NodeDefAppName=  specifies the default application: currently the weather
#                  application.
# NodeClassPath=   currently commented out.  Use this to specify the additional
#                  classpath for Java applications in the Voice Response Node.
# Group=   specifies groups of applications to be started at node
#                  startup (refers to GroupName entry in this file).
# NumToApp=        currently commented out.  Use this to specify number to
#                  dial for specific application (refers to AppName entry
```

```
#                  in this file).
# TTSService=      refers to TTSService entry in this file
# TTSDefinition=   specifies the TTSService to be used for a specific locale;
#                  refers to TTSType keyword in a TTSService entry in this
#                  file.
# RecoService=     refers to RecoService entry in this file
# RecoDefinition=  specifies the RecoService to be used for a specific locale;
#                  refers to RecoType keyword in a RecoService entry in this
#                  file.
#-------------------------------------------------------------------------------
-
NodeName=Node1
  Enabled=yes
  VRNode=yes
  NodeDefLocale=en_US
# NodeDefAppName=menu
  NodeDefAppName=weather
#  NodeDefAppName=Test
# NodeClasspath=
#  Group=group1
# NumToApp=nnnnnn,menu
# NumToApp=nnnnnn,weather
  TTSService=VXML
  TTSDefinition=vo_IC_EXML,VXMLTTS
  TTSService=WVS_TTSde_DE
#  TTSService=WVS_TTSen_GB
#  TTSService=WVS_TTSja_JP
#  TTSService=WVS_TTSzh_CN
#  TTSService=WVS_TTSfr_CA
#  TTSService=WVS_TTSpt_BR
  TTSService=WVS_TTSen_US
#  TTSService=WVS_TTSes_ES
#  TTSService=WVS_TTSfr_FR
#  TTSService=WVS_TTSit_IT
  TTSDefinition=de_DE,TTS
#  TTSDefinition=en_GB,TTSen_GB
#  TTSDefinition=ja_JP,TTSja_JP
#  TTSDefinition=zh_CN,TTSzh_CN
#  TTSDefinition=fr_CA,TTSfr_CA
#  TTSDefinition=pt_BR,TTSpt_BR
  TTSDefinition=en_US,TTS
#  TTSDefinition=es_ES,TTSes_ES
#  TTSDefinition=fr_FR,TTSfr_FR
#  TTSDefinition=de_DE,TTSit_IT
  RecoService=WVS_Recode_DE
#  RecoService=WVS_Recoen_GB
#  RecoService=WVS_Recoja_JP
#  RecoService=WVS_Recozh_CN
#  RecoService=WVS_Recofr_CA
```

```
#  RecoService=WVS_Recopt_BR
  RecoService=WVS_Recoen_US
#  RecoService=WVS_Recoes_ES
#  RecoService=WVS_Recofr_FR
#  RecoService=WVS_Recoit_IT
  RecoDefinition=de_DE,Recode_DE
#  RecoDefinition=en_GB,Recoen_GB
#  RecoDefinition=ja_JP,Recoja_JP
#  RecoDefinition=zh_CN,Recozh_CN
#  RecoDefinition=fr_CA,Recofr_CA
#  RecoDefinition=pt_BR,Recopt_BR
  RecoDefinition=en_US,Recoen_US
#  RecoDefinition=es_ES,Recoes_ES
#  RecoDefinition=fr_FR,Recofr_FR
#  RecoDefinition=it_IT,Recoit_IT
;
#-------------------------------------------------------------------------------
-
# This is the definition of the LocalHost.
#-------------------------------------------------------------------------------
-
HostName=LocalHost
  Enabled=yes
  IPName=127.0.0.1
  Node=Node1
;
```

E

Default.cff with distributed system support

In this appendix we provide the example of default.cff, which can support the distributed environment of Windows 2000 and AIX.

Example 12-3 Default.cff with distributed system support

```
#--------------------------------------------------------------------------
-
# File Name   : default.sample.cff
# Description : sample configuration file for using IBM WebSphere Voice
Response
#               with DirectTalk Technology for AIX with IBM WebSphere Voice
Server
#               for AIX.
#
# The keywords in this file are documented in the books
# "Application Development using Java and VoiceXML" and
# "WebSphere Voice Server for AIX Use with IBM WebSphere Voice Response for AIX
# "with DirectTalk technology Administrators Guide"
#
# This file defines the sample Weather.VXML application, which is written in
# VoiceXML. You can use this application to verify that you have installed
# and set up WebSphere Voice Server and the WebSphere Voice Response
text-to-speech
# or speech recognition client successfully.
# The application allows you to select a language by pressing a DTMF key
# and then presents different options that test:
# 1. No text-to-speech or speech recognition
# 2. Text-to-speech only
# 3. Speech recognition only
# 4. Both text-to-speech and speech recognition
#--------------------------------------------------------------------------
-
# 5724-C35
# (C) Copyright IBM Corp. 1998, 2002
#--------------------------------------------------------------------------
-
#--------------------------------------------------------------------------
-
# This is the definition of the plugin used for playing cached prerecorded
# audio in VoiceXML applications. DO NOT REMOVE THIS DEFINITION.
#--------------------------------------------------------------------------
-
TTSService=VXML
  PlugInClass=com.ibm.speech.VXML.DTPlayURLPlugIn
  TTSType=VXMLTTS
;


