INTERACTIVE VOICE RESPONSE SYSTEM
FOR
GAS BOOKING

by

SELVAM A

Register Number - 71206621048

of

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree
of

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY

CHENNAI 600 025

July, 2009

BONAFIDE CERTIFICATE

Certified that this project report titled **Interactive Voice Response System for Gas Booking** is the bonafide work of **Mr. SELVAM .A** (Register Number: 71206621048) who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

N. Jayanthan
Supervisor

Head of the Department

Submitted to Project and Viva Examination held on 6/07/09

6/7/09
Internal Examiner

6/7/09
External Examiner

ACKNOWLEDGEMENT

I express my grateful thanks to our beloved vice principal, **Dr. R.Annamalai** of Kumaraguru College of Technology, Coimbatore for giving me an opportunity to take up this project.

I express my deep sense of gratitude to **Dr. S. Thangasamy**, Professor and Dean of Computer Science and Engineering department and **Dr. M. Gururajan,** Professor and Head of Department of Computer Applications for extending their help in providing all the facilities at college for the successful completion of the project.

I would like to express my deepest and sincere gratitude to **Mr. N.Jayakaanthan, MCA.,M.Phil** Lecturer, Department of Computer Applications for his guidance, support, valuable suggestions and tremendous encouragement during the course of this project and I also thank **Ms. V. Geetha, MCA.,M.Phil** Assistant Professor and project-coordinator of Master of Computer Applications.

BARCiNDiA.NET

Bannari Amman Research Consultants
Coimbatore - 641 015.

Date: 04.06.2009

CERTIFICATE

This is to certify that, **Mr. A.Selvam, III M.C.A, Reg No: 71206621048** Student of **Kumaraguru College of Technology, Coimbatore** has successfully completed his project work, titled **IVRS – Interactive Voice Response System for Gas Booking System** as part of his course curriculum.

He has done the project using **.NET** during the period of **18-12-2008 to 03-06-2009.** He has completed the assigned project well within the time frame. He was sincere, hardworking and his conduct during the project was commendable.

We wish him all the best in his future endeavors.

For Bannari Amman Research Consultants

(Partner)

(V.Govindaraju)

ABSTRACT

Interactive Voice Response System for Gas Booking is an application that facilitates the gas booking process followed in a gas agency. IVR systems have invaded most fields that depend on customer interaction. They have become essential to ensure optimal efficiency. As contact for our customers is essential, our business image is dependent on how we manage our clients when they call us. Consider a customer interacting with the gas agency for gas booking. The system uses a simple system-initiative dialog structure to guide the caller through few basic choices. The caller must provide a consumer number to place the order. The system gathers the information that the caller supplies, repeats it back to the caller, and requests confirmation from the caller.

The system development is entirely based on Microsoft technologies mainly, Microsoft Office Communication server 2007 speech server, Microsoft ASP.NET, Microsoft Visual Studio 2005 with service pack1 and SQL Server 2005 Reporting Services. Speech Server is an IVR platform that uses Microsoft speech technology and combines Web technologies, speech-processing services, and telephony capabilities in an integrated system. Using Speech Server would reduce development costs, provide a more interactive customer service experience.

An IVR system can easily handle more than one call simultaneously. The system is easy to operate with industry-standard Microsoft tools and a graphical user interface for all IVR maintenance tasks. The system has powerful reporting tools and statistic gathering capabilities. The system can be operated 24/7. Text-to-speech technology is used to read text to callers therefore substantially reducing system maintenance costs. This capability greatly enhances any company's position, particularly with the advances in hand-phones and the inevitable growth of wireless communications for the general consumer. All these advantages can hike customer satisfaction. Also an IVRS is less expensive than live agents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACRONYM	FULL FORM
IVR	Interactive Voice Response
SALT	Speech Application Language Tags
PSTN	Public Switched Telephone Network
PEML	Prompt Engine Markup Language
SES	Speech Engine Services
SML	Semantic Markup Language
PBX	Private Branch Exchange
CLR	Common Language Runtime
XML	Extensible Markup Language
DTMF	Dual-Tone Multi-Frequency
DNIS	Dialed Number Identification Service
ANI	Automatic Number Identification
TTS	Text-to-speech
TAS	Telephony Application Services
SSRS	SQL Server 2005 Reporting Services
SRGS	Speech Recognition Grammar Specification

CHAPTER 1

INTRODUCTION

1.1 COMPANY PROFILE

BARCiNDiA.NET

Bannariamman Research and Consultings (BARCiNDiA), based at Coimbatore, is a part of a group involved in software development & Technology Training. They have good experience in successfully executing projects for all type of customers. BARCiNDiA is a technology oriented company promoted by professionals with rich experience and expertise in the industry. The company is focusing on providing technical training and software development for various applications. BARCiNDiA also looks at providing the necessary business computing systems for optimizing the total systems integration, with the help of its vast product range.

1.2 SYSTEM OVERVIEW

The gas agency wanted to improve the experience of the call-center refill booking line for its customers. In addition to increasing the overall level of call-center efficiency, agency wanted to increase the productivity of call-center agents and reduce support costs. The existing solution, which uses call-center live agents to interact with customers, had many performance issues and was not meeting the business needs. So, a customized IVR application built using Microsoft Office Communications Server 2007 Speech Server, Microsoft ASP.NET, Microsoft SQL Server 2005, and Microsoft Visual Studio 2005 is created to satisfy the business needs. It can enhance organizations efficiency and make customers feel happier.

1.2.1 Work Flow

Consider a customer interacting with the gas agency for refill booking. When the customer dials the number, the system answers the call and plays a greeting and prompts the customer for the necessary information. The application waits for a response for a set period of time. The caller can respond by speaking. The application takes the appropriate action based on the customer's response. For example, it might update information in a database, 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 "Cancel" or "Exit").

1.2.2 User Characteristics

The users are classified into the following groups based on the function:

Developer: Responsible for development of the project and ensures that developing project satisfies the requirements.

Speech Engineer: Responsible for producing high quality prompts.

Database Owner: SQL Server Database administrator access, with rights to create new databases and tables. In other words, full control.

Speech Operator: Has write access to existing tables of the tuning database, for example, import and pruning. In other words, Speech Analyst plus data population and pruning rights.

Speech Analyst: Privileges to read all, but write only to certain extensible fields of the database, for example, the transcription field and annotation field. In other words, Speech RO plus limited write access.

Speech RO: Has read-only access to the tuning database. Appropriate for users who want to perform read-only analysis of speech data and access non-editable reports as well as list and details views.

Advanced users or system administrators: Has all privileges for monitoring the whole system.

End users: Users uses the system with no particular knowledge on computer programming.

Testers: Responsible for testing the functionalities of the project.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In the present system, the consumer comes to the gas agency or he calls on telephone to book the gas. A live agent (gas agency employee) had to answer each consumer's call. When the consumer tells their consumer number, the live agent keys it in the system and books the cylinder manually. The gas is delivered to the consumer when it is available.The agents need to be trained appropriately for the work and perfect service.

2.1.1 Drawbacks

- ✓ Booking can be done only during office hours.
- ✓ Staffs have to receive calls concurrently.
- ✓ Have to book orders manually.
- ✓ As the consumer size grows there is a need for more staff.
- ✓ Decreased employee productivity.

2.2 PROPOSED SYSTEM

The proposed system is an interactive voice response solution that automates the refill booking process in a gas agency. It is developed to overcome all the drawbacks of the existing system. It responds intelligently consumers who are naturally speaking. For most service calls, the live operator (gas agency employee) never had to speak to a consumer. The system uses a simple system-initiative dialog structure to guide the caller through few basic choices. The caller must provide a consumer number to place the order. The system gathers the information that the caller supplies, repeats it back to the caller, and requests confirmation from the caller. If the caller confirms the information, refill booking is done and the call is disconnected.

The system development is entirely based on Microsoft technologies mainly, Microsoft Office Communication server 2007 speech server, Microsoft

ASP.NET, Microsoft Visual Studio 2005 with service pack1 and SQL Server 2005 Reporting Services. Using Speech Server would reduce development costs, provide a more interactive customer service experience. It combines Web technologies, speech-processing services, and telephony capabilities into a single, integrated system. Microsoft Speech Server is a flexible and integrated speech platform that dramatically reduces the complexity and cost of developing and deploying speech applications.

This system also uses SQL server 2005 Reporting Services. Microsoft SQL Server 2005 Reporting Services is a server-based solution for building enterprise reports that draw content from a variety of relational and multidimensional data sources, publishing reports that can be viewed in various formats, and centrally managing security and subscriptions. The reports that we create can be viewed over a Web-based connection or as part of a Microsoft Windows application or SharePoint portal. The system has powerful reporting tools and statistic gathering capabilities.

2.2.1 Advantages

✓ Automated services

✓ 24 hour operations

✓ Consistent quality voice responses

✓ Increase customer service levels

✓ Extend customer service hours

✓ Improve information dissemination

✓ Increase employee productivity

✓ Reduce operation cost

CHAPTER 3

SYSTEM CONFIGURATION

3.1 SYSTEM REQUIREMENTS

The minimum requirements for the system to run effectively are:

3.1.1 Hardware Requirements

Processor	: x86 processor with clock speed of 3.0 GHz
Main Memory	: 1 GB
Secondary Memory	: 5 GB
Video	: Super VGA (800 x 600) or higher resolution adapter and monitor
Voice over IP (VoIP) Gateway	: Required to connect to a PSTN, (unless using a legacy telephony board)
Input Unit	: Standard Mouse and Keyboard
Microphone	: Optimal performance is achieved with a high quality close-talk (headset) microphone. A universal serial bus (USB) microphone with gain adjustment support is recommended.

3.1.2 Software Requirements

Operating system	: Microsoft Windows XP Professional SP3 (x86 or x64), Microsoft Windows Server 2003 Standard Edition R2 (x86 or x64)
Server	: Microsoft Office Communication Server 2007 Speech Server
Framework	: .NET Framework 3.0
Development tool	: Visual Studio 2005 SP1 with Extensions for Windows Workflow Foundation
Database	: SQL Server 2005 standard edition

3.2 SOFTWARE TECHNOLOGIES

3.2.1 Microsoft ASP.NET

ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites, web applications and web services. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language. .NET pages, known officially as "web forms", are the main building block for application development. Web forms are contained in files with an ".aspx" extension; in programming jargon, these files typically contain static (X)HTML markup, as well as markup defining server-side Web Controls and User Controls where the developers place all the required static and dynamic content for the web page.

3.2.2 Interactive Voice Response (IVR)

Interactive Voice Response (IVR) is an interactive technology that allows a computer to detect voice and keypad inputs. IVR technology is used extensively in telecommunications, but is also being introduced into automobile systems for hands-free operation. Current deployment in automobiles revolves around satellite navigation, audio and mobile phone systems. In telecommunications, IVR allows customers to access a company's database via a telephone touchtone keypad or by speech recognition, after which they can service their own enquiries by following the instructions. IVR systems can respond with pre-recorded or dynamically generated audio to further direct users on how to proceed. IVR systems can be used to control almost any function where the interface can be broken down into a series of simple menu choices. In telecommunications applications, such as customer support lines, IVR systems generally scale well to handle large call volumes. IVR systems are typically used to service high call volumes, reduce cost and improve the customer experience.

3.2.3 Voice Response Application

Voice response applications are those in which input and output interactions come through a spoken user interface, instead of a graphical one. 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 as input. Such applications are capable of exploiting an enterprise architecture comprising servlets, ASPs, JSPs, and JavaBeans. We can integrate a Web application with voice using VoiceXML, Java beans, or Websphere Voice Response state tables. Voice applications can reside on local or remote systems, and customers can access them from a telephony-capable device such as traditional wired phones, cordless phones, and mobile phones. A typical voice application call flow consists of the following steps:

1. The customer calls the contact center and the system answers the call and invokes the appropriate application, such as the one referenced by Dialed Number Identification Service (DNIS) often referred to as the "called number" or Automatic Number Identification (ANI) often called as "calling number".

2. The application plays a greeting and prompts the customer for the necessary information.

3. The application waits for a response for a set period of time. The caller can respond by speaking or by pressing keys on a telephone keypad.

4. 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.

5. 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").

3.2.4 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, the user can simply talk to provide input. Speech recognition translates speech data into a format that voice applications can understand and process.

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). When the most likely match is identified, the engine returns it as a text string.

3.2.5 VoiceXML

VoiceXML is an XML-based, industry-standard language for creating voice applications, much as HTML is a language for developing visual applications. Using VoiceXML, the Voice Extensible Markup Language, we can create audio dialogs that feature synthesized speech, digitized audio, recognition of spoken, DTMF key input, recording of spoken input, telephony, and mixed initiative conversations. 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 through 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

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.

3.2.6 Text-to-speech

Text-to-speech (TTS) 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 can consist of text and, optionally, special tags (annotations) that can change the sound of the voice that is ultimately produced.

The TTS engine analyzes 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. After 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. The output produced is highly intelligible, although it may sound robotic or mechanical to some listeners.

3.2.7 Speech Application Language Tags (SALT)

Speech Application Language Tags (SALT) is an XML based markup language that is used in HTML and XHTML pages to add voice recognition capabilities to web based applications. Speech Application Language Tags enables multimodal and telephony-enabled access to information, applications, and Web services from PCs, telephones, tablet PCs, and wireless personal digital assistants (PDAs). The Speech Application Language Tags extend existing mark-up languages such as HTML, XHTML, and XML. Multimodal access will enable users to interact with an application in a variety of ways: they will be able to input data using speech, a keyboard, keypad, mouse and/or stylus, and produce data as synthesized speech, audio, plain text, motion video, and/or graphics.

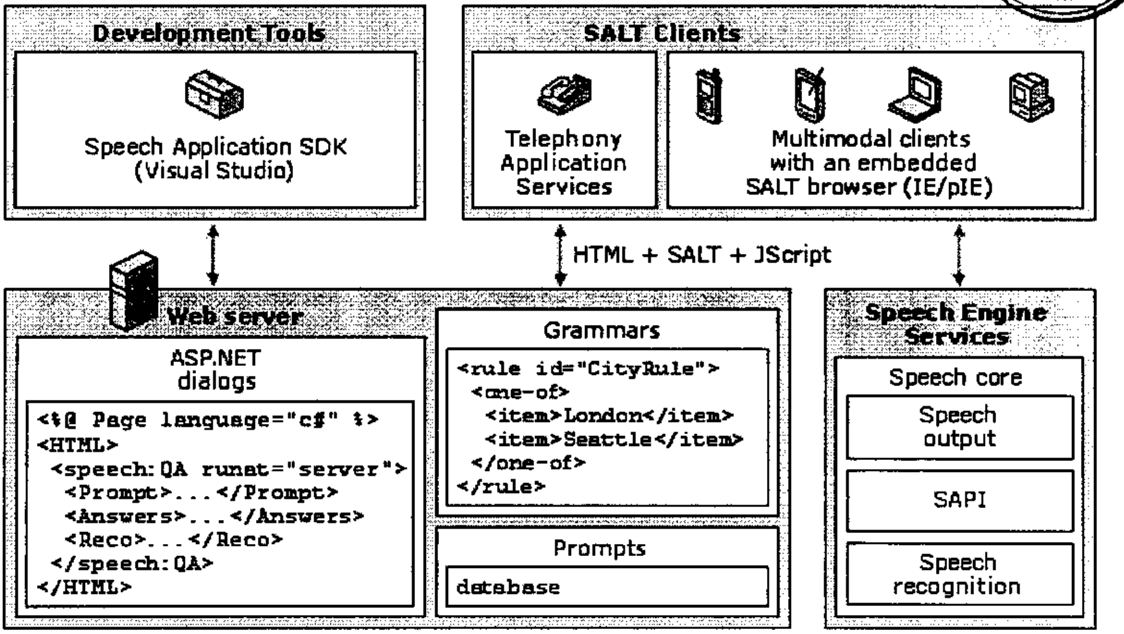
3.2.8 Dialog Speech Control

Dialog Speech Controls is a set of ASP.NET Web controls designed for building speech-enabled Web applications using Speech Application Language Tags (SALT) technology. Dialog Speech Controls technology requires a client device with a browser and scripting capabilities. A graphical user interface (GUI) in the browser is not required. When the user connects to an ASP.NET Web page containing Dialog Speech Controls, ASP.NET renders the page and sends it to the client browser. The user interacts with the rendered controls in the browser, and does not see the Web page. The rendered page contains information about the controls on the page and about the data to be collected. A client-side script called RunSpeech obtains this information and activates controls to collect the specified data and to confirm that it is correct.

