Fifth Edition (July 1996)

This edition applies to Version 2.1.0 of the IBM CallPath DirectTalk/2 Voice Processing System and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page titled “Sending Your Comments to IBM.” If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories
Information Development
Mail Point 095
Hursley Park
Winchester
Hampshire
England SO21 2JN

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1992, 1996. All rights reserved.
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.
### Contents

**Notices** ................................................................. xiii
Trademarks and Service Marks .......................................... xiii

**About This Book** ....................................................... xv
Who Should Read This Book .............................................. xv
How to Use This Book ..................................................... xv
Where to Find More Information ......................................... xvi
Other DirectTalk/2 Books and Information ........................... xvi
Non-DirectTalk/2 IBM Documentation Referenced in This Book .... xvii
Non-IBM Hardware and Software Related Information ............. xvii

### Part 1. Getting Started ................................................ 1

**Chapter 1. What is a Voice Application?** .......................... 3
Voice Programs ..................................................................... 3
Voice Logic Modules ........................................................... 4
Voice Segments .................................................................... 5
Text Segments ...................................................................... 5
Voice Application Variables ................................................. 5

**Chapter 2. Understanding the Voice Application Developer** ........ 7

**Chapter 3. Using the Voice Application Developer** ............... 9
Starting the Voice Application Developer ............................... 9
Using Voice Application Developer Windows ......................... 9
Exiting the Voice Application Developer ............................... 12

**Chapter 4. Creating a simple voice application** .................. 13
Designing the Voice Application ............................................ 13
Naming the Voice Application ............................................. 17
Creating the Voice Program ................................................ 18
Creating the Voice Logic Modules ........................................ 26
Creating and Recording Voice Segments .............................. 33
Checking and Testing the Voice Application ........................... 36
Preparing to Run the Application ......................................... 38

### Part 2. More About Voice Application Development .................. 41

**Chapter 5. Variables for Voice Applications** .................... 43
Creating and Changing Local Variables ............................... 44
Creating and Changing Global Variables ............................. 46
System Variables ................................................................ 47
User Variables .................................................................... 52

**Chapter 6. More About Voice Applications** ....................... 55
Opening Another Voice Application ..................................... 55
Printing the Voice Application ........................................... 56
Using Different Languages .................................................. 57
Chapter 7. More About Voice Programs ........................................ 59
  Editing Voice Programs .................................................. 59
  Setting Break Points .................................................... 61
  Handling System Error Return Codes .................................. 62

Chapter 8. More About Voice Logic Modules ............................ 63
  Voice Logic Module Functions .......................................... 63
  Editing Voice Logic Modules ........................................... 64
  Adding Voice Logic Modules ........................................... 66
  Deleting Voice Logic Modules ......................................... 66
  Copying Voice Logic Modules .......................................... 66
  Testing Voice Logic Modules ........................................... 67
  Printing Voice Logic Modules ......................................... 67
  Using Common Voice Logic Modules .................................. 68
  Voice Logic Module Currency Variable .............................. 69

Chapter 9. More About Voice Segments ................................ 71
  Adding and Changing Voice Segments ................................ 71
  Deleting Voice Segments ................................................ 71
  Playing Voice Segments .................................................. 71
  Copying Voice Segments .................................................. 71
  Printing Voice Segments .................................................. 72
  Using System-Supplied Voice Segments ................................ 72
  Using Common Voice Segments ......................................... 72
  Editing Voice Segments .................................................. 73
  Recording or Editing Multiple Voice Segments ....................... 75
  Recording Voice Segments Using a Tape Player or Microphone .................. 76

Chapter 10. Call Transfer and Referral .................................. 79
  Call Extension on Analog and T1 Lines ................................. 79
  Call Extension on E1 Lines .............................................. 79

Chapter 11. Password Security ............................................. 85

Chapter 12. DirectTalk/2 REXX Environment ............................ 89
  REXX Sample Voice Applications ...................................... 89
  Command Line Processing ................................................ 90
  Variable Processing ....................................................... 92
  REXX Subroutines and Functions ...................................... 95
  Protecting REXX Voice Programs ...................................... 96
  Handling System Error Return Codes ................................ 97

Part 3. Optional Features ................................................... 99

Chapter 13. Voice Messaging .................................................. 101

Chapter 14. Voice Recognition ............................................. 103
  Voice Recognition Configuration Parameters ......................... 104
  Vocabulary Files .......................................................... 106
  Application Notes ......................................................... 108
  Discrete Alphanumeric Recognition ................................... 113
  Continuous Voice Recognition .......................................... 114
  Voice Recognition Testing ............................................... 115
<table>
<thead>
<tr>
<th>Chapter 15. Communications</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emulators Supported</td>
<td>123</td>
</tr>
<tr>
<td>Terminal Emulation Applications</td>
<td>124</td>
</tr>
<tr>
<td>Multiple Host Access</td>
<td>125</td>
</tr>
<tr>
<td>Programming Considerations</td>
<td>125</td>
</tr>
<tr>
<td>Communications Actions</td>
<td>126</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16. Text-to-Speech</th>
<th>129</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating Text Segments</td>
<td>130</td>
</tr>
<tr>
<td>Using Common Text Segments</td>
<td>135</td>
</tr>
<tr>
<td>Programming Considerations</td>
<td>136</td>
</tr>
<tr>
<td>Text-to-Speech Controls</td>
<td>137</td>
</tr>
<tr>
<td>Using Phonetic Input</td>
<td>140</td>
</tr>
</tbody>
</table>