#--------------------------------------------------------------------------
-
# This is the definition of the U.S. English text-to-speech service,
# using WebSphere Voice Server text-to-speech.
```

```
#
# EngineName=    refers to the engine type as defined in the VVTdefaults file
#                on the text-to-speech server.
# EngineFree=    setting this to true means that the TTS engine will be freed
#                after each use, rather than at the end of the call.
#                (Optional, default is false).
#-------------------------------------------------------------------------
-
TTSService=WVS_TTSen_US
 PluginClass=com.ibm.telephony.directtalk.TTSVVAIX
 TTSType=TTSen_US
 InitSessionString=EngineName=ttsen_US
;

#-------------------------------------------------------------------------
-
# This is the definition of the U.K. English speech recognition service,
# using WebSphere Voice Server speech recognition.
# InitTechnologyString parameters:
# LocalBaseDir=   specifies the client-side pathname of your shared file
system.
# ServerBaseDir=  specifies the server-side pathname of your shared file
system.
# InitSessionString parameters:
# EngineName=
#   Refers to the engine type as defined in the VVTdefaults file on the speech
#   recognition server.
# Profiles=
#   Specifies the context profile to use. Specifies the filename of a context
#   profile in the default directory (/usr/lpp/dirTalk/db/viavoice/profiles/),
#   without the .vvc extension. This parameter applies to Java applications
only.
# EngineFree=
#   Setting this to true means that the Reco. engine will be freed after each
use,
#   rather than at the end of the call. (Optional, default is false)
# BargeInMode=
#   Selects the mechanism for interrupting voice prompts. Use 'BARGEIN' for
hardware
#   speech detection, 'BARGEIN_SPEECH_TECH' to have WVS do speech detection, or
#   'BARGEIN_RECO_COMPLETE' to have WVS do speech detection and only interrupt
the
#   prompt once a recognition result is obtained. (Optional, default is
BARGEIN_SPEECH_TECH)
# GruntLevel=
#   Sets the WVR for AIX system variable SV218 when using BargeInMode=BARGEIN.
#   (Optional, default is -23)
# GruntMinOnTime=
#   Sets the WVR for AIX system variable SV219 when using BargeInMode=BARGEIN.
```

```
#   (Optional, default is 100)
# GruntMinOffTime=
#   Sets the WVR for AIX system variable SV220 when using BargineMode=BARGEIN.
#   (Optional, default is 0)
# VXMLLogProblemWords=
#   Turns on reporting of problem words. Problem words are words for which the
recognition
#   engine is unable to automatically create baseforms, particularly for
Asia-Pacific
#   languages. Set to 'true' to enable. Problem words will be written in UTF-8
to
#   /var/dirTalk/DTBE/dtj_logs/ProblemWords.txt. This setting is intended for
use during
#   application development and its use on a production system is not advised
#   (Optional, default is false)
#-------------------------------------------------------------------------------
-
RecoService=WVS_Recoen_US
 PlugInClass=com.ibm.telephony.directtalk.RecoVVAIX
 InitSessionString=EngineName=asren_US, Profiles=myprofile
 InitTechnologyString=LocalBaseDir=/var/wingrammar/grammars,
ServerBaseDir=j:/grammars
 RecoType=Recoen_US
;
#-------------------------------------------------------------------------------
-
# This is the application definition for the sample Weather application.
# The VoiceXML file is fetched from the local filesystem.
#-------------------------------------------------------------------------------
-
AppName=weather
  Enabled=yes
  Parameter=URI, file:///var/dirTalk/DTBE/samples/Weather.VXML
  AppClass=com.ibm.speech.VXML.DTVoicelet
;
#-------------------------------------------------------------------------------
-
# This is the application definition for the Menu sample Java application.
#-------------------------------------------------------------------------------
-
AppName=menu
  Enabled=yes
  AppClass=com.ibm.telephony.directtalk.samples.DTDemoV
;
AppName=Test
  Enabled=yes
  Parameter=URI, file:///var/dirTalk/DTBE/samples/Test.VXML
  AppClass=com.ibm.speech.VXML.DTVoicelet
;
```