3.2.9 Microsoft Office Communications Server 2007 Speech Server

Microsoft Office Communications Server 2007 Speech Server is a comprehensive IVR platform for powerful speech-enabled self-service applications. The platform consists of a server with built-in speech recognition and speech synthesis software that can be used for speech-based solutions, plus a most comprehensive speech application toolkit available in the industry. Speech Server is an open, software-only platform that supports .NET programming languages and voice Web standards like VoiceXML and SALT, and it leverages Visual Studio 2005, which is familiar to one of the largest developer communities in the world. It is based on the Microsoft future-ready VoIP foundation that supports the convergence of the data and telephony worlds. Part of the strong Microsoft unified communications portfolio of products, this security-enhanced, scalable, robust, and proven platform is widely deployed across many leading enterprises that span many different applications. Microsoft Speech Server is composed of the following components:

- **Speech Engine Services (SES)** provides speech recognition and speech output for voice-only applications and also for Pocket PC and Smartphone devices using multimodal speech applications. SES includes the fo.lowing sub-components:
  - ✓ Enterprise-grade Automatic Speech Recognition
  - ✓ World-class Text-to-Speech (TTS) engine
  - ✓ Prompt Database Engine — manages recorded and TTS prompts
- **Telephony Application Services (TAS)** provides the communication link between the telephone system, Speech Engine Services and the Web server; SALT Interpreters interpret SALT tags in standard Web pages that define the telephony application. TAS includes the following subcomponents:
  - ✓ SALT Interpreter — for interpretation of embedded SALT speech tags
  - ✓ Media and Speech Manager — manages audio and speech input/output



```
<%@ Page language="c#" %>
<HTML>
 <speech:QA runat="server">
  <Prompt>...</Prompt>
  <Answers>...</Answers>
  <Reco>...</Reco>
 </speech:QA>
</HTML>
```

```
<rule id="CityRule">
 <one-of>
  <item>London</item>
  <item>Seattle</item>
 </one-of>
</rule>
```

Fig. 3.2.9.1 Speech Server Architecture

Speech Server is the only platform that does not charge a premium for deploying speech technology, significantly lowering the barrier to entry for organizations that want to use speech technology in their business solutions. It offers an unparalleled return on investment through its comprehensive tool sets and low entry costs to get started. Based on the .NET platform, Speech Server applications easily integrate with various back ends using ADO.NET or Web Services.

Speech Server also includes powerful reporting and business analytics tools that can give clear insight into how our solutions are performing and where to drive process improvements. State of the art Microsoft speech technology enables us to create powerful business solutions that lead to cost savings and increased customer satisfaction and retention through better service to our customers and increased call center productivity. Innovative technology like Conversational Understanding increases speech application robustness by dealing with incorrect phrases, hesitation from callers, or even understanding when callers are correcting themselves.

3.2.10 SQL Server 2005 Reporting Services (SSRS)

Microsoft SQL Server 2005 Reporting Services is a server-based solution for building enterprise reports that draw content from a variety of relational and multidimensional data sources, publishing reports that can be viewed in various formats, and centrally managing security and subscriptions. The reports that we create can be viewed over a Web-based connection or as part of a Microsoft Windows application or SharePoint portal. Reporting Services includes graphical tools and wizards for creating and publishing reports and report models; report server management tools for administering Reporting Services; and application programming interfaces (APIs) for programming against and extending the Reporting Services object model.

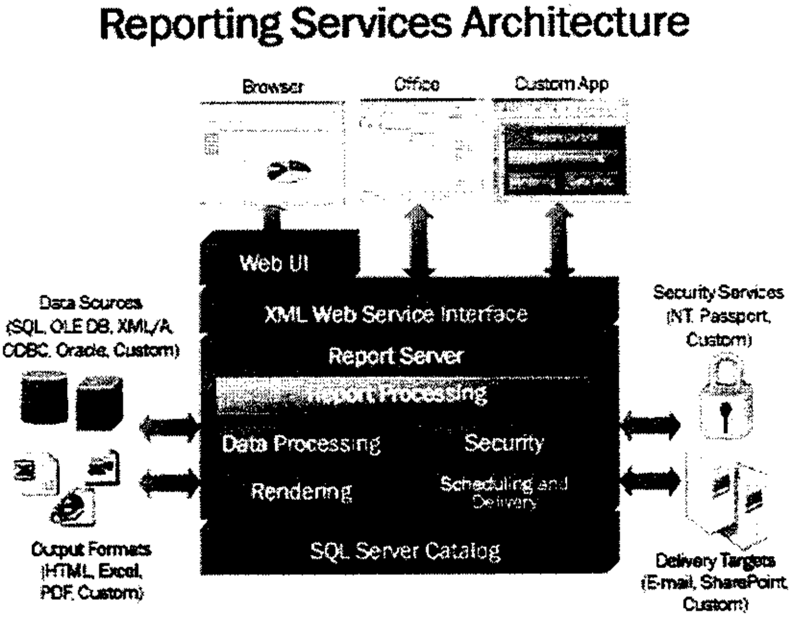

Reporting Services Architecture
Browser
Office
Custom App
Web UI
XML Web Service Interface
Report Server
Report Processing
Data Processing
Security
Rendering
Scheduling and Delivery
SQL Server Catalog
Data Sources (SQL, OLE DB, XML/A, ODBC, Oracle, Custom)
Output Formats (HTML, Excel, PDF, Custom)
Security Services (NT, Passport, Custom)
Delivery Targets (E-mail, SharePoint, Custom)

Fig. 3.2.9.1 SQL Server Reporting Services Architecture

SQL Server 2005 Reporting Services (SSRS) provides the developer several programming interfaces that can be used to extend the functionality of Reporting Services. These interfaces include Web services for managing and executing reports, a WMI provider interface for configuration management, a set of Microsoft .NET Framework managed APIs for creating Reporting Services extensions, and URL access for easy integration with Web portals and applications.

CHAPTER 4

SYSTEM DESIGN

4.1 UML DIAGRAMS

UML Diagrams allows a software engineer to express an analysis model using a modeling notation that is governed by a set of syntactic, semantic, and pragmatic rules. In UML, a system is represented using five different "views" that describe the system from distinctly different perspectives.

Each view is defined by a set of diagrams. The following views are present in UML:

- ✓ User model view - This view represents the system (product) from the user's (called actors in UML) perspective. The use-case is the modeling approach of choice for the user model view.
- ✓ Structural model view - Data and functionality are viewed from inside the system. That is, static structure (classes, objects, and relationships) is modeled.
- ✓ Behavioral model view - This part of the analysis model represents the dynamic or behavioral aspects of the system. It also depicts the interactions or collaborations between various structural elements described in the user model and structural model views.
- ✓ Implementation model view - The structural and behavioral aspects of the system are represented as they are to be built.
- ✓ Environment model view - The structural and behavioral aspects of the environment in which the system is to be implemented are represented.

In general, UML analysis modeling focuses on the user model and structural model views of the system.

4.1.1 Usecase Diagram

In software engineering, a use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Use case diagrams depict:

Use cases - A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors - An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

Associations - Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

System boundary boxes (optional) - We can draw a rectangle around the use cases, called the system boundary box, to indicate the scope of your system. Anything within the box represents functionality that is in scope and anything outside the box is not.

Packages (optional) - Packages are UML constructs that enable us to organize model elements (such as use cases) into groups. Packages are depicted as file folders and can be used on any of the UML diagrams, including both use case diagrams and class diagrams.

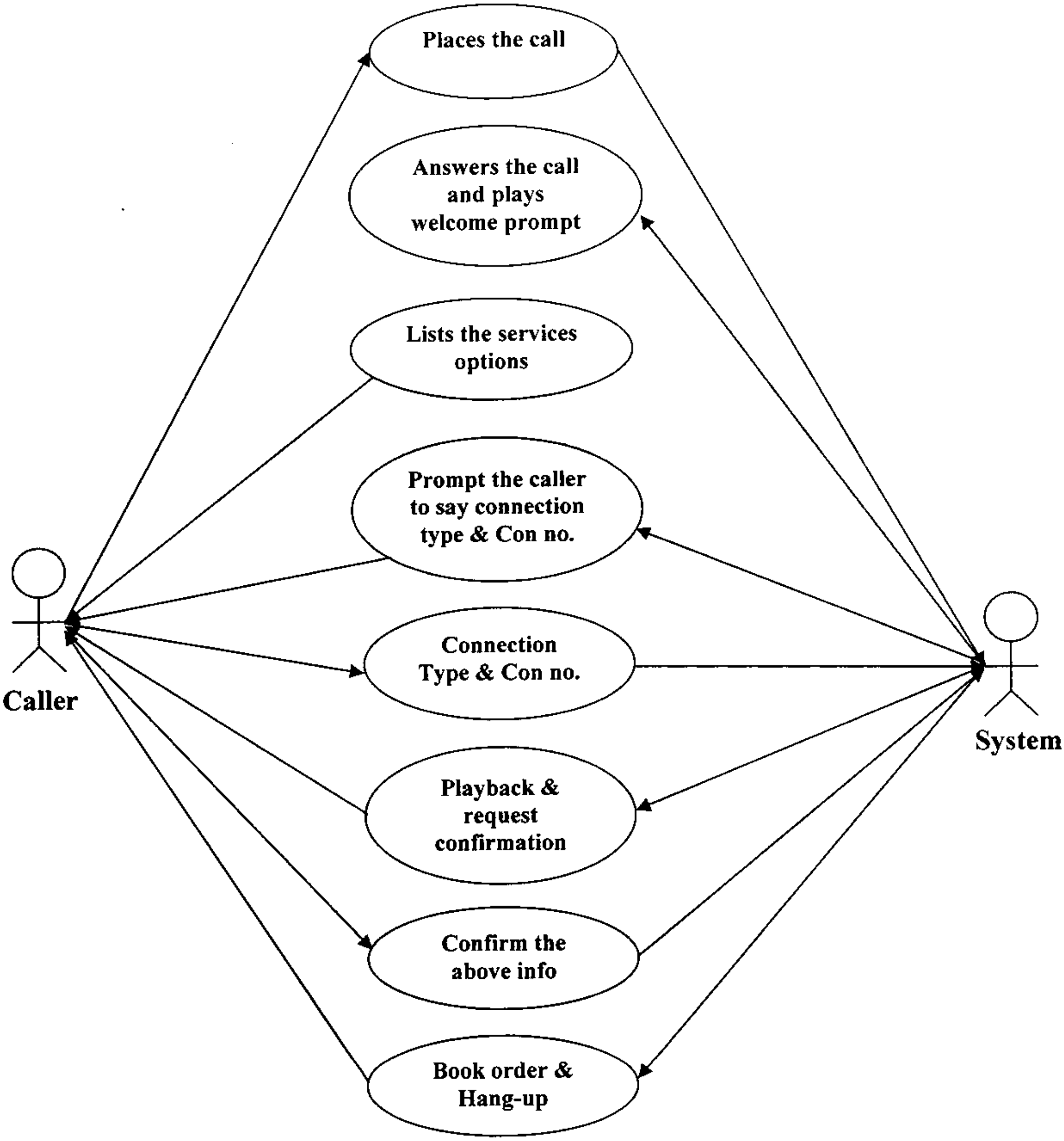

Places the call
Answers the call and plays welcome prompt
Lists the services options
Prompt the caller to say connection type & Con no.
Connection Type & Con no.
Playback & request confirmation
Confirm the above info
Book order & Hang-up
Caller
System

Fig. 4.1.1.1 Use-Case Diagram

4.1.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). With a dataflow diagram, users are able to visualize how the system will operate, what the system will accomplish and how the system will be implemented. On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. A data flow diagram illustrates the processes, data stores, and external entities in a business or other system and the connecting data flows.

Context Diagram

This is the highest level of dataflow diagram. It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization. This context-level DFD is then "exploded" to show more detail of the system being modeled. Here the primary source and destination entities are depicted with respect to the entire system.

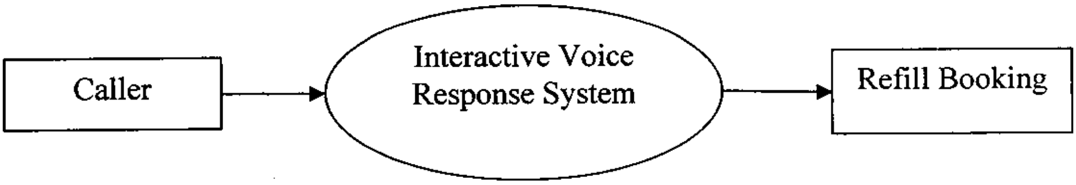


Fig. 4.1.2.1 Level 0 Context Level Diagram

Level 1 Data Flow Diagram

The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system. This is the level after the contextual diagram and it explodes the primary process to provide us with a view of all the sub processes in the system through which data flows.