| Chapter 17. Telecommunication Devices for the Deaf | 145 |

<table>
<thead>
<tr>
<th>Chapter 18. Analog Display Service Interface (ADSI)</th>
<th>147</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADSI Support in DirectTalk/2</td>
<td>147</td>
</tr>
<tr>
<td>Creating a DirectTalk/2 ADSI Application</td>
<td>148</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Part 4. Actions</th>
<th>151</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 19. Using Actions</td>
<td>153</td>
</tr>
<tr>
<td>Telephony Actions</td>
<td>153</td>
</tr>
<tr>
<td>Interacting with the Caller</td>
<td>154</td>
</tr>
<tr>
<td>Voice-Related Actions</td>
<td>155</td>
</tr>
<tr>
<td>Creating and Using Variables</td>
<td>155</td>
</tr>
<tr>
<td>Structuring the Voice Application</td>
<td>156</td>
</tr>
<tr>
<td>Interfacing with a Host Application</td>
<td>156</td>
</tr>
<tr>
<td>Collecting Statistics and Logging Messages</td>
<td>157</td>
</tr>
<tr>
<td>Using Voice Messaging</td>
<td>157</td>
</tr>
<tr>
<td>Working with a Database</td>
<td>157</td>
</tr>
<tr>
<td>Telecommunication Devices for the Deaf</td>
<td>158</td>
</tr>
<tr>
<td>Analog Display Services Interface (ADSI) Actions</td>
<td>158</td>
</tr>
<tr>
<td>Creating New Actions</td>
<td>158</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 20. List of Actions</th>
<th>159</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add_Record</td>
<td>160</td>
</tr>
<tr>
<td>Add_Stat_Item</td>
<td>161</td>
</tr>
<tr>
<td>Assign_Resource</td>
<td>162</td>
</tr>
<tr>
<td>Branch</td>
<td>164</td>
</tr>
<tr>
<td>Calculate</td>
<td>166</td>
</tr>
<tr>
<td>Call_Agent_Rel</td>
<td>167</td>
</tr>
<tr>
<td>Call_Extend_Cfg</td>
<td>168</td>
</tr>
<tr>
<td>Call_Extend_Init</td>
<td>170</td>
</tr>
<tr>
<td>Call_Extend_Ref</td>
<td>173</td>
</tr>
<tr>
<td>Call_Extend_Tran</td>
<td>175</td>
</tr>
<tr>
<td>Call_Referral</td>
<td>177</td>
</tr>
<tr>
<td>Call_Transfer</td>
<td>179</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
</tr>
<tr>
<td>Check_Password</td>
<td>182</td>
</tr>
<tr>
<td>Clear_Tones</td>
<td>183</td>
</tr>
<tr>
<td>Close_Database</td>
<td>184</td>
</tr>
<tr>
<td>Comment</td>
<td>185</td>
</tr>
<tr>
<td>Compare_Chars</td>
<td>186</td>
</tr>
<tr>
<td>Compare_Numbers</td>
<td>187</td>
</tr>
<tr>
<td>Concatenate</td>
<td>188</td>
</tr>
<tr>
<td>Connect_Screen</td>
<td>189</td>
</tr>
<tr>
<td>Create_Variable</td>
<td>191</td>
</tr>
<tr>
<td>Delete_Record</td>
<td>192</td>
</tr>
<tr>
<td>Delete_Variable</td>
<td>193</td>
</tr>
<tr>
<td>Delete_Voice</td>
<td>194</td>
</tr>
<tr>
<td>Disconnect_Scr</td>
<td>195</td>
</tr>
<tr>
<td>Free_Resource</td>
<td>196</td>
</tr>
<tr>
<td>Get_ADSI</td>
<td>197</td>
</tr>
<tr>
<td>Get_Length</td>
<td>199</td>
</tr>
<tr>
<td>Get_Messages</td>
<td>200</td>
</tr>
<tr>
<td>Get_Next_Record</td>
<td>204</td>
</tr>
<tr>
<td>Get_Record</td>
<td>206</td>
</tr>
<tr>
<td>Get_Screen_Data</td>
<td>208</td>
</tr>
<tr>
<td>Get_Substring</td>
<td>209</td>
</tr>
<tr>
<td>Get_a_Tone</td>
<td>210</td>
</tr>
<tr>
<td>Get_Tone_String</td>
<td>212</td>
</tr>
<tr>
<td>Get_Voice_Cnt</td>
<td>215</td>
</tr>
<tr>
<td>Get_Voice_Resp</td>
<td>217</td>
</tr>
<tr>
<td>Hang_up_Phone</td>
<td>221</td>
</tr>
<tr>
<td>Link_to_Appl</td>
<td>222</td>
</tr>
<tr>
<td>Log_a_Message</td>
<td>224</td>
</tr>
<tr>
<td>Log_Statistics</td>
<td>225</td>
</tr>
<tr>
<td>Pause</td>
<td>229</td>
</tr>
<tr>
<td>Place_a_Call</td>
<td>230</td>
</tr>
<tr>
<td>Play_Module</td>
<td>233</td>
</tr>
<tr>
<td>Play_Text</td>
<td>236</td>
</tr>
<tr>
<td>Play_Text_String</td>
<td>237</td>
</tr>
<tr>
<td>Play_Voice</td>
<td>238</td>
</tr>
<tr>
<td>Put_Tone_String</td>
<td>240</td>
</tr>
<tr>
<td>Put_User_Info</td>
<td>241</td>
</tr>
<tr>
<td>Receive_TDD</td>
<td>243</td>
</tr>
<tr>
<td>Record_Voice</td>
<td>248</td>
</tr>
<tr>
<td>Refresh_OIA</td>
<td>250</td>
</tr>
<tr>
<td>Remove_Stat_Item</td>
<td>254</td>
</tr>
<tr>
<td>Return_from_Appl</td>
<td>255</td>
</tr>
<tr>
<td>Search_Directory</td>
<td>256</td>
</tr>
<tr>
<td>Search_Screen</td>
<td>258</td>
</tr>
<tr>
<td>Search_String</td>
<td>260</td>
</tr>
<tr>
<td>Send_ADSI</td>
<td>261</td>
</tr>
<tr>
<td>Send_Alert</td>
<td>263</td>
</tr>
<tr>
<td>Send.Keys_to_Scr</td>
<td>264</td>
</tr>
<tr>
<td>Send_TDD</td>
<td>269</td>
</tr>
<tr>
<td>Send_TDD_String</td>
<td>270</td>
</tr>
<tr>
<td>Set_Timeout</td>
<td>271</td>
</tr>
<tr>
<td>Set_Variable</td>
<td>272</td>
</tr>
<tr>
<td>Start_Clock</td>
<td>273</td>
</tr>
<tr>
<td>Stop_Clock</td>
<td>274</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Identification Message</td>
<td>349</td>
</tr>
<tr>
<td>Decadic Dialling</td>
<td>350</td>
</tr>
<tr>
<td>Call Logging</td>
<td>350</td>
</tr>
<tr>
<td>Physical Requirements</td>
<td>350</td>
</tr>
<tr>
<td>Appendix K. Programming Considerations for Norway</td>
<td>351</td>
</tr>
<tr>
<td>Identification Signal</td>
<td>351</td>
</tr>
<tr>
<td>Duration of the Loop Condition</td>
<td>352</td>
</tr>
<tr>
<td>Dial Tone Detection</td>
<td>352</td>
</tr>
<tr>
<td>Dialing with Loop Pulsing</td>
<td>353</td>
</tr>
<tr>
<td>Dialing with DTMF Pulses</td>
<td>353</td>
</tr>
<tr>
<td>Duration of the Call</td>
<td>353</td>
</tr>
<tr>
<td>Repetition of a Call</td>
<td>353</td>
</tr>
<tr>
<td>Identification Signal</td>
<td>354</td>
</tr>
<tr>
<td>Appendix L. Programming Considerations for Portugal</td>
<td>355</td>
</tr>
<tr>
<td>Ringing Detection</td>
<td>355</td>
</tr>
<tr>
<td>Number of Rings</td>
<td>356</td>
</tr>
<tr>
<td>Automatic Answering</td>
<td>356</td>
</tr>
<tr>
<td>Call Termination</td>
<td>356</td>
</tr>
<tr>
<td>Maintaining Connection</td>
<td>357</td>
</tr>
<tr>
<td>Recording Conversation</td>
<td>357</td>
</tr>
<tr>
<td>Dialing</td>
<td>357</td>
</tr>
<tr>
<td>Pulse Dialing</td>
<td>357</td>
</tr>
<tr>
<td>Multifrequency Dialing (DTMF)</td>
<td>358</td>
</tr>
<tr>
<td>Automatic Dialing</td>
<td>358</td>
</tr>
<tr>
<td>Appendix M. Programming Considerations for South Africa</td>
<td>359</td>
</tr>
<tr>
<td>Ringing Detection</td>
<td>359</td>
</tr>
<tr>
<td>Automatic Answering</td>
<td>360</td>
</tr>
<tr>
<td>Call Termination</td>
<td>360</td>
</tr>
<tr>
<td>Maintaining Connection</td>
<td>360</td>
</tr>
<tr>
<td>Recording</td>
<td>361</td>
</tr>
<tr>
<td>Timed Break (Flash) Signal</td>
<td>361</td>
</tr>
<tr>
<td>Dialing</td>
<td>362</td>
</tr>
<tr>
<td>Pulse Dialing</td>
<td>362</td>
</tr>
<tr>
<td>Multifrequency Dialing (DTMF)</td>
<td>362</td>
</tr>
<tr>
<td>Automatic Dialing</td>
<td>363</td>
</tr>
<tr>
<td>Remote Supervisory Functions</td>
<td>363</td>
</tr>
<tr>
<td>Outbound Dialing Exchange Delays</td>
<td>363</td>
</tr>
<tr>
<td>Repertory Dialing</td>
<td>364</td>
</tr>
<tr>
<td>Appendix N. Programming Considerations for Spain</td>
<td>365</td>
</tr>
<tr>
<td>Ringing Signal Detection</td>
<td>365</td>
</tr>
<tr>
<td>Identification Signal</td>
<td>366</td>
</tr>
<tr>
<td>Duration of the Loop Condition</td>
<td>366</td>
</tr>
<tr>
<td>Appendix O. Programming Considerations for Sweden</td>
<td>369</td>
</tr>
<tr>
<td>Answer Function</td>
<td>369</td>
</tr>
<tr>
<td>Ringing Signal</td>
<td>369</td>
</tr>
<tr>
<td>Voice-activated Circuits</td>
<td>370</td>
</tr>
<tr>
<td>Dial Tone</td>
<td>371</td>
</tr>
<tr>
<td>Appendix P. Programming Considerations for Switzerland</td>
<td>373</td>
</tr>
</tbody>
</table>
Figures

1. Voice Application Developer Components .................................. 7
2. Flowchart for the TIME Sample Application ................................. 15
3. Voice Logic Module Insert Line Window ..................................... 63
4. Vocabulary File Formats .................................................. 108
5. Process Flow for VR_Get_String ........................................... 121
6. Sample Action Flow for Switch_Screen .................................. 276

Tables

1. DirectTalk/2 Actions required for TIME ........................................ 16
2. Voice Logic Modules and Voice Segments for TIME ......................... 16
3. Voice segments to be recorded ................................................ 36
4. DirectTalk/2 System Variables ............................................... 47
5. DirectTalk/2 User Variables .................................................. 53
6. Supported languages and file identifiers ..................................... 57
7. Types and Corresponding Operands for the PLAY Function .................. 63
8. Conditions and Their Corresponding Meanings ................................ 64
10. DirectTalk/2 Password Prerequisites ........................................ 86
11. Sampling Rates of Sample Applications ..................................... 117
12. Prerecorded Tones for Voice Recognition Option ............................. 117
13. Text only Segments for Voice Recognition Option ............................ 118
14. DirectTalk/2 Terminal Emulation Support ................................... 123
16. American English Phonetic Alphabet for BeST speech TTS Systems .... 142
17. Configuration settings for Extend Call ....................................... 169
18. Characters Returned for Errors ............................................... 219
19. Record Format ....................................................................... 227
20. TDD Code Conversion ................................................................ 244
21. 3270/5250 Emulator keyboard symbols for the Send_Keys_to_Scr action 267
22. ASCII Terminal Emulator non-attention key symbols for the Send_Keys_to_Scr action ........................................ 268
23. Application Design Form ........................................................ 298
24. Application Design Form (Voice Programs) ................................. 299
25. Application Design Form (Voice Logic Modules) ............................ 300
26. Application Design Form (Voice Segments) .................................. 301
27. Application Design Form (Text Segments) ................................... 302
28. Record Format ....................................................................... 308
29. Information to Change for the WMSCH and WMREC Applications ...... 309
30. Inbound and Outbound Calls .................................................. 317
31. Inbound and Outbound Calls .................................................. 319
32. Inbound and Outbound Calls .................................................. 323
33. Inbound and Outbound Calls .................................................. 327
34. Inbound and Outbound Calls .................................................. 333
35. Inbound and Outbound Calls .................................................. 339
36. Inbound and Outbound Calls .................................................. 343
37. Parameters for Ringing Detection ............................................... 344
Notices

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

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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 of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

Any additional information necessary to achieve interoperability of this program with other programs is available from: IBM United Kingdom Laboratories, Mail Point 151, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Trademarks and Service Marks

The following terms used in this publication are trademarks or service marks of IBM Corporation in the United States or other countries:

- AS/400
- DirectTalk
- DirectTalk/2
- Operating System/2
- Personal System/2
- PS/2
- S/390
- CallPath
- IBM
- MicroChannel
- OS/2
- Portmaster
- S/370

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.
About This Book

The *IBM CallPath DirectTalk/2 Application Development User’s Guide* explains how to use the Voice Application Developer to design, create, edit, test, and debug voice applications. This book also contains a detailed reference of each of these procedures, a description of the optional features of DirectTalk/2, and the actions included with DirectTalk/2.