```
#-------------------------------------------------------------------------
-
# This is a group definition. Applications listed here are started when
# the node is started, provided that the group is specified in the Group=
# keyword in the NodeName definition.
# The weather application is commented out because it does not need to be
# started, provided that it is the NodeDefApp= application for the node.
#-------------------------------------------------------------------------
-
#GroupName=group1
#  Enabled=yes
#  Application=weather
#  Application=menu
#;
#-------------------------------------------------------------------------
-
# This is the definition of the voice response node on LocalHost.
# VRNode=          specifies that this is a voice response node.
# NodeDefLocale=   specifies the locale which will be used for the weather
#                  application and any other applications that do not specify
#                  their own locale.
#                  Change this to the locale you want to use.
# NodeDefAppName=  specifies the default application: currently the weather
#                  application.
# NodeClassPath=   currently commented out.  Use this to specify the additional
#                  classpath for Java applications in the Voice Response Node.
# Group=   specifies groups of applications to be started at node
#                  startup (refers to GroupName entry in this file).
# NumToApp=        currently commented out.  Use this to specify number to
#                  dial for specific application (refers to AppName entry
#                  in this file).
# TTSService=      refers to TTSService entry in this file
# TTSDefinition=   specifies the TTSService to be used for a specific locale;
#                  refers to TTSType keyword in a TTSService entry in this
#                  file.
# RecoService=     refers to RecoService entry in this file
# RecoDefinition=  specifies the RecoService to be used for a specific locale;
#                  refers to RecoType keyword in a RecoService entry in this
#                  file.
#-------------------------------------------------------------------------
-
NodeName=Node1
  Enabled=yes
  VRNode=yes
  NodeDefLocale=en_US
# NodeDefAppName=menu
  NodeDefAppName=weather
#  NodeDefAppName=Test
# NodeClasspath=
```

```
#  Group=group1
# NumToApp=nnnnnn,menu
# NumToApp=nnnnnn,weather
  TTSService=VXML
  TTSDefinition=vo_IC_EXML,VXMLTTS
#  TTSService=WVS_TTSde_DE
#  TTSService=WVS_TTSen_GB
#  TTSService=WVS_TTSja_JP
#  TTSService=WVS_TTSzh_CN
#  TTSService=WVS_TTSfr_CA
#  TTSService=WVS_TTSpt_BR
  TTSService=WVS_TTSen_US
#  TTSService=WVS_TTSes_ES
#  TTSService=WVS_TTSfr_FR
#  TTSService=WVS_TTSit_IT
#  TTSDefinition=de_DE,TTS
#  TTSDefinition=en_GB,TTSen_GB
#  TTSDefinition=ja_JP,TTSja_JP
#  TTSDefinition=zh_CN,TTSzh_CN
#  TTSDefinition=fr_CA,TTSfr_CA
#  TTSDefinition=pt_BR,TTSpt_BR
  TTSDefinition=en_US,TTSen_US
#  TTSDefinition=es_ES,TTSes_ES
#  TTSDefinition=fr_FR,TTSfr_FR
#  TTSDefinition=de_DE,TTSit_IT
#  RecoService=WVS_Recode_DE
#  RecoService=WVS_Recoen_GB
#  RecoService=WVS_Recoja_JP
#  RecoService=WVS_Recozh_CN
#  RecoService=WVS_Recofr_CA
#  RecoService=WVS_Recopt_BR
  RecoService=WVS_Recoen_US
#  RecoService=WVS_Recoes_ES
#  RecoService=WVS_Recofr_FR
#  RecoService=WVS_Recoit_IT
#  RecoDefinition=de_DE,Recode_DE
#  RecoDefinition=en_GB,Recoen_GB
#  RecoDefinition=ja_JP,Recoja_JP
#  RecoDefinition=zh_CN,Recozh_CN
#  RecoDefinition=fr_CA,Recofr_CA
#  RecoDefinition=pt_BR,Recopt_BR
  RecoDefinition=en_US,Recoen_US
#  RecoDefinition=es_ES,Recoes_ES
#  RecoDefinition=fr_FR,Recofr_FR
#  RecoDefinition=it_IT,Recoit_IT
;
#-------------------------------------------------------------------------------
-
# This is the definition of the LocalHost.
```