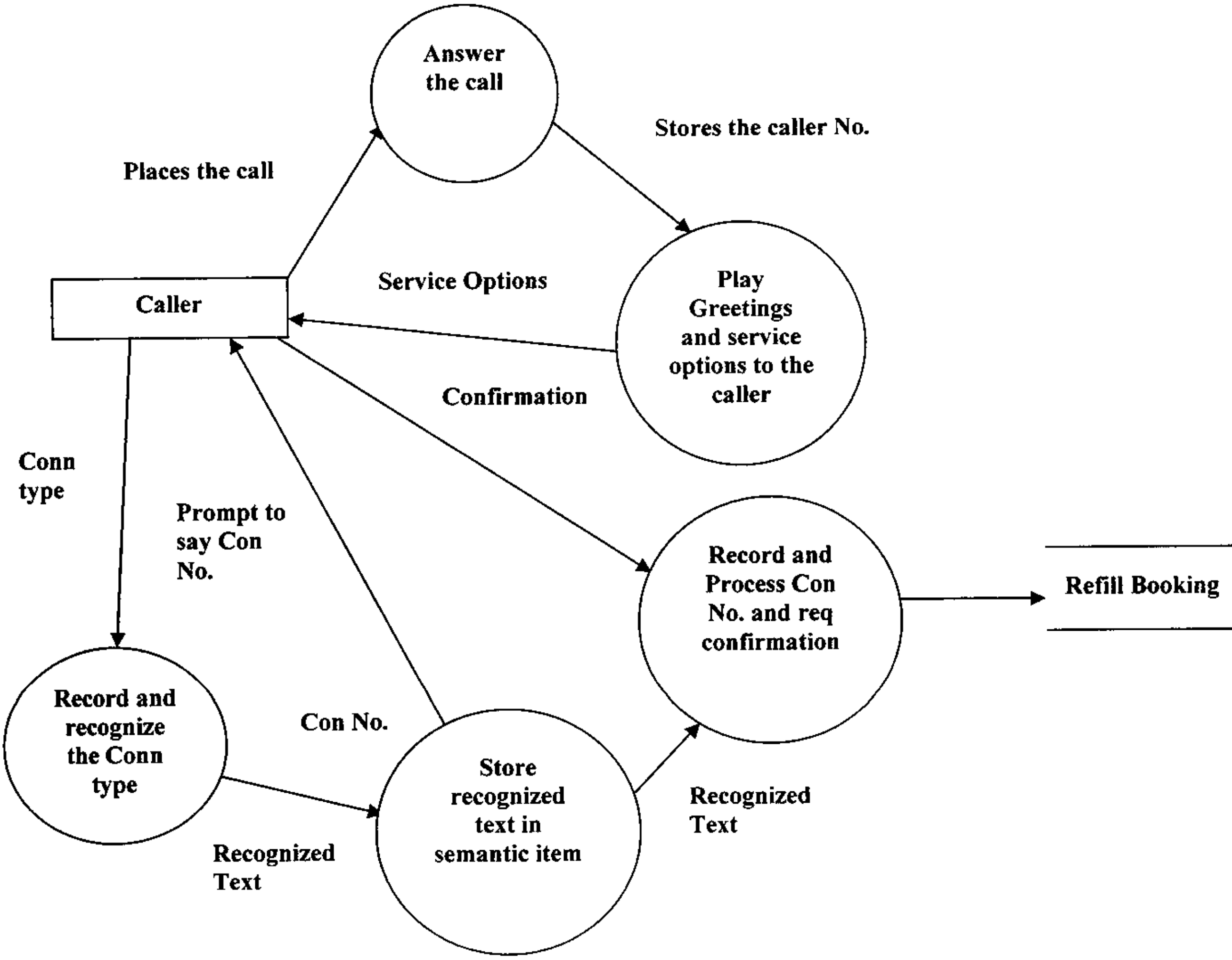

Answer the call
Stores the caller No.
Places the call
Caller
Service Options
Play Greetings and service options to the caller
Confirmation
Conn type
Prompt to say Con No.
Record and Process Con No. and req confirmation
Refill Booking
Record and recognize the Conn type
Con No.
Store recognized text in semantic item
Recognized Text
Recognized Text

Fig. 4.1.2.2 Level 1 Data Flow Diagram

4.2 ENTITY RELATIONSHIP DIAGRAM

In software engineering, an Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top down fashion. Diagrams created using this process are called entity-relationship diagrams, or ER diagrams or ERDs for short.

An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. It is an abstraction from the complexities of some domain. An entity may be a physical object such as a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order. Entities can be thought of as nouns. Every entity (unless it is a weak entity) must have a minimal set of uniquely identifying attributes, which is called the entity's primary key. Examples: a computer, an employee, a song, a mathematical theorem. Entities are represented as rectangles.

A relationship captures how two or more entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Relationships are represented as diamonds, connected by lines to each of the entities in the relationship.

Entities and relationships can both have attributes. Attributes are represented as ellipses connected to their owning entity sets by a line. Entity-relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets and relationship sets. Lines are drawn between entity sets and the relationship sets they are involved in. If all entities in an entity set must participate in the relationship set, a thick or double line is drawn. This is called a participation constraint. If each entity of the entity set can participate in at most one relationship in the relationship set, an arrow is drawn from the entity set to the relationship set. This is called a key constraint. To indicate that each entity in the entity set is involved in exactly one relationship, a thick arrow is drawn.

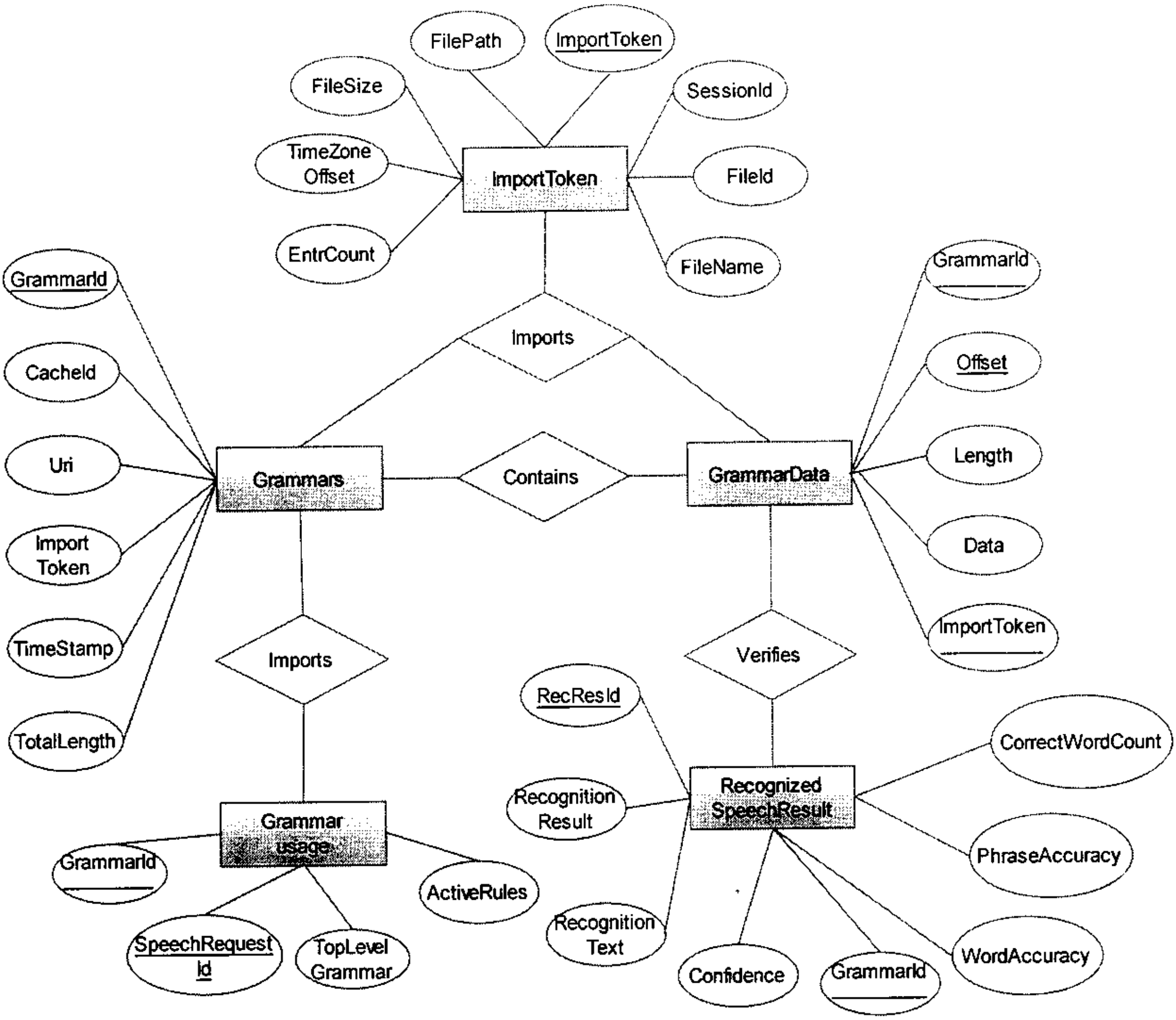


Fig. 4.2 Entity Relationship Diagram

4.3 SYSTEM FLOW DIAGRAM

A system flow diagram can consist of a subdivision to show sequential steps, with if-then-else conditions, repetition, and/or case conditions. Suitably annotated geometrical figures are used to represent operations, data, or equipment, and arrows are used to indicate the sequential flow from one to another.

In software and systems development, system flow diagrams can be used in system flow analysis, data flow analysis, algorithm analysis, and simulation. Control and data flow analysis are most applicable for real time and data driven systems. These flow analyses transform logic and data requirements text into graphic flows which are easier to analyze than the text.

A flow diagram can be developed for the process control system for each critical activity. System Flow Diagrams are diagrams used in systems design to represent all external entities that may interact with a system. System flow diagrams are related to Data Flow Diagram, and show the interactions between a system and other actors with which the system is designed to interface. System flow diagrams can be helpful in understanding the context which the system will be part of.

It is used early in a project to get agreement on the scope under investigation. System flow diagrams are typically included in a requirements document. These diagrams must be read by all project stakeholders and thus should be written in plain language, so the stakeholders can understand items within the document.

System Flow Diagrams are drawn using oval symbol, diamond box, and arrows to indicate the flow of process. The best System Flow Diagrams are used to display how system operates at a very high level or how systems operate and interact logically. The system flow diagram is a necessary tool in developing a baseline interaction between systems and actors; actors and system or systems and systems.

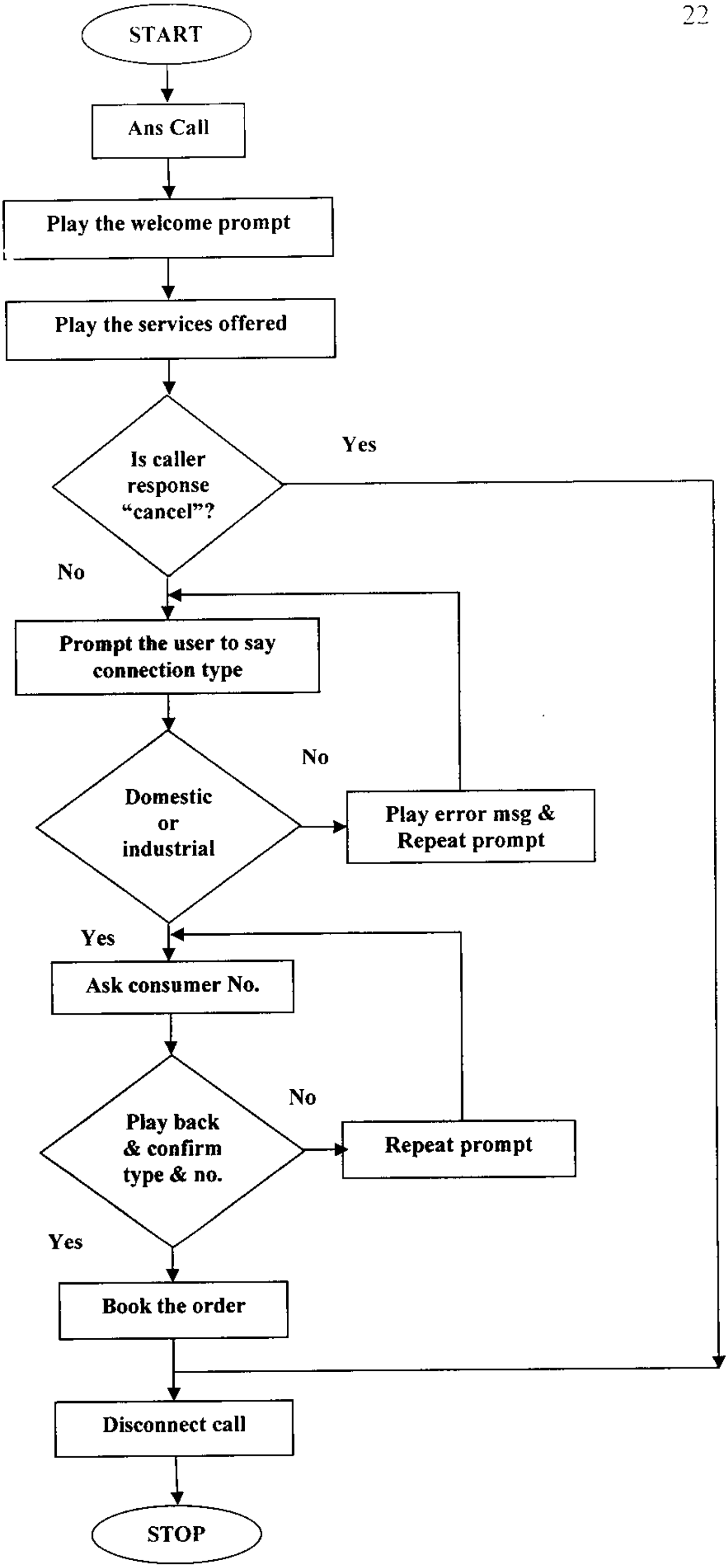

START
Ans Call
Play the welcome prompt
Play the services offered
Is caller response "cancel"?
Yes
No
Prompt the user to say connection type
Domestic or industrial
No
Play error msg & Repeat prompt
Yes
Ask consumer No.
Play back & confirm type & no.
No
Repeat prompt
Yes
Book the order
Disconnect call
STOP

Fig. 4.3 System Flow Diagram

CHAPTER 5

DATABASE DESIGN

5.1 CREATING A TUNING DATABASE

Microsoft SQL Server 2005 standard edition or Microsoft SQL Server 2005 Express Edition can be used to create the tuning database. The empty tuning database is created either by using server explorer or by using command prompt window. The schema is created only when we import speech application trace log files into the tuning database. SQL Server Express supports a single processor, and a maximum database size of 4 GB.

Database roles are used to regulate access to the tuning database. Roles can simplify security administration in databases with a large number of users or a complex security system. With database roles, the database administrator can improve the security of applications by granting the minimum required permission to the identity that connects to the tuning database. The following are guidelines to prepare the database for data import:

- ✓ Ensure that all database files are presized, including tempdb, to accommodate the amount of data in the import.
- ✓ For 100,000 calls, consider presizing the tuning database with 100 GB and set the transaction log size to 5 GB.
- ✓ Set auto grow to 100 MB.
- ✓ Transaction log files and database data files should reside on different physical spindles.

If the database is new and empty, all schema artifacts (for example, tables, views, procedures, and roles) are created and data is imported. If data has already been imported into the database and therefore the artifacts already exist, the new data is simply imported. Information contained in the tuning databases is intended for use in analyzing and improving the functionality of Speech Server applications.

5.1.1 Table's Description

Name	Description
Annotations	Contains fields for user data and calculated values that are related to an occurrence of a session, task, or turn.
ApplicationData	Contains the data logged by instances of the ApplicationDataEvent class.
Applications	Contains application type, name, and version information.
ApplicationTasks	Contains the names of tasks.
ApplicationTurns	Contains the names of turns.
CommandTypes	Contains the types of commands used in turns.
Cultures	Describes the language of a turn instance.
Databases	Stores the identity of the database holding the data. This table is used for Business Intelligence import scenarios.
Dates	A reference table holding dates.
Devices	Describes the calling device and the called device (phone numbers).
EngineRecoAudio	Contains the properties of the audio recorded by the recognition engine for an input turn.
EngineRecoAudioData	Contains the audio data recorded by the recognition engine.
EventTypes	Contains the names of different Speech Server event types as imported into the database.

GrammarData	Stores binary grammar streams associated with an audio record.
Grammars	Contains the properties of grammars logged for recognitions.
ImportFiles	Reference and fingerprint table for trace file data previously imported into the database.
ImportSessions	Tracks the history of each import session or batch. As multiple trace logs might be imported for any given session and trace logs might be reimported under certain scenarios, there is a many-to-many relationship between this table and the ImportFiles table.
ImportTokens	Intersection table for the many-to-many relationship between the ImportFiles and ImportSessions tables. It provides an audit mechanism for associating tuning data with the trace log from which it originated. The possible values for TimeZoneOffset are between 0 to 12 and 244 to 255. This is because this value is calculated from a signed BYTE. It stores time zone offsets in term of hour differences; that is +12 to -12.
Instances	Reference table for prompt instance requests.
Messages	Contains the data from Speech Server events or messages imported into the database.
NonEngineRecoAudio	Contains the properties of the audio recorded when Recognizer is not active. This data is used for session playback.
NonEngineRecoAudioData	Contains the audio data recorded when the recognition engine is not active. This data is used for session playback.