Who Should Read This Book

This book is written for the person who wants to define and create a voice application. This book explains the steps and the DirectTalk/2 software you need to create the voice application.

The *IBM CallPath DirectTalk/2 Application Development User’s Guide* assumes that you are familiar with Operating System/2 (OS/2) and basic programming concepts.

How to Use This Book

Read the following chapters to become familiar with DirectTalk/2:

- Chapter 1 gives you a general overview of voice applications.
- Chapter 2 discusses the DirectTalk/2 components you use to create, run, and manage a voice application.
- Chapter 3 explains how to open and close the Voice Application Developer and how use the windows it displays.

Read Chapter 4 to learn how to create a simple voice application.

The following chapters provide information to enable you to create your own voice applications:

- Chapter 5 describes procedures for creating, reviewing, and working with variables in voice applications.
- Chapter 6 provides more information about voice applications.
- Chapter 7 provides more information about voice programs.
- Chapter 8 provides more information about voice logic modules.
- Chapter 9 provides more information about voice segments.
- Chapter 10 provides information about call referral and transfer.
- Chapter 11 provides information about the application password security functions.
- Chapter 12 explains how you can write a voice program using the REXX programming language provided with OS/2, in place of using the voice program editor.
The following chapters provide information on the various optional DirectTalk/2 features:

- Chapter 13 describes the Voice Messaging Feature.
- Chapter 14 describes the Voice Recognition Feature.
- Chapter 15 describes the Communications Feature.
- Chapter 16 describes the Text-to-Speech Feature.
- Chapter 17 describes the Telecommunication Devices for the Deaf Feature.
- Chapter 18 describes the Analog Display Service Interface Feature.

The following chapters describe the actions that are included with DirectTalk/2:

- Chapter 19, “Using Actions” on page 153 describes the major task areas of DirectTalk/2 and lists the actions associated with each task.
- Chapter 20, “List of Actions” on page 159 describes each action in detail. The actions are listed in alphabetical order.

The appendixes provide the following reference material:

- Appendix A contains a sample form for recording your voice application design and implementation.
- Appendix B, “Sample Applications” on page 305 contains information about the sample applications that are supplied with DirectTalk/2.
- Appendix C to Appendix Q contain special programming considerations for specific countries.

  If there is no Programming Considerations appendix for your country, it is still your responsibility to ensure that your applications conform to any PTT requirements in your country.

This book also includes a glossary of terms and abbreviations and an index.

**Note:** DirectTalk/2 voice applications can answer incoming calls or make outgoing calls. To avoid the repetitive use of both the terms caller and the person called each time voice applications are discussed, this book uses the term caller to indicate both the person calling the voice application and the person called by the voice application.

---

**Where to Find More Information**

Here, we describe the information available to you, from both IBM and non-IBM sources.

**Other DirectTalk/2 Books and Information**

This book is part of a library of DirectTalk/2 books. To get the most out of your system, refer to the other DirectTalk/2 books as well. The following books are included on the DirectTalk/2 CD-ROM and can be read on-line using the Library Reader program which is also included on the DirectTalk/2 CD-ROM. If you prefer to have hard copy of any of the documents, you can print pages, sections, or whole books from Library Reader. You can also buy the printed books by placing orders through your IBM representative, or the IBM branch office serving your locality.
The on-line versions of these books are also included on the Networking Systems Library CD-ROM.

- IBM CallPath DirectTalk/2 General Information and Planning, GB35-4403
- IBM CallPath DirectTalk/2 Installation Guide, GB34-4406
- IBM CallPath DirectTalk/2 Administrator’s Guide, SB35-4405*
- IBM CallPath DirectTalk/2 Application Development User’s Guide, SB35-4408* (this book)
- IBM CallPath DirectTalk/2 Application Programmer’s Guide, SB35-4404*
- IBM CallPath DirectTalk/2: IBM CallPath DirectTalk/2 Problem Solving Guide, GC33-1548*
- IBM CallPath DirectTalk/2: IBM CallPath DirectTalk/2 ADSI Programmer’s Guide, SC33-1762
- IBM CallPath DirectTalk/2: IBM CallPath DirectTalk/2 National Language Information, SC33-1865

The books marked with * in the list above are also provided in IPF on-line readable format. (Use the OS/2 VIEW command to display these manuals.)

All the programs and utilities provided with DirectTalk/2 also include on-line Help to assist you with the various DirectTalk/2 related tasks.

Non-DirectTalk/2 IBM Documentation Referenced in This Book
- IBM Distributed Console Access Facility documentation
- IBM Real-Time Interface Co-Processor documentation
- IBM Communications Manager/2 documentation
- IBM Personal Communications documentation

Non-IBM Hardware and Software Related Information
The following non-IBM documentation and contacts may also be of use:
- Dialogic Products and Services Guide (Available in hardcopy or CD ROM titled World View 3)
- Dialogic Software Installation Guide
- Dialogic System Release 4.2 Software Installation Reference
- Dialogic Application Note 17 (AN017) Ordering Service and Installing Equipment for T-1 Applications
- XXX Voice SW Reference Guide for OS/2 and XXX Hardware Reference Guide - specific to the hardware being used/purchased. (Where XXX is the particular board level product being used)
- Dialogic Network Hardware Reference, 05-0176-001
- Dialogic Voice Hardware Reference, 05-0147-002
- L&H TTS SW Reference for OS/2 (for Antares)
- VCS ASR SW Reference for OS/2 (for Antares)
Aculab Technical Support

Aculab plc
Lakeside, Bramley Road
Mount Farm
Milton Keynes MK1 1PT
U.K.
Tel: 01908 273800

Voice Control Systems (VCS) Technical Support

Voice Control Systems
14140 Midway, Suite 100
Dallas, Texas
TX 75244
Tel. 214-386-0300
Part 1. Getting Started

This section introduces the concept of voice applications and the Voice Application Developer and takes the user through the creation of a simple application.

**Chapter 1. What is a Voice Application?**
- Voice Programs ........................................ 3
- Voice Logic Modules .................................. 4
- Voice Segments ........................................ 5
- Text Segments ......................................... 5
- Voice Application Variables ........................... 5

**Chapter 2. Understanding the Voice Application Developer** .................. 7

**Chapter 3. Using the Voice Application Developer** ............................ 9
- Starting the Voice Application Developer ................. 9
- Using Voice Application Developer Windows .............. 9
  - Making Selections .................................... 11
  - Canceling Selections .................................. 11
  - Using Function Keys .................................. 11
  - Getting Help .......................................... 12
- Exiting the Voice Application Developer .................. 12

**Chapter 4. Creating a simple voice application** .............................. 13
- Designing the Voice Application ........................ 13
  - Purpose and Requirements ............................. 14
  - Creating a Flowchart ................................ 14
  - Assigning DirectTalk/2 Actions ........................ 15
  - Specifying the Voice Logic Modules and Voice Segments 16
  - Implementing Your Design ............................ 17
- Naming the Voice Application ............................ 17
- Creating the Voice Program ............................. 18
- Creating the Voice Logic Modules ....................... 26
- Creating and Recording Voice Segments ................. 33
- Checking and Testing the Voice Application ............ 36
  - Checking the Completeness of the Voice Application 36
  - Testing the Voice Application ........................ 37
- Preparing to Run the Application ....................... 38
  - Creating the Application Control File ................. 39
Chapter 1. What is a Voice Application?

A DirectTalk/2 system will have at least one voice application associated with each of its telephone lines. These applications control the interaction with the people who call the DirectTalk/2 system or are called by the system. Unlike most applications that interact with the user through a display monitor and keyboard or mouse, DirectTalk/2 applications communicate with users over telephone lines. The applications recognize user's voice input, or dual tone multifrequency (DTMF) tones and text generated by telephones or by Telecommunication Devices for the Deaf (TDD). The output from a voice application is speech, either from prerecorded voice messages or from voice synthesized from text input, text for interaction with TDD, and Analog Display Services Interface (ADSI) scripts to communicate with ADSI devices. See the DirectTalk/2 General Information and Planning Manual for examples of DirectTalk/2 voice applications.

DirectTalk/2 voice applications are composed of a number of components:

- Voice programs
- Voice logic modules
- Voice segments
- Text segments
- Application variables
- ADSI scripts

Every voice application contains at least one voice program, and most applications include a number of voice logic modules and voice or text segments.

Voice Programs

A voice application contains one or more voice programs that define how the voice application operates. Each program consists of a set of DirectTalk/2 or user-written actions. DirectTalk/2 provides a wide selection of actions. Some examples include:

- Wait_for_Call, which waits for and answers an incoming telephone call
- Get_a_Tone, which waits for and receives a DTMF tone from the caller
- Play_Module, which speaks prompts and other information to the caller
- Link_to_Appl, which causes a new voice program to start

See Part 4, “Actions” on page 151 for the complete set of DirectTalk/2 actions.