```
#-------------------------------------------------------------------------
-
HostName=LocalHost
  Enabled=yes
  IPName=127.0.0.1
  Node=Node1
;
HostName=m23wph67
  Enabled=yes
  IPName=9.24.104.191
  Node=Node1
;
```

Related publications

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

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 571.

- *Using LDAP for Directory Integration A Look at IBM SecureWay Directory, Active Directory and Domino*, SG24-6163
- *IBM WebSphere V4.0 Advanced Edition Handbook,* SG24-6176
- *Access Integration Pattern Using IBM WebSphere Portal Server*, SG24-6267
- *IBM WebSphere Portal V4.1 Handbook Volume 1,* SG24-6883
- *IBM WebSphere Portal V4.1 Handbook Volume 2,* SG24-6920
- *IBM WebSphere Portal V4.1 Handbook Volume 3,* SG24-6921
- *IBM WebSphere Voice Server 2.0,* SG24-6537
- *New Capabilities in IBM WebSphere Transcoding Publisher Version 3.5 Extending Web Applications to the Pervasive World,* SG24-6233

Other resources

These publications are also relevant as further information sources:

- *DirectTalk Application Development Using Java and Voice XML, Version 2.3*, GC34-6049
- *WebSphere Voice Server Version 2.0 - Use with DirectTalk for AIX Telephony Platform Administrator's Guide*, G210-1259
- *WebSphere Voice Server Version 2.0 - Use with DirectTalk for AIX Telephony Platform Application Development with State Tables*, G210-1260
- *WebSphere Voice Server Version 2.0 Software Developers Kit (SDK) VoiceXML Programmer's Guide*, G210-1261
- *WebSphere Voice Server Version 2.0 - Use with Cisco Telephony Platform Administrator's Guide*, G210-1262

- *WebSphere Voice Server Version 2.0 - Use with Intel Dialogic Telephony Platform Administrator's Guide*, G210-1263
- *WebSphere Voice Server Version 2.0 Speech Technologies Administrator's Guide*, G210-1264
- *WebSphere Voice Server Version 2.0 Speech Technologies API Reference*, G210-1265

To obtain these publications, please refer to the following URL:
http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi

Referenced Web sites

These Web sites are also relevant as further information sources:

- http://www.ibm.com/software/speech/support
- http://developer.intel.com/design/telecom/support/
- http://www.cisco.com/warp/public/cc/pd/rt/2600/prodlit/2600_ds.htm
- http://www.cisco.com/univercd/cc/td/doc/pcat/dit1e1p1.htm
- http://www.cisco.com/warp/public/788/products/hdv_netmod.html
- http://www.ibm.com/software/data/db2/udb/downloads.html
- http://www-4.ibm.com/software/speech/enterprise/wvs-rdc.html
- http://www-4.ibm.com/software/webservers/voiceserver/library.html
- http://www.voicexml.org
- http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html
- http://www.ecma.ch/ecma1/stand/ECMA-262.htm
- http://w3.hursley.ibm.com/~walterja/Books/Java/html/Application_Development_Using_Java_&_VoiceXML/wvraj017.htm
- http://www.intel.com/network/csp/trans/dialogic.htm
- http://w3.hursley.ibm.com/~walterja/Books/Windows/html/gip/dtgim054.htmD/32
- http://www.aculab.com/company_main/company_main.htm
- http://www2.ibmlink.ibm.com/cgi-bin/master?xh=TsNuzMkJtIp2*G2USenGnN9332&request=announcements&parms=H%5F202%2D224&xfr=N
- http://www.voicexml.org

- http://www.microsoft.com/windows/netmeeting/download/

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

D

E

F

G

H

I

J

L

M

N

P

Q

R

S

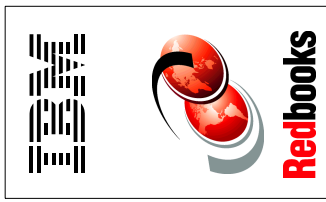
T

U

V

W

X



IBM WebSphere Voice Systems Solutions

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM WebSphere Voice Systems Solutions Implementation Guide

Covers features and functionality of WebSphere Voice Server with WebSphere Voice Response V3.1

Includes scenarios of Dialogic, Cisco and speech technologies

Highlights application development and SDK 3.1

The WebSphere Voice Server product is a member of the IBM WebSphere software family. It provides a platform that enables the creation of voice applications through industry standards such as VoiceXML and Java. The WebSphere Voice Server facilitates the deployment of voice applications by interfacing with voice standards such as Cisco VoIP, IBM DirectTalk, and Dialogic platforms. It further aids the development of these applications by providing development tools.

This IBM Redbook discusses the functionality of WebSphere Voice Server in the context of real business environments. We introduce the voice environment and the WebSphere Voice Server. We cover in great detail the various operating platforms supported by WebSphere Voice Server (Dialogic, Cisco, and DirectTalk - also referred to as WebSphere Voice Response). The product has been further enhanced to function on both Intel and AIX systems. The Redbook goes into some depth about this new functionality.

Additionally, we discuss both the Software Developers Kit (SDK) and the Voice Toolkit that are available for the development of voice applications. A step-by-step approach was taken to walk through the development of a VoiceXML application utilizing both of these tools, and taking advantage of the new functions provided within them.

SG24-6884-00 ISBN 073842773X

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