PromptAudio	Contains the properties of the output audio of a prompt. This data is used for session playback.
PromptAudioData	Contains the audio data recorded for a prompt. This data is used for session playback.
PromptData	Contains the plain text and markup of prompt content.
PromptInstanceRequests	Reference table for prompt instance requests.
Prompts	Describes the execution of a prompt request.
RecognitionData	Contains binary data associated with a speech recognition request.
RecognitionGroups	Holds properties of speech recognition cycles, both original and Re-recognition cycles.
RecognitionResults	Holds properties of speech recognition results.
RecognitionWords	Contains words and transcription comparison information for individual recognition results.
ResponseTypes	Contains the characterization of the user responses for a turn; typically Answered, Confirmed, Corrected, or Denied.
Runtime	Contains the type and version of the Speech Server runtime from which the data is generated.
SemanticAnswers	Details the state changes of a semantic item as occurring within a turn.
SemanticItems	Reference table for semantic items.
Servers	Reference table for servers running Speech Server.
SessionAggregate	An aggregation holding the latest session time stamp in the data.

SessionErrorCounts Contains counts of any errors for each session.

SessionErrors Contains the list of error types.

Sessions Contains the properties associated with a session, including called and calling parties, time stamps, ending status, and other information.

SpeechGrammarUsage Contains references to the grammars used for speech recognitions.

SpeechInstanceRequests Reference table for speech recognition instance requests.

SpeechRecognitions Contains the properties of individual speech recognitions, including configuration information and ending status.

SpeechTranscriptions Contains the transcription for a given turn, as entered in Analytics and Tuning Studio.

Subtasks Reference table for subtask information held by sessions and tasks.

Tasks Contains the properties of task instances, including name reference, ending status, duration, and turn count.

Times A reference table holding times of day.

Turns Contains the properties associated with turns, including name reference, ending status, and duration.

TurnTypes Contains the purposes of a turn, if known; typically Ask, Confirm, or Statement.

UserEvents Contains the data logged by instances of the **SessionLabelEvent** event and used for reporting with an application-specific custom filter.

UserEventTypes Reference table for **SessionLabelEvent** types.

5.1.2 Master Tables

5.1.2.1 ImportTokens Table

Column Name	Date Type	Constraint
ImportToken	Int	Primary key
SessionId	Int	Not Null
FileId	Int	Not Null
Filename	nvarchar(265)	Not Null
Filepath	nvarchar(265)	Not Null
Filesize	bigint	Not Null
Timezoneoffset	tinyint	Not Null
Entrycount	Int	Not Null

5.1.2.2 Grammars Table

Column Name	Date Type	Constraint
GrammarId	int	Primary key
CacheId	uniqueindentifier	Not Null
Uri	nvarchar(265)	Not Null
Totallength	int	Not Null
Timestamp	datetime	Not Null
ImportToken	Int	Foreign key

5.1.2.3 GrammarData Table

Column Name	Date Type	Constraint
Offset	Int	Primary key
GrammarId	Int	Foreign key
Length	Int	Not Null

Column Name	Date Type	Constraint
Data	Image	Not Null
ImportToken	Int	Foreign key

5.1.2.4 SpeechGrammarUsage Table

Column Name	Date Type	Constraint
SpeechRequestId	Bigint	Primary key
GrammarId	Int	Foreign key
TopLevelGrammar	Bit	Not Null
ActiveRules	nvarchar(265)	Not Null

5.1.2.5 RecognitionResults Table

Column Name	Date Type	Constraint
RecognitonResultId	Bigint	Primary key
RecognitionGroupId	Int	Not Null
RecognitionResult	Image	Allows Null
GrammarId	Int	Not Null
RecognitionText	nvarchar(265)	Allows Null
CorrectWordCount	Int	Allows Null
SubstituteWordCount	Int	Allows Null
InsertWordCount	Int	Allows Null
PhraseAccuracy	Float	Allows Null
Confidence	Float	Not Null
TranscriptionWordCount	Int	Allows Null

CHAPTER 6

SYSTEM DEVELOPMENT

System development is a series of operations performed to translate the system design produced during the design phase into code in user programming language. Modular approach is used for the development of the system. The system is a divided into several subsystems which make it easier to develop and perform tests on the whole system. The subsystems are also known as the modules and the process of dividing an entire system into subsystems/modules is known as Decomposition.

6.1 MODULE DESCRIPTION

The modules of this system are

- ✓ Speech Recognition Grammar Specification Module.
- ✓ Dialog Structure Framework Module.
- ✓ Prompt Generation Module.
- ✓ Response Gathering and Processing Module.
- ✓ Response Confirmation Module.
- ✓ Order Cancellation Module.
- ✓ Analysis and Tuning Module.

6.1.1 Speech Recognition Grammar Specification (SRGS) Module

For the speech recognition (SR) engine to return recognition from audio input, a grammar must be added to each Speech Control. In simple terms, a grammar is a representation of everything the user can be expected to say, with certain selections tagged with semantic properties and values. It is a structured list of rules that identify the words or phrases that can be used for speech input and subsequent automatic recognition. The grammar file is an XML file, and when saved using Speech Grammar Editor, has a .grxml file name extension. Grammar files include one or more rules.

In this application, the application welcomes the caller, and then prompts the caller for information in the following order:

- ✓ Type of the connection.
- ✓ Consumer number.

A grammar with rules for each of these cases is necessary to recognize the responses that a caller might make to those prompts. The rules that should be created are

- ✓ Type- This rule is to recognize phrases that are natural response to the question "What is your connection Type?"
- ✓ Consumer Number- This rule is to recognize phrases that are natural response to the question "What is your consumer number?"
- ✓ Confirm- This rule is to recognize phrases that are natural response to the question "is that correct?"
- ✓ Cancel- This rule is to recognize phrase "cancel"

6.1.2 Dialog Structure Framework Module

Effective dialog structure design is the key component for building successful interaction between the application and a user. The dialog structure for this application is created using series of speech QA controls. The call flow is determined and the controls are placed according to that flow. It manages the interaction between the system and the caller. The call flow of this system is as follows:

1. The system answers the consumer's call.

2. It plays a greeting and prompts the consumer for necessary information.

3. The system waits for consumer's response for a set period of time. The caller can respond by speaking.

4. The system play backs the information and prompts the consumer to confirm. It books the order on confirmation.

5. The consumer 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 "Cancel").

6.1.3 Prompt Generation Module

Prompts are audio files that speak to the user. It serves a number of functions such as for asking the user a question, for greeting the user, to provide information to the user or to direct the user to take specific action. Speech Prompt Editor is used to create, record, and edit prompts, which are then stored in prompt database as .wav files for use. It provides an interface for creating prompts. It is used to create transcriptions, recording audio data for each transcription, importing, exporting, and editing audio data, creating extractions, tuning word alignments. The prompt database contains all the audio and data that define the application's prompts. Some of the prompts created for this system are as follows:

- ✓ Please say your connection type.
- ✓ Please say your consumer number.
- ✓ Say Cancel at any time to cancel this order.

6.1.4 Response Gathering and Processing Module

The system needs to gather information from the caller's responses to the prompts that each control plays. It also needs to retain that information for use throughout the application. Each rule in the grammar specifies semantic information in Semantic Markup Language (SML) returned by the speech recognition (SR) engine after it analyzes the caller's responses.

The SemanticMap control is the repository for this information. It consists of a collection of SemanticItem controls, which contain information about semantic state and binding and postback properties. These SemanticItem controls contain the SML markup. A script tag element is added to each rule in the grammar. Appropriate semantic script is written to each rule to return the recognized text as a value in the SML output.

In this application, confirm rule semantic script returns "Yes" or "No" as a value, tagged as either Yes or No, depending on which phrase the SR engine recognizes in the SML output.

6.1.5 Response Confirmation Module

The system needs to confirm the responses that it has recognized to ensure that it has not recognized some phrase incorrectly and to give the caller a chance to correct the error if it exists. In this system, if the caller indicates that the information is not correct, the dialog flow returns to the first prompt for a response from the user, and repeats all the prompts again.

To play back the caller's responses as a prompt, it is necessary to create a prompt function. This prompt function extracts the text from the semantic items that were filled when the caller responded to previous prompts. It takes the type and consumer number text from siType and siConsumerNumberText and then concatenates them with some phrases in the prompt database to produce a prompt that is not in the prompt database. This requires a small prompt function to extract the contents of the semantic items from the SemanticMap control. The prompt function may contain the following code:

```
var sConfirmPromptStart = "You booked a ";
var sConfirmPromptMid = " cylinder, and your consumer number is ";
var sConfirmPromptEnd = " Is that correct?";
var sPrompt = "";
sPrompt = sConfirmPromptStart + sSize + " " + sConfirmPromptMid;
sPrompt = sPrompt + sConsumerNumberText + sConfirmPromptEnd;
return(sPrompt);
```

6.1.6 Order Cancellation Module

In this system, if the response to the question, "Is that correct?" is "Yes", the dialog flow simply continues and the application ends. However, if the response is "No," the system needs to take different action. In this case, the dialog flow begins again at the Type prompt and continues through the prompts until the Confirm prompt again. The onNotConfirmed function referred to in the OnClientChanged property of the siNo semantic item clears the values of the semantic items to which the controls refer. That starts the dialog flow again from the first in the speech index order with empty semantic items. To have the prompts play again if the caller does not confirm the information, the PlayOnce item in control properties must be cleared.

6.1.7 Analysis and Tuning Module

Analysis and tuning are a major part of the speech application development cycle. The most successful speech applications in deployment are those that have been tuned on the basis of data collected from real callers to the system. Using real data, speech developers can significantly improve the user experience and overall performance of their application.

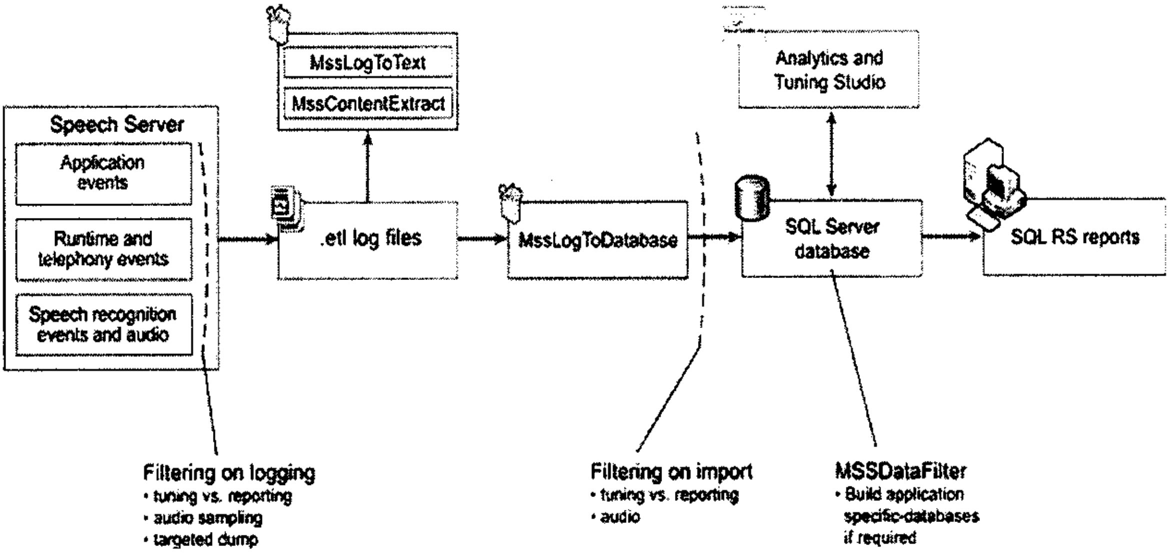


Fig. 6.1.7.1 Speech Application Data Flow

The system log files are imported into a tuning database and the reports are created using Analytics and Tuning Studio. Those reports are used to analyze the application tuning data. Analytics and Tuning Studio presents speech application call information by session, turn, and task. For each of these levels, Analytics and Tuning Studio offers statistical and detailed information views in reports, list views, and detail views.

The types of reports offered by analytics and tuning studio are as follows:

- ✓ Reco Accuracy Reports View - used to evaluate the results of Re-recognition cycles.
- ✓ Error Reports View - used to see general information about the frequency of errors with the application.
- ✓ Server Reports View - used to see general information about individual server usage for a Speech Server application.
- ✓ Session Reports View - to see general information about the frequency and characteristics of calls to a Speech Server voice response application.
- ✓ Task Reports View - used to see summary information about tasks in calls to a Speech Server voice response application.
- ✓ Turn Reports View - used to see summary information about turns in calls to a Microsoft Speech Server (MSS) application.

This system is analyzed and tuned based on session, task and turn reports. Sessions, tasks, and turns relate to each other. Typically, a session contains one or more tasks and a task contains one or more turns. Based

A session can generally be equated with a single call in the database. It shows statistical session data in graph and table format, including session volume, a breakdown of the session ending type, and session shape characteristics. A task is a module, where a goal is accomplished. In this application, event-logging schema defines tasks. It shows statistical task data in graph and table format, including task volume, completion status breakdown, and task shape characteristics. A turn is a unit of interaction between the system and a user. Typically, the interaction is a combination of prompt output and user input. It maps to a QuestionAnswerActivity, which generally implements a Statement. It shows statistical turn data in graph and table format, including turn volume and types, recognition and ending status, and other turn characteristics.

CHAPTER 7

SYSTEM TESTING

7.1 INTERACTIVE VOICE RESPONSE TESTING

IVR testing is a key element within a new IVRS development process. Testing involves not just the functioning of the IVR program but also how well the IVR program performs under stress and error conditions. Testing can occur during design, during development or after deployment. In the final stage of development, all of the system's components are working together for the first time and usability testing can focus on the interactions of all these components. Testing at this stage enables designers and engineers to fine-tune and optimize the system.

The first phase of testing enables designers to evaluate the overall success or failure of their design by using simulations of interactions with callers. Flaws are revealed before money and time are spent on implementation, allowing designers to collect information and evaluate user responses to the system design. In addition, testing can validate the design and track factors such as user satisfaction and successful automation rates in comparison to previously deployed (DTMF) systems.

7.1.1 Testing Objectives

Here are some questions that provide general information the designer want to receive from the design phase tests:

- ✓ Can the user complete a specific task?
- ✓ Are the prompts causing users to speak effective utterances in grammar?
- ✓ Is one design more efficient than another?
- ✓ Is the user easily able to learn the system? Evaluating this necessitates testing the same users over the same tasks multiple times.

- ✓ Is the user quickly able to share the mental model that the designers intended?
- ✓ Does the user like working with the system?

Some of these questions are highly subjective and can be difficult to answer objectively without tester bias.

7.1.2 IVR Test Plan Considerations

An IVR development project requires the thorough testing of the application prior to its installation in a live environment. The following are different test processes that need to be considered when constructing a test plan for an IVR installation:

- ✓ **Program Flow Test** - Are all of the IVR phone key paths fully functional? Is the user directed to the proper response when a speech response is made?
- ✓ **Voice Recordings Verification** - This step requires that every voice recording that is played to the user is fully functional and plays the complete message as designed.
- ✓ **Error Condition Tests** - If the user gives a wrong response, how does the system respond? Is the error message played to the user sufficiently clear and does the user know how to proceed?
- ✓ **Database Access Test** - Does the system retrieve the proper information over the network or from the internet? Are there failsafe procedures in place to ensure the proper information is accessible?
- ✓ **Disabled Communications Test** - If a communications breakdown occurs during any phase of the IVR process, Is the user given an appropriate warning and does the system recover properly from a transaction update?
- ✓ **Text To Speech Test** - Does the text to speech converter work properly with the supplied information retrieved? How does the converter respond under heavy usage?
- ✓ **Database Update Test** - During a transaction, is the information submitted by the user properly updated in the database? Is the data saved

and readily retrievable? Are there reasonable checks placed upon the captured data?