Each action can take a number of parameters, which determine the precise operation of the action. For example, the Wait_For_Call action can take two parameters. The first specifies the number of times the line should ring before it is answered, and second defines the length of time to wait for a call.

Actions give one of 15 return codes when they complete. These return codes have the names 0 through 11, T1, T2, and HUP. Return codes T1 and T2 usually indicate timeouts. HUP is usually used to indicate that the phone call was disconnected (hung up) before the action completed. The meaning of the other return codes is different for each action. Part of the definition of a voice program step is the setting of the step to go to next for each possible return code for the action.
DirectTalk/2 has a set of over 50 actions. The various optional features, such as Text-to-Speech, are necessary to enable some of these actions. If the actions that are a part of DirectTalk/2 do not suit the needs of your application, you can extend the DirectTalk/2 system by creating new actions. New actions could include verifying a password, performing a scientific calculation, or performing complex data manipulation. You write these new actions in C. See the DirectTalk/2 Application Programmer’s Guide for details about writing new actions.

Voice programs are written using the Voice Application Developer (see Chapter 2, “Understanding the Voice Application Developer” on page 7) or in the REXX language (see Chapter 12, “DirectTalk/2 REXX Environment” on page 89). Both types of program can be run from and managed by the DirectTalk/2 Application Manager, and those written in REXX can also be run using the T-REXX command.

Voice Logic Modules

Voice Logic Modules are the component of a voice application that control the playing (“speaking”) of the messages that are spoken to a caller. These messages are a major part of a voice application and can include:

- Welcome messages
- Requests that the caller select a service
- Requests that the caller provide some data
- Confirmation that an action has occurred
- Information retrieved from local or host databases
- Information retrieved from host terminals

It is often necessary to concatenate phrases, data, or both to make up a complete message, such as in a response to a customer query about a bank account balance. The response to such a query could be:

- “Your account is 1 234 dollars in credit.”
- “Your account is 1 234 dollars overdrawn.”

Note: DirectTalk/2 will expand “1 234” to “one thousand two hundred and thirty-four”.

This message is made up of three parts:

Your account is This part of the message is recorded by the application developer. Such portions of messages are called voice segments.

1 234 dollars This portion of the message is generated by the DirectTalk/2 system from recorded voice segments. Its value depends on a value read from a host screen.

in credit or overdrawn These are again recorded voice segments; only one is spoken, depending on the account balance value.

Every voice logic module used by a voice application lists each part of the message that is to be played. For each part of the message, the voice logic module indicates what is to be played (a voice segment or an item of data generated by the application), and how it is to be played (for example, as a date or as a currency value). It is also possible to add logic to indicate when a particular portion of a message is to be played. For the bank account example above, the voice logic module would indicate:
- Play the recorded voice segment ‘balance_start’.
- Play ‘balance’ as a currency value.
- If ‘balance’ is positive, play recorded voice segment ‘credit’.
- If ‘balance’ is negative, play recorded voice segment ‘debit’.

If the optional Text-to-Speech feature is installed, a voice logic module can also indicate that a previously-entered piece of text (a text segment) or some other text data should be synthesized into spoken words.

Voice logic modules are created using the VAD, but they can be used by both VAD and REXX voice programs.

### Voice Segments

A voice segment is a piece of recorded voice. Voice segments are played from within voice logic modules and can be combined with other spoken data. Except when the Text-to-Speech feature is used, all DirectTalk/2 speech is made up of a combination of voice segments. DirectTalk/2 has a series of system voice segments that are used in combination to speak, for example, numbers and dates.

Voice segments are created using the VAD, but they can be used by both VAD and REXX voice programs.

### Text Segments

**Note:** Text segments can only be used if the optional Text-to-Speech (TTS) feature has been installed.

A text segment is a piece of typed text which the Text-to-Speech function will attempt to speak according to a set of built-in rules. TTS takes the words and the punctuation into account to determine pronunciation, stress, and spacing. It has the ability to parse certain abbreviations; for example, “etc.” is spoken as “et cetera”. Users can add new words and the associated pronunciations to the dictionary. It is also possible to add control words to the text that indicate, for example, that the speech rate should be increased or decreased or that a word should be spelled as individual characters.

Text segments, like voice segments, are played from within voice logic modules.

Text segments are created using the VAD, but they can be used by both VAD and REXX voice programs.

### Voice Application Variables

Voice applications can use variables to hold data as a string of characters. The way in which the variable data is used depends on the context. For example, a variable may, depending on the context, hold a number, a date, a voice segment name, or a line of text. All variables are named, with a name no longer than 16 characters.

See Chapter 12, “DirectTalk/2 REXX Environment” on page 89 for specific information on using variables with programs written in REXX.
Chapter 2. Understanding the Voice Application Developer

IBM CallPath DirectTalk/2 consists of three components you use to create and run all or part of a voice application (see Chapter 12, “DirectTalk/2 REXX Environment” on page 89 for information on writing voice programs in REXX.). These components are:

- The **Voice Application Developer** (VAD), which you use to create, edit, and test the voice application. The use of the VAD is described in detail in this book.

- The **Setup** program, which you use to configure your system and to define the applications that are started on a phone line. The use of Setup is described in *IBM CallPath DirectTalk/2 Installation Guide*.

- The **Application Manager**, which executes and runs the voice application in a production environment. You interact with the Application Manager through the **Node Manager** user interface. The use of the Node Manager is described in *IBM CallPath DirectTalk/2 Administrator’s Guide*.

![Figure 1. Voice Application Developer Components](image-url)

Figure 1 shows the components of the Voice Application Developer.
As described in Chapter 1, “What is a Voice Application?” on page 3 voice applications consist of a number of components:

- Voice programs
- Voice logic modules
- Voice segments
- Text segments
- Variables
- ADSI scripts

The VAD has corresponding components to create each part of a voice application:

**Voice program editor**
To create a voice program

**Voice logic module editor**
To create voice logic modules

**Voice segment editor**
To create voice segments

**Text segment editor**
To create text segments.

**Choices on the Options menu**
To create variables

Chapter 3, “Using the Voice Application Developer” on page 9, Chapter 4, “Creating a simple voice application” on page 13, and Chapter 5, “Variables for Voice Applications” on page 43, guide you through the use of the VAD to create the various components.

**Note:** Unless it is absolutely necessary, you should not develop voice applications on a system that is running production voice applications. The development activity may seriously degrade the performance of the production system and defective development applications can cause the entire system to fail.
Chapter 3. Using the Voice Application Developer

This chapter explains how to start and end the Voice Application Developer, and how to use the windows it displays.

Starting the Voice Application Developer

To start using the Voice Application Developer program, do the following:

1. Ensure the Voice System is started.
   If it isn’t, start it by double-clicking on the Voice System program object.
   **Note:** If you are developing applications remotely, ensure that the Voice System is started on the remote node.

2. Double-click on the Voice Application Developer program object.

DirectTalk/2 displays the Voice Application Developer window.

---

Using Voice Application Developer Windows

The Voice Application Developer consists of a series of windows that you use to define your voice application.
DirectTalk/2 windows contain:

1 **Window title**  DirectTalk/2 displays the title of the window at the top of each window.

2 **Action bar**  DirectTalk/2 displays an action bar below the title of the window on all task windows. The action bar shows you the items you can select on that task window.

3 **Function key area**  DirectTalk/2 displays the actions you can complete by pressing or selecting a function key in the function key area at the bottom of each window (except help windows). You can only press or select the function keys in the current window.

4 **Message area**  DirectTalk/2 displays messages in a message area just above the function key area on all windows (except help windows).

If the information in a window is longer than the window, the word *More* and a down arrow (↓) appear in the upper right corner of the window. To scroll the window down, do one of the following:

- Press the Page Down key.
- Click on the ‘page down’ arrow with mouse button 1.
While you are scrolling through the windows, the word More, a down arrow, and an up (↑) arrow appear in the upper right corner of the window.

When you reach the last window of the information, DirectTalk/2 removes the down arrow and displays an up arrow. To scroll the window up, do one of the following:

- Press the Page Up key.
- Click on the ‘page up’ arrow with mouse button 1.

**Making Selections**

To select an action in the action bar, you can use the keyboard or the mouse.

- Use the Tab or cursor keys to move the highlight block to the action and press Enter.
- Click on the action with mouse button 1.

When you select an action on the action bar, DirectTalk/2 displays a pull-down with choices related to the action you selected. While DirectTalk/2 displays the pull-down, you can only use the actions in the pull-down. For example, if you select a pull-down and the action you want is not listed in it, you can select another pull-down on the action bar.

If you select an item on the pull-down that is followed by an ellipsis (…), DirectTalk/2 removes the pull-down and displays a window. You use this window to type the information that DirectTalk/2 needs to complete the action.

While DirectTalk/2 displays the window, you can only specify the information in that window. For example, you cannot select an action on the action bar in the previous window until DirectTalk/2 removes the current window.

**Canceling Selections**

To cancel a selection, press Esc or F12. For example, you can press F12 if you do not want to perform an action you selected.

If DirectTalk/2 displays a pull-down, DirectTalk/2 removes the pull-down and takes you to the action bar. If DirectTalk/2 displays a window, DirectTalk/2 closes the window and takes you to the window from which you selected the action.

**Using Function Keys**

To select a function key, do one of the following:

- Press the function key.
- Use the mouse to move the mouse pointer to the function key and press mouse button 1.