- ✓ **Voice Recording Test** - Does the system properly save information about the transaction that has been recorded?
- ✓ **System Load Test** - Does the system perform properly when subjected to a heavy load of calls? Is the system configured properly to handle expected peak loads? Are all of the IVR components (recordings, text to speech, and network access) capable of performing under extreme call volumes?
- ✓ **Intuitive Logic Test** - Finally, does the IVR perform in a manner that allows the intuitive user to understand the prompts and navigation? Is the system self-correcting and can the user start over or return to a prior response without terminating the transaction?

7.1.3 Testing Prompts

One of the debugging tasks to complete before deploying the application is to verify prompt coverage. Coverage is a measure of the proportion of prompts that an application can play. For example, consider this application; it produces the prompt "Confirm that you have ordered <some type> cylinder." The type in this sentence can be any type of cylinder. The application logic may call for the prompt "Confirm that you have ordered no cylinders." The word "no" might not have been recorded and marked as an extraction.

If any part of a prompt is not present in the database, the prompt engine plays the prompt using text-to-speech (TTS) rather than using the recorded voice from the database. To create more professional-sounding applications, ensure that there is 100 percent prompt coverage. Prompt Validation tool, included in Speech Prompt Editor, is used to validate prompt coverage.

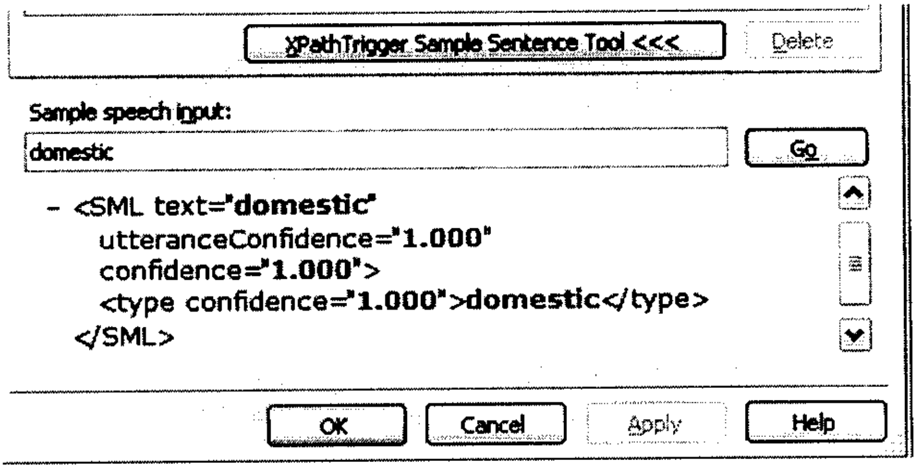
7.1.4 Testing Extractions

A second, related debugging task is to check the quality of prompts when various extractions are combined and played in sequence. This application might have the option of three types and a phrase cylinder, which can be combined in a variety of confirmation prompts. One such prompt might be "You ordered a commercial cylinder." How do the extractions "commercial" and "cylinder" sound when combined in this phrase? How do "domestic" and "cylinder" sound when played together, or "industrial" and "cylinder"? Prompt Validation tool is used to test extraction sound quality.

7.1.5 Testing user responses

The XPathTrigger Sample Sentence Tool is used to test sample phrases or key presses against a grammar associated with a control, and identify the path through the grammar that creates resulting Semantic Markup Language (SML).

Type the words or phrases In the Sample speech input text box to test against the associated speech grammar or grammars, and then click Go. If the grammars are able to recognize the words, phrases, or keypresses, the resulting SML appears in the window below the Sample speech input field. If the grammars are not able to recognize the words, phrases, or keypresses, a window containing the error message appears, "The sample text 'input' was not found in any of the speech grammars in the control.



Sample speech input: domestic

```
- <SML text="domestic"
    utteranceConfidence="1.000"
    confidence="1.000">
    <type confidence="1.000">domestic</type>
  </SML>
```

Fig. 7.1.5 XpathTrigger Sample Sentence Tool

CHAPTER 7

CONCLUSION

This project is an effective voice recognition solution that automates the refill booking process in a gas agency. It responds intelligently to individuals who are naturally speaking. For most service calls, the live operator (gas agency employee) never had to speak to a customer. This cut down calls to operators by 60%, allowing them to do other back office works without having to hire and train new employee. The impact of effective software attuned to the words of someone who is naturally speaking can be massive for any business.

Custom speech recognition software development offers a fantastic means of increasing productivity and efficiency with automated systems. Tailored IVR systems can help to reach business goals. The customization and integration of each voice recognition solution mean that the effectiveness is maximized for business, with a carefully crafted interface and optimal functioning based on company's particular needs. This system will cut costs, increase productivity, and heighten customer satisfaction.

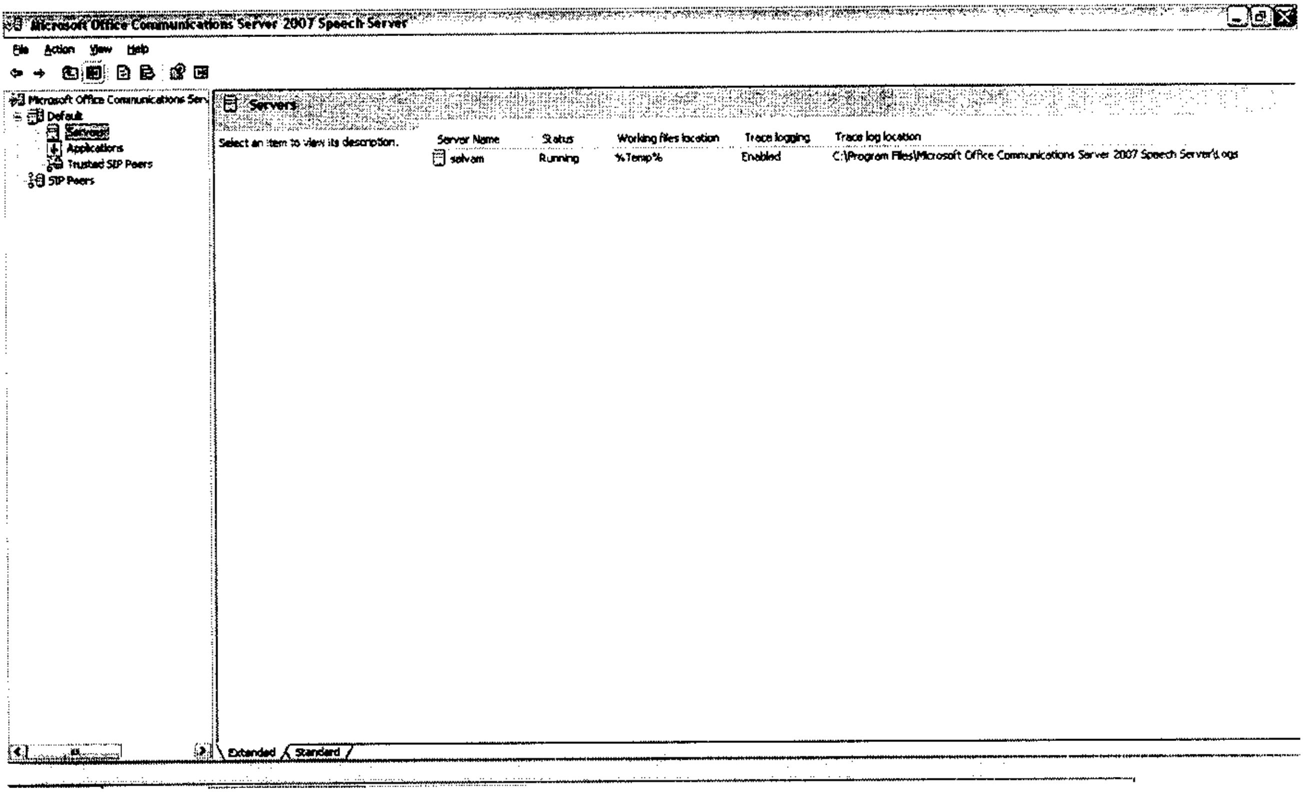
The system allows further enhancements to be done with ease. New modules can be added to the system with less effort. Future enhancements that are recommended to this system includes

- ✓ Support for outbound calls,
- ✓ Operator transfer,
- ✓ Instant and unlimited voice prompt recording,
- ✓ Call flow with unlimited level of navigation,
- ✓ Conversation recording…etc

8 APPENDICES

SCREEN SHOTS

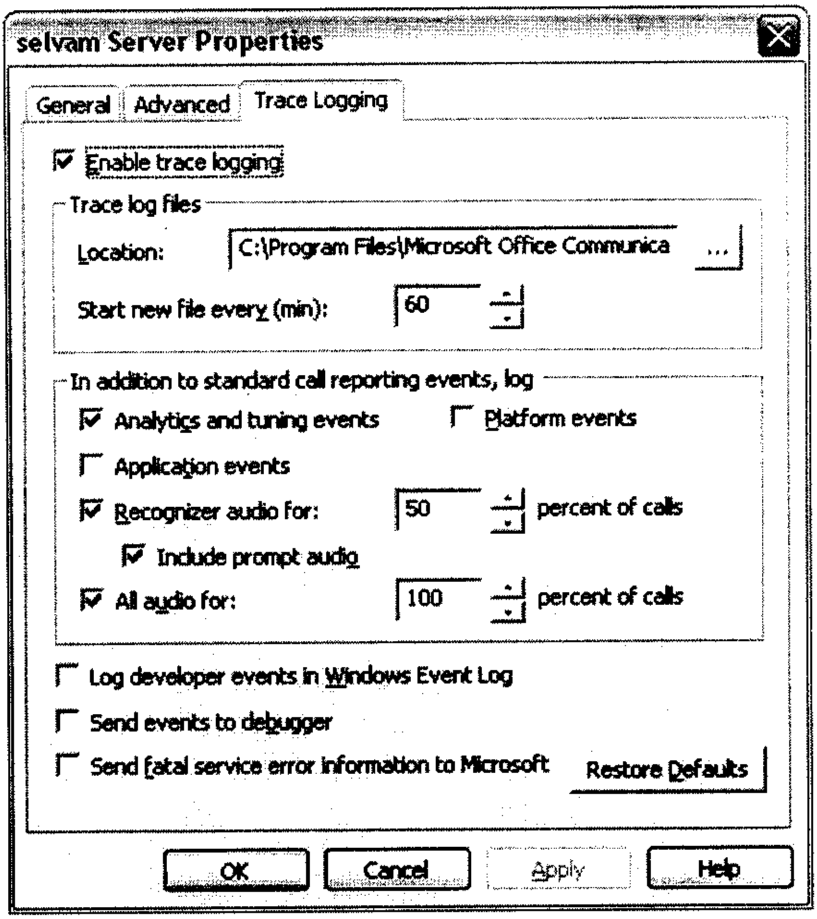
Microsoft Office Communication Server 2007 Speech Server


Microsoft Office Communications Server 2007 Speech Server
File
Action
View
Help
Microsoft Office Communications Serv
Default
Servers
Applications
Trusted SIP Peers
SIP Peers
Servers
Extended
Standard

Select an item to view its description.

Server Name	Status	Working files location	Trace logging	Trace log location
selvam	Running	%Temp%	Enabled	C:\Program Files\Microsoft Office Communications Server 2007 Speech Server\Logs

Microsoft Office Communication Server 2007 Speech Server Properties



☑ Enable trace logging

Trace log files

Location: C:\Program Files\Microsoft Office Communica ...

Start new file every (min): 60

In addition to standard call reporting events, log

☑ Analytics and tuning events □ Platform events

□ Application events

☑ Recognizer audio for: 50 percent of calls

☑ Include prompt audio

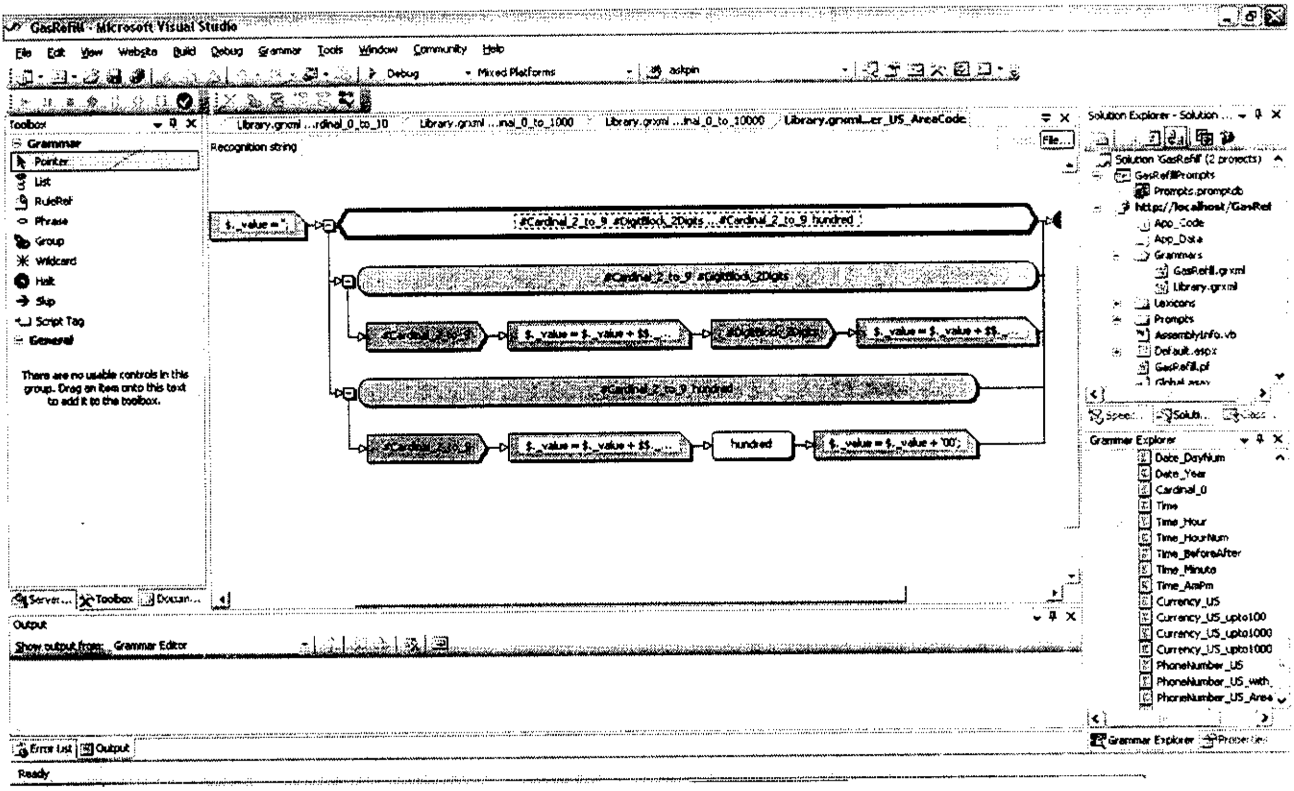
☑ All audio for: 100 percent of calls

□ Log developer events in Windows Event Log

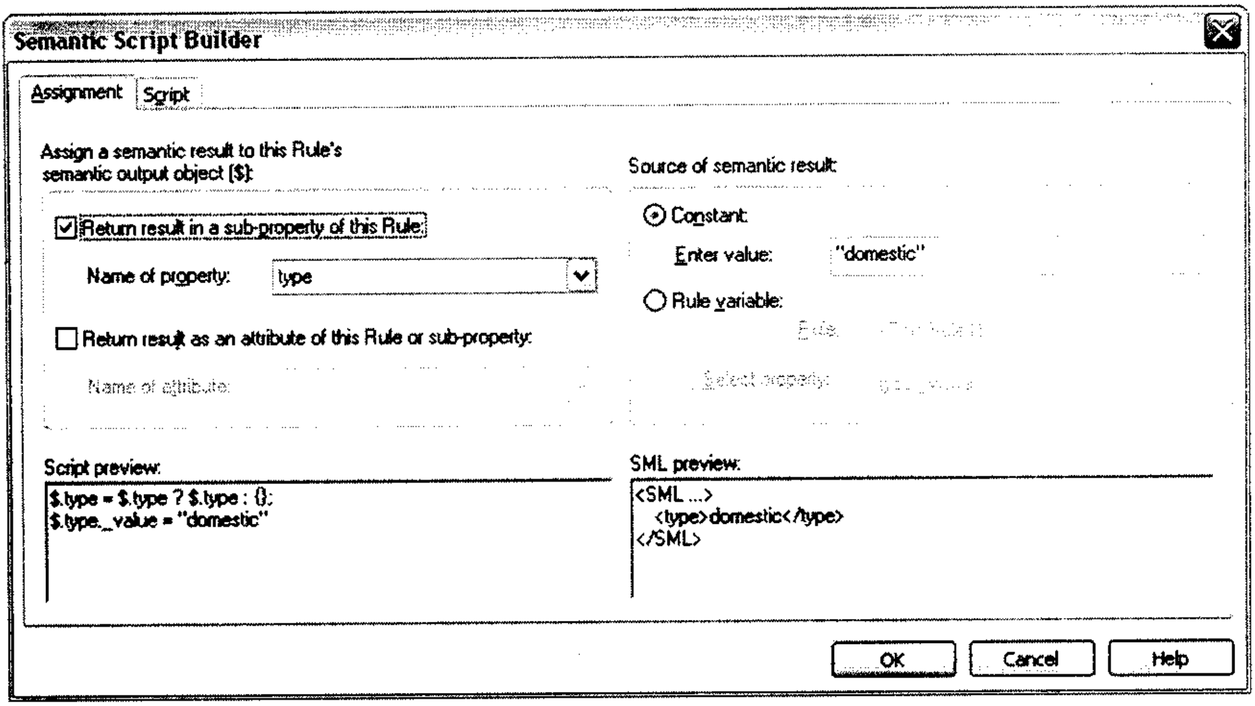
□ Send events to debugger

□ Send fatal service error information to Microsoft

Grammar: Rule Creation


GasRefill - Microsoft Visual Studio
File Edit View Website Build Debug Grammar Tools Window Community Help
Debug
Mixed Platforms
askpin
Toolbox
Grammar
Pointer
List
RuleRef
Phrase
Group
Wildcard
Halt
Skip
Script Tag
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Server... Toolbox Docum...
Library.grxml ...rdinal_0_to_10
Library.grxml ...inal_0_to_1000
Library.grxml ...inal_0_to_10000
Library.grxml...er_US_AreaCode
File...
Recognition string
$._value = '';
#Cardinal_2_to_9 #DigitBlock_2Digits ... #Cardinal_2_to_9 hundred
#Cardinal_2_to_9 #DigitBlock_2Digits
#Cardinal_2_to_9
$._value = $._value + $$._...
#DigitBlock_2Digits
$._value = $._value + $$._...
#Cardinal_2_to_9 hundred
#Cardinal_2_to_9
$._value = $._value + $$._...
hundred
$._value = $._value + '00';
Output
Show output from: Grammar Editor
Error List Output
Ready
Solution Explorer - Solution ...
Solution 'GasRefill' (2 projects)
GasRefillPrompts
Prompts.promptdb
http://localhost/GasRef
App_Code
App_Data
Grammars
GasRefill.grxml
Library.grxml
Lexicons
Prompts
AssemblyInfo.vb
Default.aspx
GasRefill.pf
Global.asax
Spee... Soluti... Class
Grammar Explorer
Date_DayNum
Date_Year
Cardinal_0
Time
Time_Hour
Time_HourNum
Time_BeforeAfter
Time_Minute
Time_AmPm
Currency_US
Currency_US_upto100
Currency_US_upto1000
Currency_US_upto1000
PhoneNumber_US
PhoneNumber_US_with_
PhoneNumber_US_Area
Grammar Explorer Properties

Adding Semantic script



Assign a semantic result to this Rule's semantic output object ($):

☑ Return result in a sub-property of this Rule:

Name of property: type

☐ Return result as an attribute of this Rule or sub-property:

Name of attribute:

Source of semantic result:

● Constant

Enter value: "domestic"

○ Rule variable:

Select property:

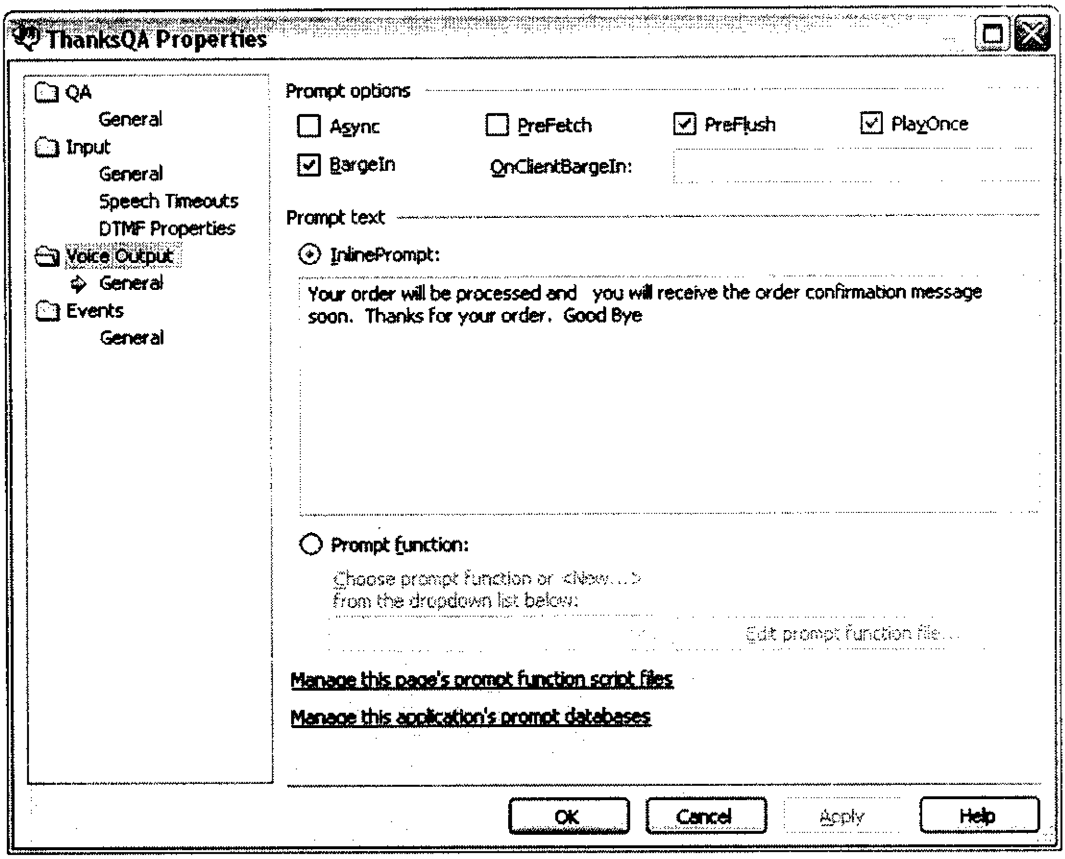
Script preview:

```
$.type = $.type ? $.type : {};
$.type._value = "domestic"
```

SML preview:

```
<SML ...>
  <type>domestic</type>
</SML>
```

Inline Prompt



ThanksQA Properties

QA
General
Input
General
Speech Timeouts
DTMF Properties
Voice Output
General
Events
General

Prompt options

□ Async □ PreFetch ☑ PreFlush ☑ PlayOnce

☑ BargeIn OnClientBargeIn:

Prompt text

● InlinePrompt:

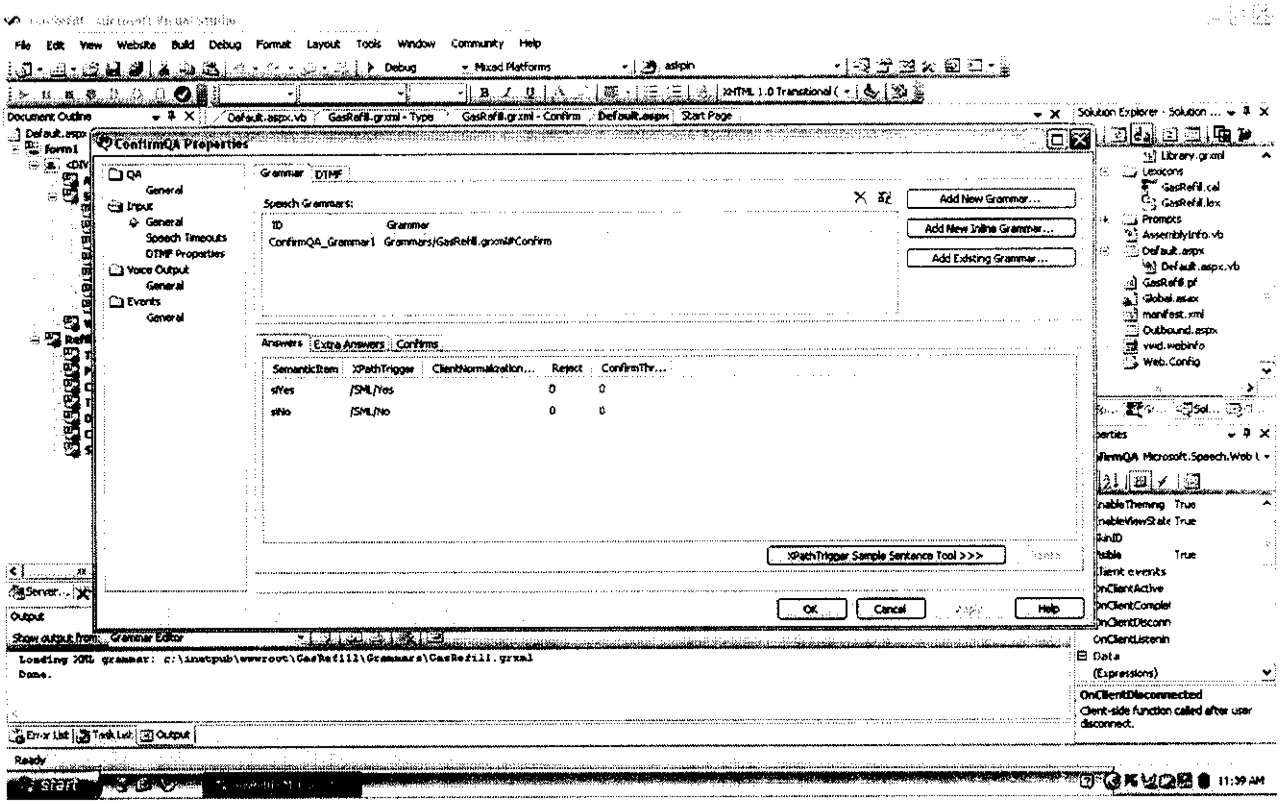
Your order will be processed and you will receive the order confirmation message soon. Thanks for your order. Good Bye

○ Prompt function:

Choose prompt function or <New...> from the dropdown list below:

Edit prompt function file...
Manage this page's prompt function script files
Manage this application's prompt databases
OK
Cancel
Apply
Help

Rule Binding



Speech Grammars:

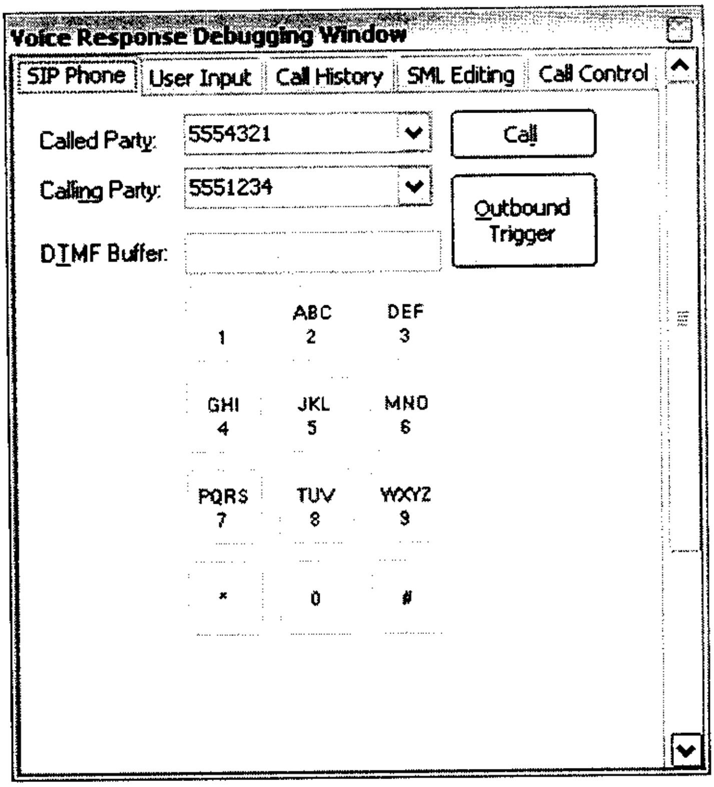
ID	Grammar
ConfirmQA_Grammar1	Grammars/GasRefill.grxml#Confirm

SemanticItem	XPathTrigger	ClientNormalization...	Reject	ConfirmThr...
siYes	/SML/Yes		0	0
siNo	/SML/No		0	0

Show output from: Grammar Editor

```
Loading XML grammar: c:\inetpub\wwwroot\GasRefill\Grammars\GasRefill.grxml
Done.
```