The following keys perform the same functions throughout the Voice Application Developer.

**F1=Help**

- Look at help for the current window.

**F3=Exit**

- Exit from the Voice Application Developer.

**F12=Cancel**

- Return to the previous window.
Getting Help

You can get help on individual windows, the action bar in the window, and the pull-downs in the action bar. To get help on a window, press F1.

In addition, each task window contains a Help pull-down with the following items:

**Help for help**
To get instructions on what types of help are available and how to display help.

**Keys help**
To get a list of the common function keys and the actions they perform, and to get help on making selections.

**Extended help**
To get help on the main task window and its action bar.

**Copyright**
To display the version and release date of DirectTalk/2.

Exiting the Voice Application Developer

Press F3 from any task window.

DirectTalk/2 displays a confirmation window for you to verify that you want to end the Voice Application Developer session.

<table>
<thead>
<tr>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are you sure you want to exit?</td>
</tr>
<tr>
<td>To exit, press F3.</td>
</tr>
<tr>
<td>To cancel, press F12.</td>
</tr>
</tbody>
</table>

- To end the session, press F3.
- If you do not want to end the session, press Esc or F12.
Chapter 4. Creating a simple voice application

The purpose of the TIME application is to ask a caller if they want to hear the time, the date, or both, and to respond with the appropriate information. The application must:

1. Wait for and answer an incoming telephone call
2. Play a menu to the caller asking if the caller wants to hear the time, date, or both
3. Play the time, date, or both, based on the caller’s response
4. End the call
5. Wait for the next call

After you learn the basic steps for creating an application, you can use the TIME application as a model for creating your own applications.

Creating the TIME application requires the following tasks to be completed:

1. Design the voice application.
2. Name the voice application.
3. Create the voice program for the voice application.
4. Create the voice logic modules for the voice application.
5. Create and record the voice segments for the voice application.
6. Check and test the voice application.
7. Prepare to run the application (optional).

The remainder of this chapter provides the steps you must follow to complete these tasks.

Designing the Voice Application

The time you spend in designing a voice application is as important as the time you spend implementing it. Not only does it take less time to implement a well-designed application, but the application runs more efficiently and better meets the user’s needs.

The steps to design the voice application are:

1. Define and document the purpose and requirements of the voice application.
2. Create a flowchart for the voice application.
3. Assign DirectTalk/2 actions or create your own actions to perform the voice application activities.
4. Specify the voice logic modules and voice segments for the voice application.

You may find it helpful to use some type of standard layout to document your design. Appendix A, “Sample Voice Application Form” on page 297 contains a series of forms designed for this purpose.

See the DirectTalk/2 General Information and Planning Manual for more advice on designing voice applications.
Purpose and Requirements

The purpose and requirements for the TIME application are:

Purpose
To ask a caller if they want to hear the time, the date, or both, and to respond with the appropriate information.

Requirements
The time application must perform the following functions:
- Wait for an incoming call
- Answer the call
- Play a menu to the caller asking if the caller wants to hear the time, date, or both
- Play the time, date, or both, based on the caller’s response
- End the call
- Wait for the next call

Creating a Flowchart
The flowchart for the time application is shown in Figure 2 on page 15.
Assigning DirectTalk/2 Actions

The next step is to choose the DirectTalk/2 actions to perform the activities described in the TIME flow chart. All the actions provided with DirectTalk/2 are described in Part 4, “Actions” on page 151. If you want to create a more complex application you may find that you need to create your own actions to supplement those supplied with DirectTalk/2. Detailed information on how to create your own actions is supplied in the IBM CallPath DirectTalk/2 Application Programmer’s Guide.
The actions required for the TIME program are shown in Table 1:

<table>
<thead>
<tr>
<th>Flowchart activity</th>
<th>DirectTalk/2 action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait for and answer call</td>
<td>Wait_for_Call</td>
</tr>
<tr>
<td>Play welcome</td>
<td>Play_Module</td>
</tr>
<tr>
<td>Play menu</td>
<td>Play_Module</td>
</tr>
<tr>
<td>Get callers choice</td>
<td>Get_a_Tone</td>
</tr>
<tr>
<td>Play time, date, or both</td>
<td>Play_Module</td>
</tr>
<tr>
<td>Play Goodbye</td>
<td>Play_Module</td>
</tr>
<tr>
<td>End call</td>
<td>Hang_up_Phone</td>
</tr>
<tr>
<td>Play invalid I/P warning</td>
<td>Play_Module</td>
</tr>
<tr>
<td>Play last repeat warning</td>
<td>Play_Module</td>
</tr>
</tbody>
</table>

Specifying the Voice Logic Modules and Voice Segments

The TIME application uses several Play_Module actions, each of which require one or more voice modules to play. Each voice logic module also requires one or more voice segments. Some of these voice segments, such as the time and date, are provided by the system, the rest need to be recorded, or supplied from another source. Each module and segment also requires a unique name of up to 15 characters. Table 2 shows the voice logic modules and voice segments required for the TIME application.

<table>
<thead>
<tr>
<th>Flowchart activity</th>
<th>DirectTalk/2 voice module</th>
<th>DirectTalk/2 voice segment</th>
<th>system or user</th>
<th>Voice segment text</th>
</tr>
</thead>
<tbody>
<tr>
<td>play welcome</td>
<td>hello</td>
<td>hello</td>
<td>user</td>
<td>Hello, welcome to the sample application.</td>
</tr>
<tr>
<td>play menu</td>
<td>choices</td>
<td>menu</td>
<td>user</td>
<td>Press one for time, two for date, three for both, or zero or star to exit.</td>
</tr>
<tr>
<td>Play last repeat warning</td>
<td>choices</td>
<td>warning</td>
<td>user</td>
<td>If you do not make a valid selection in (timeout value) seconds you will be disconnected</td>
</tr>
<tr>
<td>Play time</td>
<td>say_time</td>
<td>time_is system_time</td>
<td>user</td>
<td>The time is (system time)</td>
</tr>
<tr>
<td>Play date</td>
<td>say_date</td>
<td>date_is system_date</td>
<td>user</td>
<td>Today is (system date)</td>
</tr>
<tr>
<td>Play both</td>
<td>both</td>
<td>date_is system_date</td>
<td>user</td>
<td>Today is (system date) The time is (system date)</td>
</tr>
<tr>
<td>Play goodbye</td>
<td>goodbye</td>
<td>goodbye</td>
<td>user</td>
<td>Good bye. Thank you for using TIME.</td>
</tr>
<tr>
<td>Play invalid I/P warning</td>
<td>invalid</td>
<td>the_key last_dmf_tone invalid</td>
<td>user</td>
<td>The key you pressed (last key press) is not a valid response</td>
</tr>
</tbody>
</table>
Implementing Your Design

When your design is complete and documented it needs to be implemented as an application. The following sections describe in detail how to use the Voice Application Developer to create the TIME application.

If you would like more information about the fields you are completing, or about other functions that are available from the Voice Application Developer, press <F1> in any window to get the associated online help.

Naming the Voice Application

The TIME application needs to be defined to the DirectTalk/2 system with a name. To create this name, first open the Voice Application Developer as described in Chapter 3, “Using the Voice Application Developer” on page 9.

1. To name the voice application, select New in the File pull-down in the Voice Application Developer window to display the New Application window.

2. Type TIME to specify the name of the new voice application.
   
   Note: A voice application name can have a maximum of five characters.

3. Press <Enter> to complete the creation of the name.

The name TIME is added to the Voice Application Developer window title to show that this is the application you are now working on.
Creating the Voice Program

The actions you assigned to your design flowchart activities now need to be entered as steps in the TIME voice program. For each step you must specify:

- A unique number for the step
- The name of the action the step performs
- A comment to describe the step
- The details for the step, including the values for any required input or output parameters, and the step number to run for each result (return code) the action can generate.

Use the following procedure to enter this information with the help of the Voice Program Editor, which is part of the VAD.

1. Select Voice program editor from the Editors pull-down in the Voice Application Developer window.

Since the voice program does not contain any steps, DirectTalk/2 automatically displays the Insert window. Use this window to select the first action you want to use.

2. Select the Wait_for_Call action and then press <Enter> to display the Step Details window in which you enter all the parameter values and the step numbers for the return codes for the Wait_for_Call action.
All the fields you need to provide information for are highlighted. You can ignore all the other fields in the window.

3. Accept the default value of 10 for the step number.

4. Type Begin call loop in the Comment field, to describe what the action is doing in your program.

5. Accept the default value of '1' for Parameter 1, Number of rings.

   This parameter specifies the number of rings before the application answers the telephone.

6. Type '3' in the Value field for Parameter 2, Wait time in minutes.

   This specifies the number of minutes the action will wait without receiving a call before it returns control to the voice program.

7. Type 11 in the Go to Step field for Return Code 0, Phone answered.

   This indicates that when the action answers the telephone, processing should continue with step 11.

8. Type 10 in the Go to Step field for Return Code 1, Wait expired.

   This indicates that if no call is received within the time specified in Parameter 2, processing continues with Step 10 and the Wait_for_Call action is repeated.

9. Type 40 in the Go to Step field for Return Code 2, Application Stop.

   This indicates that if a command is issued to stop the application, processing continues with Step 40.

10. Press <Enter> to complete the input and to continue creating the voice program.

    DirectTalk/2 automatically displays the Insert window again (see picture and description on page 18) ready for you to select your next action.