Voice Response Debugging Window

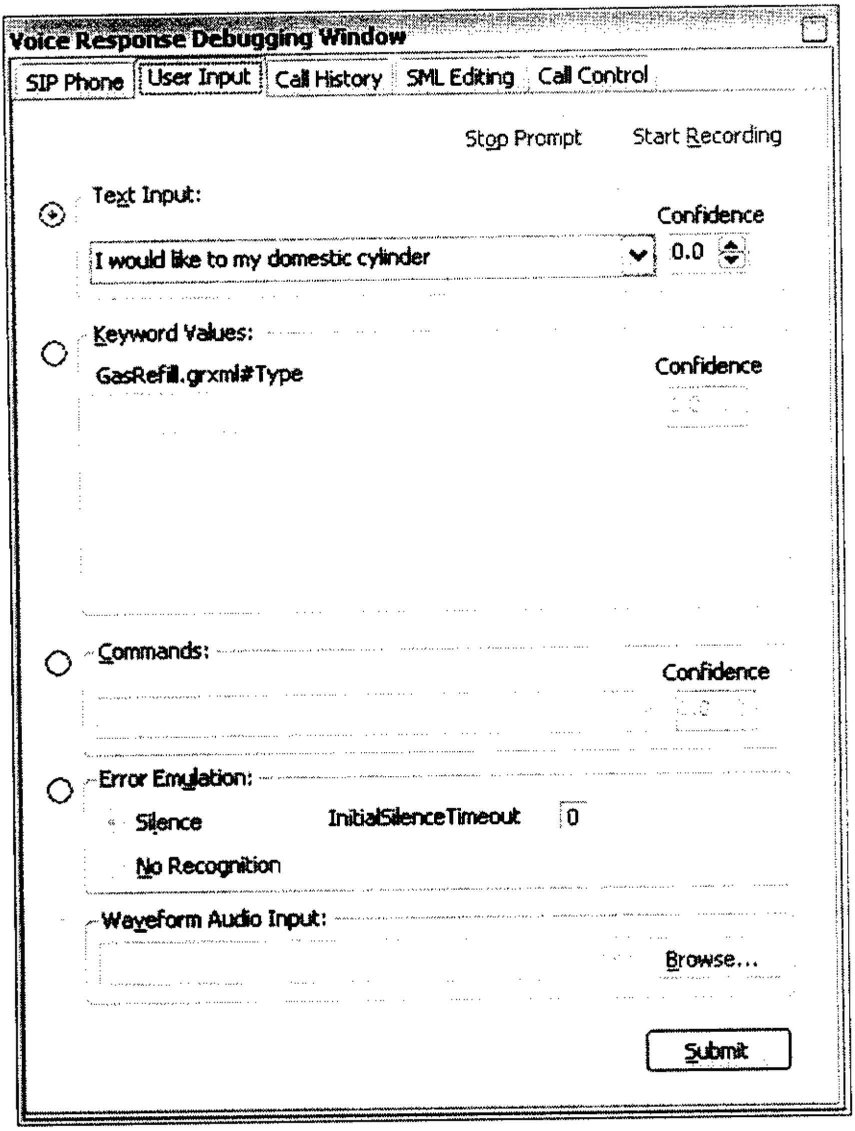


Called Party: 5554321

Calling Party: 5551234

DTMF Buffer:

Getting Connection Type from the Caller



Voice Response Debugging Window

● Text Input: I would like to my domestic cylinder — Confidence 0.0

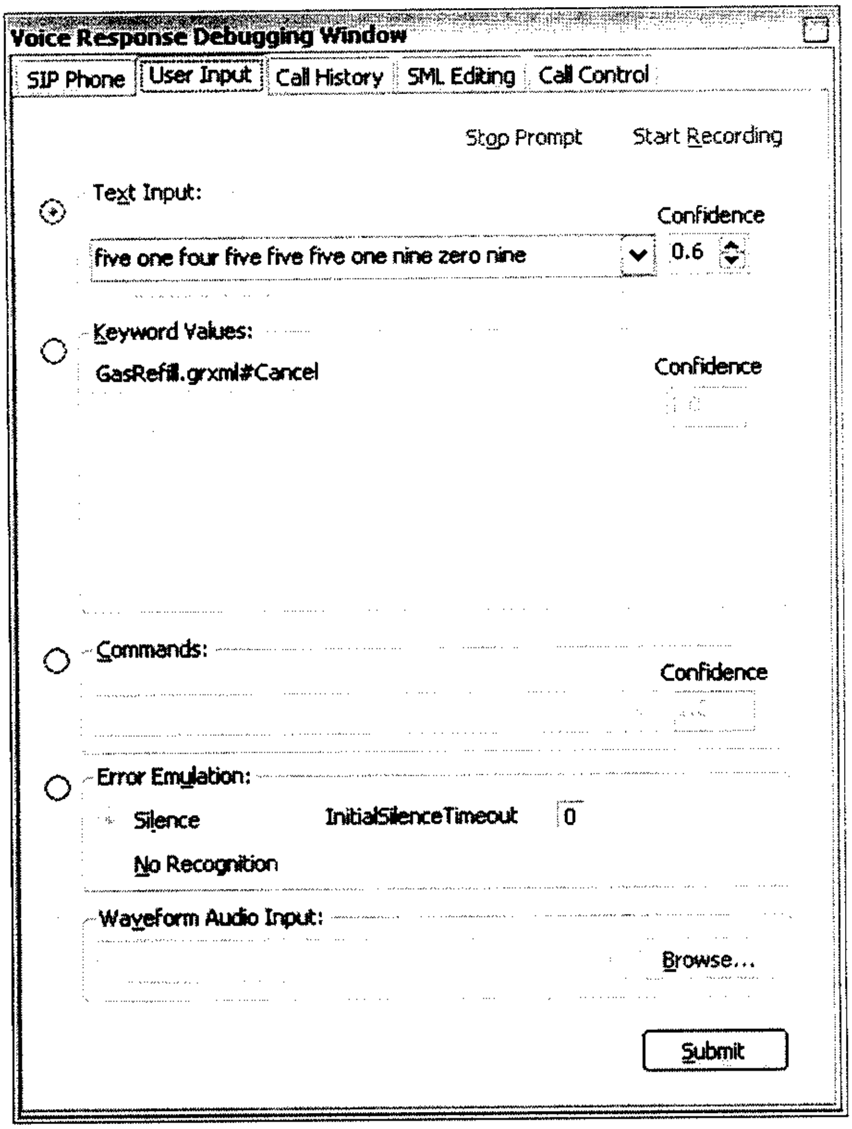
○ Keyword Values: GasRefill.grxml#Type — Confidence

○ Commands: — Confidence

○ Error Emulation: Silence — InitialSilenceTimeout 0 — No Recognition

Waveform Audio Input:

Getting Consumer Number from the Caller



Voice Response Debugging Window

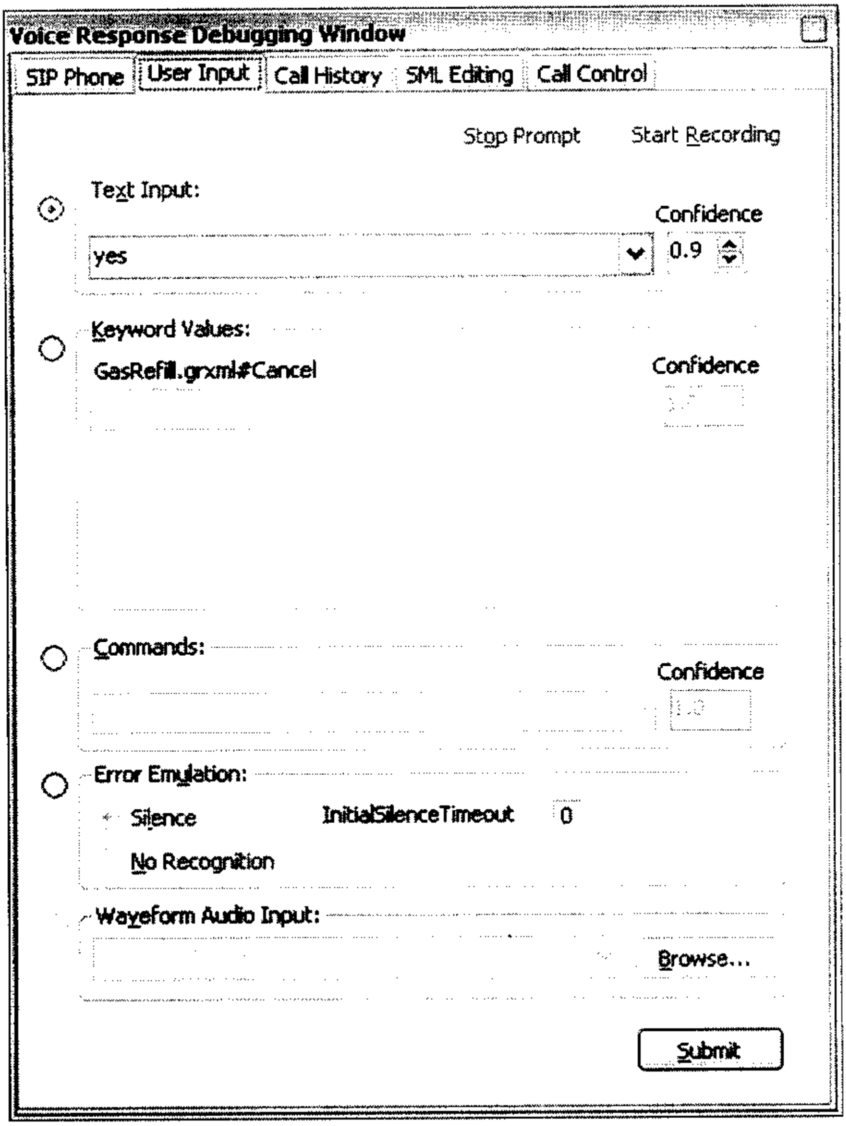
Text Input: five one four five five five one nine zero nine

Confidence 0.6

Keyword Values: GasRefill.grxml#Cancel

InitialSilenceTimeout 0

Getting Confirmation from the caller



Voice Response Debugging Window

● Text Input: yes — Confidence 0.9

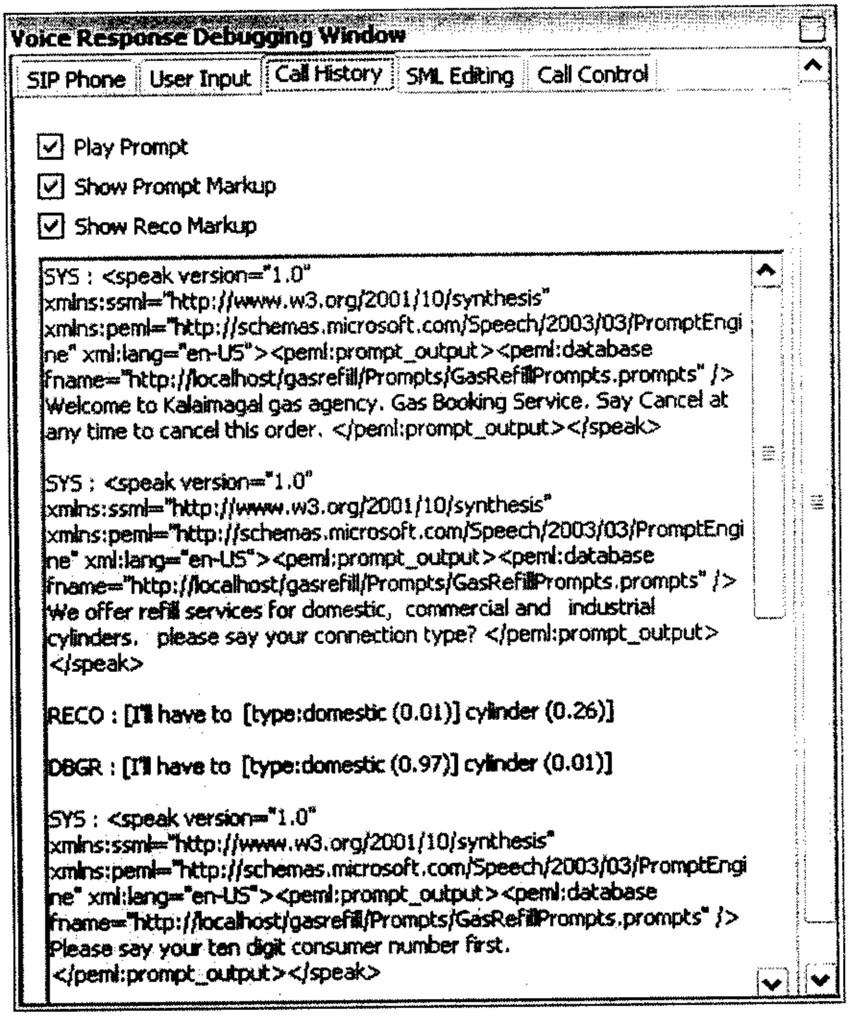
○ Keyword Values: GasRefill.grxml#Cancel — Confidence

○ Commands: — Confidence

○ Error Emulation: Silence — InitialSilenceTimeout 0; No Recognition

Wayeform Audio Input:

Call History



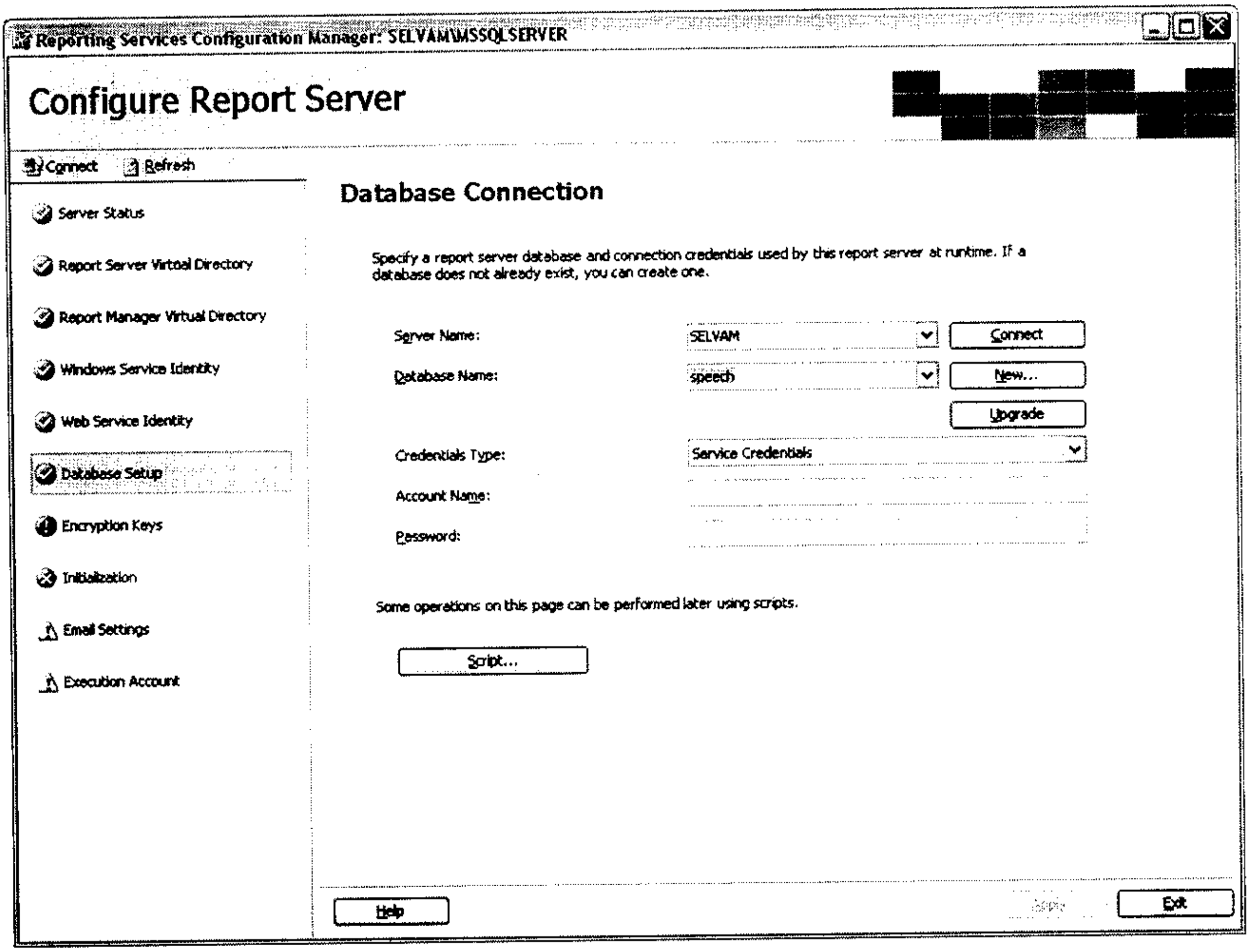
☑ Play Prompt

☑ Show Prompt Markup

☑ Show Reco Markup

```
SYS : <speak version="1.0"
xmlns:ssml="http://www.w3.org/2001/10/synthesis"
xmlns:peml="http://schemas.microsoft.com/Speech/2003/03/PromptEngi
ne" xml:lang="en-US"><peml:prompt_output><peml:database
fname="http://localhost/gasrefill/Prompts/GasRefillPrompts.prompts" />
Welcome to Kalaimagal gas agency. Gas Booking Service. Say Cancel at
any time to cancel this order. </peml:prompt_output></speak>

SYS : <speak version="1.0"
xmlns:ssml="http://www.w3.org/2001/10/synthesis"
xmlns:peml="http://schemas.microsoft.com/Speech/2003/03/PromptEngi
ne" xml:lang="en-US"><peml:prompt_output><peml:database
fname="http://localhost/gasrefill/Prompts/GasRefillPrompts.prompts" />
We offer refill services for domestic,  commercial and  industrial
cylinders.  please say your connection type? </peml:prompt_output>
</speak>

RECO : [I'll have to  [type:domestic (0.01)] cylinder (0.26)]

DBGR : [I'll have to  [type:domestic (0.97)] cylinder (0.01)]

SYS : <speak version="1.0"
xmlns:ssml="http://www.w3.org/2001/10/synthesis"
xmlns:peml="http://schemas.microsoft.com/Speech/2003/03/PromptEngi
ne" xml:lang="en-US"><peml:prompt_output><peml:database
fname="http://localhost/gasrefill/Prompts/GasRefillPrompts.prompts" />
Please say your ten digit consumer number first.
</peml:prompt_output></speak>
```

Report Server Configuration



Database Connection

Specify a report server database and connection credentials used by this report server at runtime. If a database does not already exist, you can create one.