11. Select the Play_Module action and press <Enter>. 
DirectTalk/2 displays the **Step Details** window for the Play_Module action. Use this window to enter the parameter values and steps for the return codes for the Play_Module action which will perform the 'Play Welcome' step of your program.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Type</th>
<th>Comment</th>
<th>Parameter Description</th>
<th>Value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0011</td>
<td>Play_Module</td>
<td>SYSTEM</td>
<td>Say hello</td>
<td>Voice module name</td>
<td>'hello'</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Force play</td>
<td>no</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VR stop vocabulary</td>
<td>0</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Go to Step</th>
<th>Description</th>
<th>Return Code</th>
<th>Go to Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>Play complete</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>Key detected</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>Word detected</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>Voice detected</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>Unknown response</td>
<td>T1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>No VR line</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>No text line</td>
<td>HUP 30</td>
<td>Caller hung up</td>
<td></td>
</tr>
</tbody>
</table>

Shift+F1=Action Help  F1=Help  F2=List  F12=Cancel

12. Overtype the default step number with **11**.

13. Type **Say hello** in the **Comment** field to describe what the action is doing in your program.

14. Type the name you defined for the voice logic module, **'hello'**, in the **Value** field for Parameter 1.

   The names of voice logic modules must be surrounded with single quotes when they are entered as parameters.

15. Accept the default value in the **Value** field for Parameter 2, Force play.

   This indicates that a caller does not have to listen to the whole message after responding by pressing a telephone button. Since this value is a literal, you must enclose it in single quotes.

16. Accept the default value in the **Value** field for Parameter 3, VR stop vocabulary.

   This indicates you are not using the voice cutthrough function of voice recognition. Single quotes should be used here as well.

17. Type **12** in the **Go to Step** field for Return Code 0, Play complete.

   This indicates that after the action plays the entire voice logic module without any interruption, processing should continue with step 12.

18. Type **12** in the **Go to Step** field for Return Code 1, Key detected.

   This indicates that if the caller interrupts the voice logic module by pressing a button on the telephone key pad, processing should continue with step 12.

19. Type **12** in the **Go To Step** fields for Return Codes 2, 3, 4, 5, and 6.
Although these return codes will not be returned, because we are not using voice recognition, you need to complete all the highlighted fields for the action.

20. Type 30 in the **Go to Step** field for Return Code HUP, Caller hung up.

This indicates that if the caller interrupts the voice logic module by hanging up, processing should continue with step 30 which hangs up our phone and waits for another call.

21. Press **<Enter>** to complete the input and to continue the voice program definition.

DirectTalk/2 displays the **Insert** window again (see picture and description on page 18), ready for you to select your next action.

22. Select **Play_Module** again, and enter the following values for the 'Play Menu' step of your program in the **Step Details** window for the Play_Module action (see picture and description on page 20):

<table>
<thead>
<tr>
<th>Step</th>
<th>Comment</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
<th>Return Codes 0-6</th>
<th>Return Code HUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Ask caller for choice</td>
<td>'choices'</td>
<td>'no'</td>
<td>'0'</td>
<td>13</td>
<td>30</td>
</tr>
</tbody>
</table>

23. Press **<Enter>** to complete the input and to continue the voice program definition.

DirectTalk/2 displays the **Insert** window again (see picture and description on page 18), ready for you to select your next action.

24. Select the **Get_a_Tone** action and press **<Enter>**.

DirectTalk/2 displays the **Step Details** window for the Get_a_Tone action. Use this window to enter the steps for the return codes for the Get_a_Tone action which will detect the callers response. (Note that this action has no input parameters.)
25. Overtype the default step number with 13.

26. Type **Get the caller’s choice** in the **Comment** field to describe what the action is doing in your program.

27. Type 23 in the **Go to Step** field for Return Code 0, 'Key 0 pressed', and Return Code 10, 'Key * pressed'.

   This indicates that if the caller presses 0, or * on the key pad, we assume that the caller wants to exit the TIME application and we say ‘goodbye’ and wait for another call.

28. Type 20 in the **Go To Step** fields for Return Code 1, 'Key 1 pressed'.

   This response means the caller wants us to play the time module, which we have defined as step 20.

29. Type 21 in the **Go To Step** fields for Return Code 2, 'Key 2 pressed'.

   This response means the caller wants us to play the date module, which we have defined as step 21.

30. Type 22 in the **Go To Step** fields for Return Code 3, 'Key 3 pressed'.

   This response means the caller wants us to play the both time and date module, which we have defined as step 22.

31. Type 14 in the **Go To Step** fields for Return Codes 4 to 9, and 11.

   All these responses indicate that an invalid key was pressed and we should therefore play the 'invalid' module which we have defined as step 14.

32. Type -1 in the **Go To Step** fields for Return Code T1, 'Time out'.

   This means that if no response is received within the Wait time which will be set in the 'choices' voice logic module, the program will go back to the previous Play_Module and play the 'choices' module again.

33. Type 23 in the **Go To Step** fields for Return Code T2, 'Last repeat'.

   When you create the 'choices' voice logic module, you will be asked to define the number of times the module should be repeated if the caller does not respond, or gives invalid input. When this repeat value has been reached, code T2 is returned and the voice program will go to step 23 to say 'goodbye' and exit.

34. Type 30 in the **Go to Step** field for Return Code HUP, Caller hung up.

   This indicates that if the caller interrupts the action by hanging up, processing should continue with step 30 which hangs up our phone and waits for another call.

35. Press **<Enter>** to complete the input and to continue the voice program definition.

   DirectTalk/2 displays the **Insert** window again (see picture and description on page 18), ready for you to select your next action.

36. Select **Play_Module** again, and enter the following values for the ‘Play invalid I/P warning’ step of your program in the **Step Details** window for the Play_Module action (see picture and description on page 20):

<table>
<thead>
<tr>
<th>Step</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>Repeat invalid response</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>'invalid'</td>
</tr>
</tbody>
</table>
Parameter 2  'no'
Parameter 3  '0'
Return Codes 0-6  12
Return Code HUP  30

37. Press <Enter> to complete the input and to continue the voice program definition.

DirectTalk/2 displays the Insert window again (see picture and description on page 18), ready for you to select your next action.

38. Select Play_Module again, and enter the following values for the 'Play time' step of your program in the Step Details window for the Play_Module action (see picture and description on page 20):

Step 20
Comment  Tell the time
Parameter 1  'say_time'
Parameter 2  'no'
Parameter 3  '0'
Return Codes 0-6  12
Return Code HUP  30

39. Press <Enter> to complete the input and to continue the voice program definition.

DirectTalk/2 displays the Insert window again (see picture and description on page 18), ready for you to select your next action.

40. Select Play_Module again, and enter the following values for the 'Play date' step of your program in the Step Details window for the Play_Module action (see picture and description on page 20):

Step 21
Comment  Play the date
Parameter 1  'say_date'
Parameter 2  'no'
Parameter 3  '0'
Return Codes 0-6  12
Return Code HUP  30

41. Press <Enter> to complete the input and to continue the voice program definition.

DirectTalk/2 displays the Insert window again (see picture and description on page 18), ready for you to select your next action.

42. Select Play_Module again, and enter the following values for the 'Play both' step of your program in the Step Details window for the Play_Module action (see picture and description on page 20):

Step 22
Comment  Play both time and date
Parameter 1  'both'
Parameter 2  'no'
Parameter 3  '0'
Return Codes 0-6  12
Return Code HUP  30

43. Press <Enter> to complete the input and to continue the voice program definition.
DirectTalk/2 displays the Insert window again (see picture and description on page 18), ready for you to select your next action.

44. Select Play_Module again, and enter the following values for the ‘Play goodbye’ step of your program in the Step Details window for the Play_Module action (see picture and description on page 20):

<table>
<thead>
<tr>
<th>Step</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>say goodbye</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>'goodbye'</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>'no'</td>
</tr>
<tr>
<td>Parameter 3</td>
<td>'0'</td>
</tr>
<tr>
<td>Return Codes 0-6</td>
<td>30</td>
</tr>
<tr>
<td>Return Code HUP</td>
<td>30</td>
</tr>
</tbody>
</table>

45. Select the Hang_up_Phone action and press <Enter>.

DirectTalk/2 displays the Step Details window for the Hang_up_Phone action. Use this window to enter the parameter values and the steps for the return codes for the Hang_up_Phone action.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0030</td>
<td>Hang_up_Phone</td>
<td>SYSTEM</td>
<td>End of call</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Offhook</td>
<td>'no'</td>
<td>O</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Go to Step Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Phone Hung up</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>T1</td>
</tr>
<tr>
<td>5</td>
<td>T2</td>
</tr>
<tr>
<td>6</td>
<td>HUP</td>
</tr>
</tbody>
</table>

Shift+F1=Action Help  F1=Help  F2=List  F12=Cancel

46. Overtype the default step number with 30.

47. Type End of call in the Comment field, to describe the effect of this action in your program.

48. Accept the value of 'no' for Parameter 1, Offhook to indicate that the telephone should be put on hook ready to wait for another call.

49. Type 10 in the Go to step field for Return Code 0, Phone hung up. This specifies that when the call is disconnected, processing should continue by looping back to step 10 to and issuing another Wait_for_Call action ready for another incoming call.

50. Press <Enter> to complete the input and to continue the voice program definition.

DirectTalk/2 displays the Insert window again.
Although all the actions described in the design of the TIME application have now been entered, the application needs an extra action to enable it to stop when required.

51. Select the **Return_from_Appl** action and then press <Enter>.

DirectTalk/2 displays the **Step Details** window for the Return_from_Appl action. Use this window to define the parameter value for the Return_from_Appl action. (Note that this action has no return codes.)

```
Step Details: 0040 0031-9999

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0040</td>
<td>Return_from_Appl</td>
<td>SYSTEM</td>
<td>Application shutdown</td>
</tr>
</tbody>
</table>

Parameter | Description | Value | Status |
-----------|-------------|-------|--------|
1          | Return code | 0     | O      |
2          |             |       |        |
3          |             |       |        |
4          |             |       |        |

Return Code | Go to Code | Return Code | Go to Code |
------------|-----------|-------------|-----------|
0           | 8         | 1           | 9         |
1           |           | 2           | 10        |
2           |           | 3           | 11        |
3           |           | 4           | T1        |
4           |           | 5           | T2        |
5           |           | 6           | HUP       |
6           |           | 7           |           |

Shift+F1=Action Help  F1=Help  F2=List  F12=Cancel
```

52. Accept the default value of 40 for the step number.

53. Type **Application shutdown** in the **Comment** field.

54. Accept the value of '0' for Parameter 1, Return code.

55. Press <Enter> to complete the input.

DirectTalk/2 redisplay the **Insert Step** window.

56. As you have no more actions to enter, Press <F12> or <Esc> to close the **Insert Step** window.

DirectTalk/2 displays the complete TIME voice program in the **Voice Program Editor** window.
Creating the Voice Logic Modules

During the design phase we decided that we required a number of voice logic modules to play two voice segments. This section describes how to use the Voice Logic Module Editor, which is part of the VAD, to create these voice logic modules. To create the voice logic modules for the TIME application

1. Select Voice logic module editor from the Editors pull-down in the Voice Application Developer window.

DirectTalk/2 displays the Voice Logic Module Editor window.
2. Select **Add or Change** from the **Edit** pull-down in the **Voice Logic Module Editor** window to display the **Add or Change Voice Logic Module** window.

<table>
<thead>
<tr>
<th>Add or Change Voice Logic Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice Logic Module.............. <strong>hello</strong></td>
</tr>
<tr>
<td>F1=Help  F2=List  F12=Cancel</td>
</tr>
</tbody>
</table>

3. Type **hello**, the name we defined for the first voice logic module in the **TIME** application.

4. Press **<Enter>** to display the **Add Voice Logic Module** window.

<table>
<thead>
<tr>
<th>Add Voice Logic Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voice Logic Module ............... <strong>hello</strong></td>
</tr>
<tr>
<td>Repeat Value ................. <strong>3</strong></td>
</tr>
<tr>
<td>Time-out value ............... <strong>10</strong></td>
</tr>
<tr>
<td>Type comment text: <strong>Hello, welcome to the sample application</strong></td>
</tr>
<tr>
<td>F1=Help  F12=Cancel</td>
</tr>
</tbody>
</table>

5. Accept the value of 3 for the **Repeat Value**.

   This value specifies that the voice segment will be repeated up to 3 times if the caller does not respond to a prompt for input. This voice logic module is not a prompt, however, so the value in this case is ignored.

6. Accept the value of 10 for the **Time-out** value. This value specifies that the system waits for 10 seconds for a caller response. This parameter defaults to 10. This voice logic module is not a prompt, so the value in this case is also ignored.

7. In the **Type comment text** field, enter the text that will be spoken when this module is played. In the case of the 'hello' voice logic module the text is:

   **Hello, welcome to the sample application.**

8. Press **<Enter>** to display the **Insert Line** window.

<table>
<thead>
<tr>
<th>Insert Line (Remaining Space: 1964 Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>PLAY</td>
</tr>
<tr>
<td>F1=Help</td>
</tr>
</tbody>
</table>

In this window you insert the details of the actions the voice logic module is to perform

9. Enter the details of the 'hello' voice logic module action

   - Type **PLAY** in the **Function** field as the 'hello' voice logic module requires a voice segment to be played.
   - Type **SEG** in the **Type** field to specify that this is to be a voice segment
- In the **Operand One** field, type the name of the segment that is to be played. In the case of the 'hello' module this is 'hello'.

  Use single quotes when you specify voice segment names.

10. Press `<Enter>` to complete the input.

    DirectTalk/2 displays the **Insert Line** window again.

11. As we have no more functions for the 'hello' voice logic module to perform, press `<Esc>` or `<F12>` to close the **Insert Line** window.

    The content of the voice logic module is displayed in the **Add or Change Voice Logic Module** window.

12. Press `<F4>` to save the 'hello' voice logic module and return to the **Voice Logic Module Window** window. (See picture and description on page 27.)

    Select **Add or Change** from the **Edit** pull-down in the **Voice Logic Module Editor** window to display the **Add or Change Voice Logic Module** window.

13. Overtype the input field with the name of your second voice logic module - choices.

14. Press `<Enter>` to display the **Add Voice Logic Module** window again. (See picture and description on page 27.)

15. Accept the value of 3 for the **Repeat Value**.

    This value specifies that the voice segment will be repeated up to 3 times if the caller does not respond to a prompt for input.

16. Accept the value of 10 for the **Time-out value**. This value specifies that the system waits for 10 seconds for a caller response.

17. As this voice logic module does more than play a simple text segment, type the function of the module in the **Type comment text** field. In the case of the 'choices' voice logic module this is:

    to play a menu of choices to the caller:

18. Press `<Enter>` to display the **Insert Line** window. (See picture and description on page 27.)
19. The first function of the 'choices' module is to play the 'menu' voice segment to the caller. Enter the details of this function as follows:

Function: PLAY
Type: SEG
Operand One: 'menu'
Cond: (leave blank)
Operand Two: (leave blank)

20. Press <Enter> to complete the input and display the Insert Line window again.

21. We have defined that the 'choices' segment should only be repeated three times if the caller does not provide a valid response within the allowed time, and we want to warn them if this is the last repeat. The next function of the 'choices' module is to determine if this is the last repeat.

To do this we use a conditional statement which uses the 'last_repeat' variable which is set by the system to decide if this is the last repeat of the 'choices' segment. Enter the following values to perform this function:

Function: IF
Type: NUM
Operand One: last_repeat
Cond: EQ
Operand Two: '1'

This means that the immediately following statement in this voice logic module should only be performed if this is the last repeat.

22. Press <Enter> to complete the input and display the Insert Line window again.

23. If this is the last repeat we want to play the 'last repeat warning' to the user. This warning consists of three parts:

   a. The 'warning' voice segment
   b. The system variable timeout_value
   c. The 'seconds' voice segment

   The next function of the 'choices' module is therefore to play the 'warning' text segment. Enter the values for this as follows:

Function: PLAY
Type: SEG
Operand One: 'warning'
Cond: (leave blank)
Operand Two: (leave blank)

24. Press <Enter> to complete the input and display the Insert Line window again.

25. As the conditional 'IF' statement above only applies to the immediately following statement in the voice logic module, we have to enter it again before each of the other parts of the warning. Enter the following:

Function: IF
Type: NUM
Operand One: last_repeat
Cond: EQ
Operand Two: '1'

26. Press <Enter> to complete the input and display the Insert Line window again.
27. The second part of the warning is to play the number that is held in the timeout_value system variable. Enter this function as follows:

<table>
<thead>
<tr>
<th>Function</th>
<th>PLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>NUM</td>
</tr>
<tr>
<td>Operand One</td>
<td>timeout_value</td>
</tr>
<tr>
<td>Cond</td>
<td>(leave blank)</td>
</tr>
<tr>
<td>Operand Two</td>
<td>(leave blank)</td>
</tr>
</tbody>
</table>

28. Press <Enter> to complete the input and display the Insert Line window again.

29. Enter the condition test once again:

<table>
<thead>
<tr>
<th>Function</th>
<th>IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>NUM</td>
</tr>
<tr>
<td>Operand One</td>
<td>last_repeat</td>
</tr>
<tr>
<td>Cond</td>
<td>EQ</td>
</tr>
<tr>
<td>Operand Two</td>
<td>‘1’</td>
</tr>
</tbody>
</table>

30. Press <Enter> to complete the input and display the Insert Line window again.

31. Enter the final part of the warning as follows:

<table>
<thead>
<tr>
<th>Function</th>
<th>PLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SEG</td>
</tr>
<tr>
<td>Operand One</td>
<td>‘seconds’</td>
</tr>
<tr>
<td>Cond</td>
<td>(leave blank)</td>
</tr>
<tr>
<td>Operand Two</td>
<td>(leave blank)</td>
</tr>
</tbody>
</table>

32. Press <Enter> to complete the input and display the Insert Line window again.

33. As we have no more functions for the 'choices' voice logic module to perform, press <Esc> or <F12> to close the Insert Line window.

The content of the 'choices' voice logic module is displayed in the Add or Change Voice Logic Module window.

<table>
<thead>
<tr>
<th>Add or Change Voice Logic Module - choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>PLAY</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>PLAY</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>PLAY</td>
</tr>
<tr>
<td>IF</td>
</tr>
<tr>
<td>PLAY</td>
</tr>
</tbody>
</table>

F1=Help  F2=Comment  F4=File  F5=Delete  F6=Insert  F12=Cancel

34. Press <F4> to save the 'choices' voice logic module and return to the Voice Logic Module Editor window. (See picture and description on page 27.)
Select **Add or Change** from the **Edit** pull-down in the **Voice Logic Module Editor** window to display the **Add or Change Voice Logic Module** window.