Server Name: SELVAM

Database Name: speech

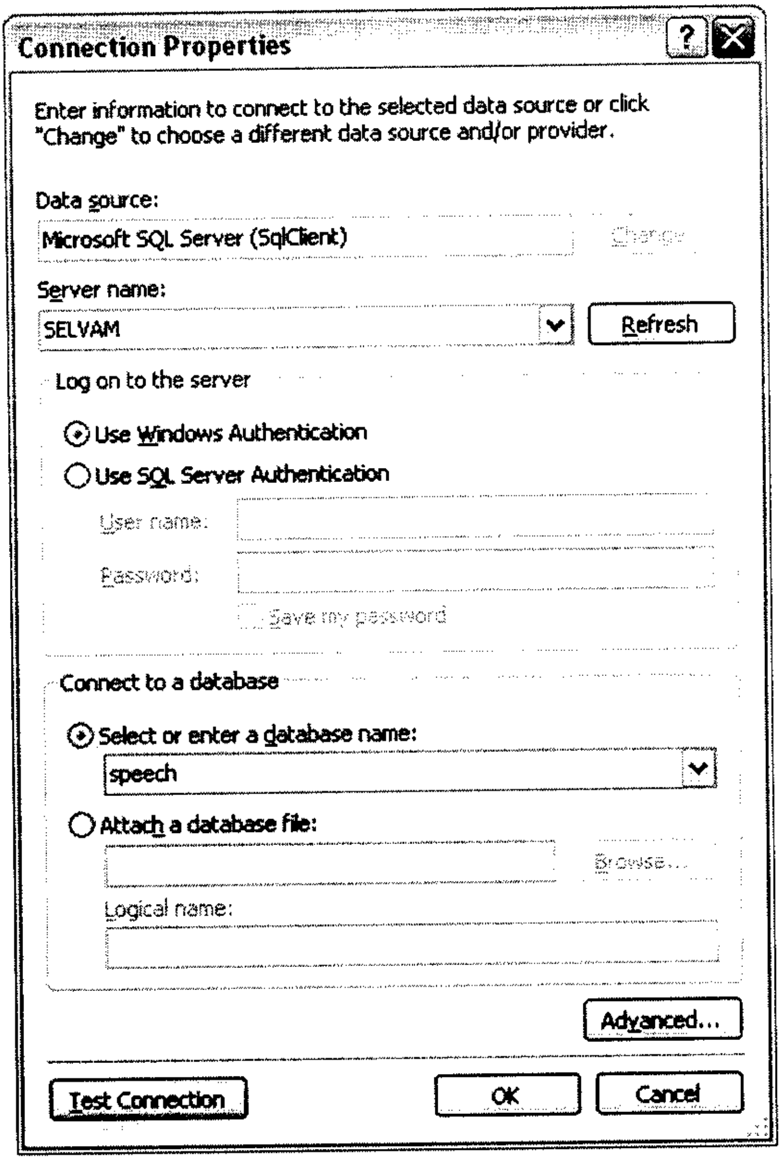
Credentials Type: Service Credentials

Account Name:

Password:

Some operations on this page can be performed later using scripts.

Analytics and Tuning Studio Connection Properties



Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: SELVAM

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

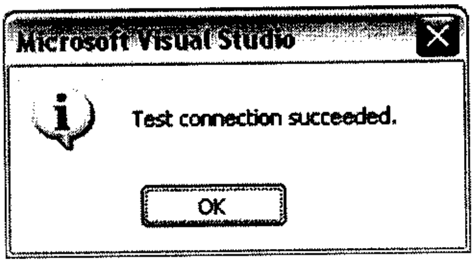
□ Save my password

Connect to a database

● Select or enter a database name: speech

○ Attach a database file:

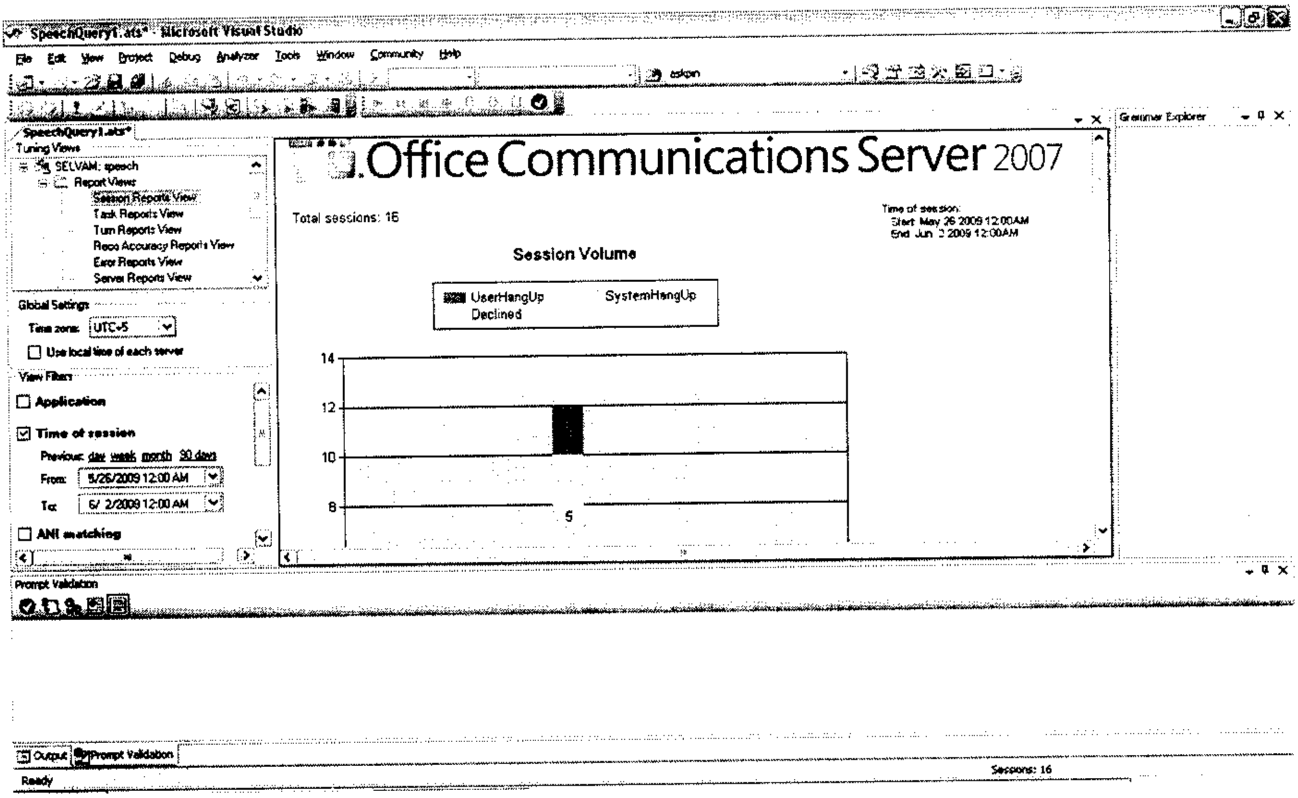
Logical name:



Microsoft Visual Studio

Test connection succeeded.

Session Report View



Total sessions: 16

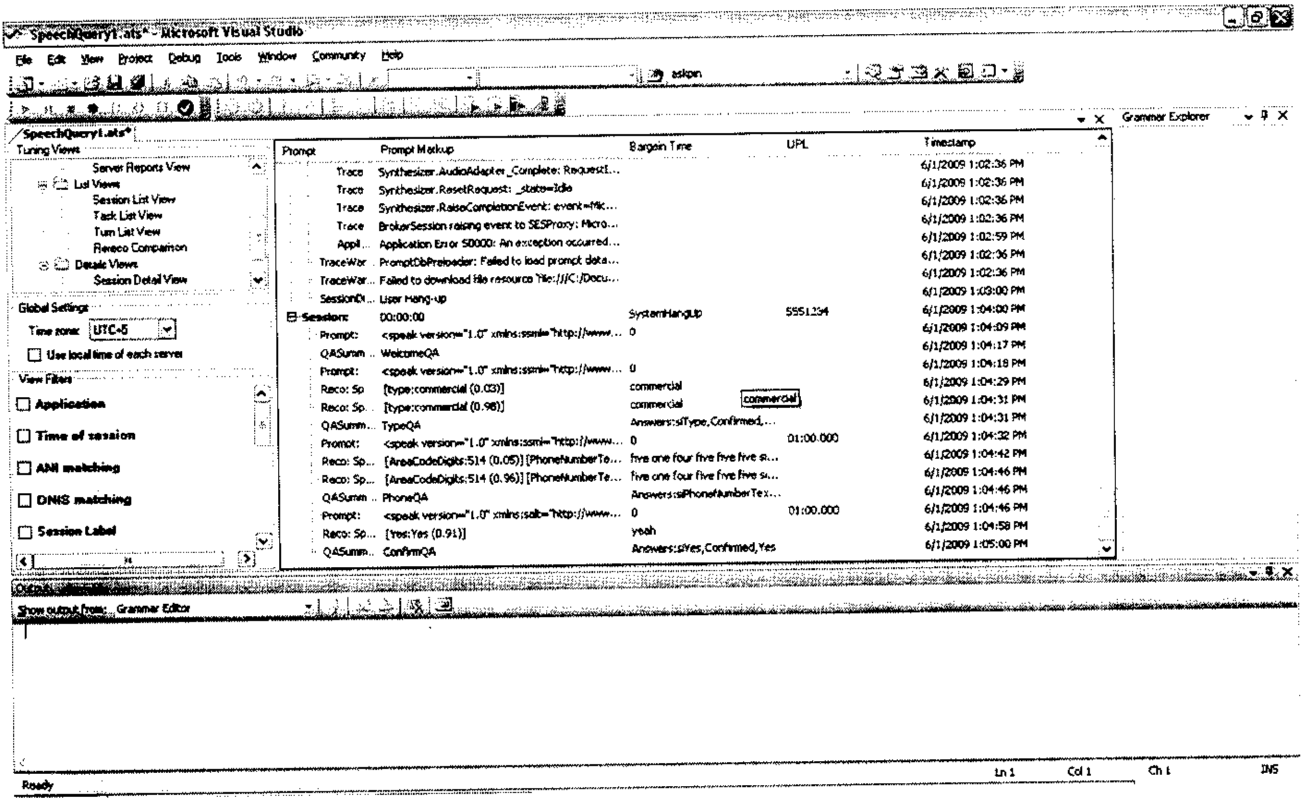
Time of session:
Start May 26 2009 12:00AM
End Jun 2 2009 12:00AM

Global Settings
Time zone: UTC+5
☐ Use local time of each server

View Filters
☐ Application
☑ Time of session
Previous: day week month 90 days
From: 5/26/2009 12:00 AM
To: 6/ 2/2009 12:00 AM
☐ ANI matching

Ready
Sessions: 16

Session Detail View



Prompt	Prompt Markup	Bargein Time	UPL	Timestamp
Trace	Synthesizer.AudioAdapter_Complete: RequestI...			6/1/2009 1:02:36 PM
Trace	Synthesizer.ResetRequest: _state=Idle			6/1/2009 1:02:36 PM
Trace	Synthesizer.RaiseCompletionEvent: event=Mic...			6/1/2009 1:02:36 PM
Trace	BrokerSession raising event to SESProxy: Micro...			6/1/2009 1:02:36 PM
Appl...	Application Error 50000: An exception occurred...			6/1/2009 1:02:59 PM
TraceWar	PromptDbPreloader: Failed to load prompt data...			6/1/2009 1:02:36 PM
TraceWar...	Failed to download file resource "file:///C:/Docu...			6/1/2009 1:02:36 PM
SessionD...	User Hang-up			6/1/2009 1:03:00 PM
⊟ Session:	00:00:00	SystemHangUp	5551234	6/1/2009 1:04:00 PM
Prompt:	<speak version="1.0" xmlns:ssml="http://www...	0		6/1/2009 1:04:09 PM
QASumm...	WelcomeQA			6/1/2009 1:04:17 PM
Prompt:	<speak version="1.0" xmlns:ssml="http://www...	0		6/1/2009 1:04:18 PM
Reco: Sp	[type:commercial (0.03)]	commercial		6/1/2009 1:04:29 PM
Reco: Sp...	[type:commercial (0.98)]	commercial		6/1/2009 1:04:31 PM
QASumm...	TypeQA	Answers:siType,Confirmed,...		6/1/2009 1:04:31 PM
Prompt:	<speak version="1.0" xmlns:ssml="http://www...	0	01:00.000	6/1/2009 1:04:32 PM
Reco: Sp...	[AreaCodeDigits:514 (0.05)] [PhoneNumberTe...	five one four five five five s...		6/1/2009 1:04:42 PM
Reco: Sp...	[AreaCodeDigits:514 (0.96)] [PhoneNumberTe...	five one four five five five s...		6/1/2009 1:04:46 PM
QASumm...	PhoneQA	Answers:siPhoneNumberTex...		6/1/2009 1:04:46 PM
Prompt:	<speak version="1.0" xmlns:salb="http://www...	0	01:00.000	6/1/2009 1:04:46 PM
Reco: Sp...	[Yes:Yes (0.91)]	yeah		6/1/2009 1:04:58 PM
QASumm...	ConfirmQA	Answers:siYes,Confirmed,Yes		6/1/2009 1:05:00 PM

9 REFERENCES

Books

- G. Andrew Duthie "Microsoft ASP.NET Programming with Microsoft Visual C# .NET Version 2003 Step By Step", Microsoft Press.
- Reed Jacobson, Stacia Misner, "Microsoft SQL Server 2005 Reporting Services step by step", 2005 Edition – Microsoft Corporation.
- Michael Dunn, "Pro Microsoft Speech Server 2007: Developing Speech Enabled Applications with .NET"

Websites

- http://msdn.microsoft.com
- http://beyondivr.com
- http://www.simple-talk.com
- http://www.microsoft.com/speech
- http://en.wikipedia.org