35. The next voice logic module in the TIME application is the ‘say_time’ module. Overtype the input field with the module name and accept the defaults for the Repeat and Time-out values. Enter the following comment text:

    Say the time

This module needs to play two segments:

a. The ‘time_is’ voice segment

b. The system variable ‘system_time’

Enter the two lines required to perform these functions as follows.

Play the ‘time_is’ voice segment:

Function      PLAY
Type          SEG
Operand One   ‘time_is’
Cond          (leave blank)
Operand Two   (leave blank)

Play the ‘system_time’ system variable:

Function      PLAY
Type          TIM
Operand One   system_time
Cond          (leave blank)
Operand Two   (leave blank)

36. Press <Esc> or <F12> to close the **Insert Line** window.

The content of the ‘time_is’ voice logic module is displayed in the **Add or Change Voice Logic Module** window.

<table>
<thead>
<tr>
<th>Add or Change Voice Logic Module - say_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>PLAY</td>
</tr>
<tr>
<td>PLAY</td>
</tr>
</tbody>
</table>

37. Press <F4> to save the ‘time_is’ voice logic module and return to the **Voice Logic Module Editor** window. (See picture and description on page 27.)
Select **Add or Change** from the Edit pull-down in the **Voice Logic Module Editor** window to display the **Add or Change Voice Logic Module** window.

38. Repeat the voice logic module creation process for the 'say_date' module using the following values.

Play the 'date_is' voice segment:

- **Function**: PLAY
- **Type**: SEG
- **Operand One**: 'date_is'
- **Cond**: (leave blank)
- **Operand Two**: (leave blank)

The 'system_date' system variable:

- **Function**: PLAY
- **Type**: DAT
- **Operand One**: system_date
- **Cond**: (leave blank)
- **Operand Two**: (leave blank)

39. Repeat the voice logic module creation process for the 'both' voice logic module, entering the following four lines.

Play the 'time_is' voice segment:

- **Function**: PLAY
- **Type**: SEG
- **Operand One**: 'time_is'
- **Cond**: (leave blank)
- **Operand Two**: (leave blank)

Play the 'system_time' system variable:

- **Function**: PLAY
- **Type**: TIM
- **Operand One**: system_time
- **Cond**: (leave blank)
- **Operand Two**: (leave blank)

Play the 'date_is' voice segment:

- **Function**: PLAY
- **Type**: SEG
- **Operand One**: 'date_is'
- **Cond**: (leave blank)
- **Operand Two**: (leave blank)

The 'system_date' system variable:

- **Function**: PLAY
- **Type**: DAT
- **Operand One**: system_date
- **Cond**: (leave blank)
- **Operand Two**: (leave blank)

40. Repeat the voice logic module creation process for the 'goodbye' module using the following values.

Play the 'goodbye' voice segment:

- **Function**: PLAY
41. Repeat the voice logic module creation process for the 'invalid' module using the following values.

Play the 'the_key' voice segment:

<table>
<thead>
<tr>
<th>Function</th>
<th>PLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SEG</td>
</tr>
<tr>
<td>Operand One</td>
<td>'the_key'</td>
</tr>
<tr>
<td>Cond</td>
<td>(leave blank)</td>
</tr>
<tr>
<td>Operand Two</td>
<td>(leave blank)</td>
</tr>
</tbody>
</table>

Play the 'last_dtmf_tone' system variable:

<table>
<thead>
<tr>
<th>Function</th>
<th>PLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SEG</td>
</tr>
<tr>
<td>Operand One</td>
<td>last_DTMF_tone</td>
</tr>
<tr>
<td>Cond</td>
<td>(leave blank)</td>
</tr>
<tr>
<td>Operand Two</td>
<td>(leave blank)</td>
</tr>
</tbody>
</table>

Play the 'invalid' voice segment:

<table>
<thead>
<tr>
<th>Function</th>
<th>PLAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>SEG</td>
</tr>
<tr>
<td>Operand One</td>
<td>'invalid'</td>
</tr>
<tr>
<td>Cond</td>
<td>(leave blank)</td>
</tr>
<tr>
<td>Operand Two</td>
<td>(leave blank)</td>
</tr>
</tbody>
</table>

42. 'Invalid' is the last voice logic module required for the TIME application. When you have saved the 'invalid' voice logic module, press <Esc> or <F12> until you return to the Voice Application Developer window.

Creating and Recording Voice Segments

The TIME application requires a number of voice segments to be played by the voice logic modules. Some of the segments are provided with DirectTalk/2, the others have to be recorded. You can also re-record the system supplied segments so that all the messages are in the same voice.

To create the voice segments for the TIME application:

1. Select Voice segment editor from the Editors pull-down in the Voice Application Developer window to display the Voice Segment Editor window.
2. Select **Add or Change** from the **Edit** pull-down in the **Voice Segment Editor** window to display the **Add or Change Voice Segment** window.

3. In the **Voice segment** field, type the name of the first voice segment you want to create. For the TIME application, type **hello**.

4. Press **<Enter>** to display the **Add or Change Voice Segment** window.

5. Type the message you want to record. For the hello segment enter:
   
   **Hello, welcome to the sample application.**

6. Press **<F4>** to file the text.

7. Press **<F10>** when you are ready to record the voice segment.

   You use the telephone to record voice segments and DirectTalk/2 displays the **Make Phone Connection** window to prompt you to make a telephone connection.
8. Dial the telephone number shown in the window.

   **Note:** Depending on how your system has been configured, this window may indicate a Line number rather than a specific telephone number.

   When the telephone connection has been made, the **Record Voice Segment** window is displayed.

   **Make Phone Connection**

   Make phone connection by dialing 555-1212.
   To cancel, press F12.
   Canceling may take a few seconds.
   F12=Cancel

9. To begin recording the voice segment, press <#> on the telephone key pad. DirectTalk/2 plays a tone to signal you to begin speaking.

10. Say:

    Hello, welcome to the sample application.

11. When you finish speaking, press <#> on the telephone key pad.

    Your new voice segment is saved and the **Record Voice Segment** window is closed.

12. Press <F11> to listen to the voice segment you have just recorded.

13. Press <Esc> or <F12> to close each of the windows until you return to the **Add or Change Voice Segment** window. (See picture and description on page 34.)

    **Note:** To avoid making repeated telephone connections, **do not hang up the telephone** until you have recorded and listened to all your voice segments.

14. Repeat the recording process for each of the user voice segments in the TIME application:
When you have recorded all your voice segments, Press <Esc> or <F12> to return to the Voice Application Developer window.

Table 3. Voice segments to be recorded

<table>
<thead>
<tr>
<th>Segment name</th>
<th>Text to be recorded</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu</td>
<td>Press one for time, two for date, three for both, or zero or star to exit.</td>
</tr>
<tr>
<td>warning</td>
<td>If you do not make a valid selection in...</td>
</tr>
<tr>
<td>seconds</td>
<td>...seconds you will be disconnected</td>
</tr>
<tr>
<td>goodbye</td>
<td>Good bye. Thank you for using TIME.</td>
</tr>
<tr>
<td>the_key</td>
<td>The key you pressed...</td>
</tr>
<tr>
<td>invalid</td>
<td>...is not a valid response.</td>
</tr>
<tr>
<td>time_is</td>
<td>The time is...</td>
</tr>
<tr>
<td>date_is</td>
<td>Today is...</td>
</tr>
</tbody>
</table>

15. When you have recorded all your voice segments, Press <Esc> or <F12> to return to the Voice Application Developer window.

Checking and Testing the Voice Application

When you have created your voice program, and all your voice logic modules and voice segments, you should check the completeness of your voice application and test its function.

Checking the Completeness of the Voice Application

To test the completeness of the voice application:

1. Select **Check** from the **Utilities** pulldown in the Voice Application Developer window.

   DirectTalk/2 checks the following:
   
   - All steps are completely defined
   - All steps referred to in the voice program are defined
   - All steps defined in the voice program are referred to
   - All voice logic modules used in the voice program exist
   - All voice segments used in the voice logic modules exist
   - All text segments used in the voice logic modules exist

   If any errors are found, a message is displayed. You can view or print the error list as follows:
   
   - Press <F8> to look at a list of any errors.
   - Press <F7> to print the list of errors.

2. To return to the Voice Application Developer window, press <Esc> or <F12>.
3. If you have errors:

- Use the voice program editor to correct steps. See “Creating the Voice Program” on page 18.
- Use the voice logic module editor to create any voice logic modules you have not defined. See “Creating the Voice Logic Modules” on page 26.
- Use the voice segment editor to create any voice segments you have not recorded. See “Creating and Recording Voice Segments” on page 33.

**Testing the Voice Application**

You should also test the voice application, as follows:

1. Select **Debug** in the **Utilities** pull-down in the **Voice Application Developer** window to display the **Debug Voice Application** window.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Comments</th>
<th>Break</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Wait_for_Call</td>
<td>Begin call processing loop</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'1'</td>
<td>Number of rings</td>
</tr>
<tr>
<td>2</td>
<td>'3'</td>
<td>Wait time in minutes</td>
</tr>
</tbody>
</table>

2. Use either of the following methods to test the voice application:

a. Press **<F7>** to run the complete voice application.

   DirectTalk/2 displays the **Make Phone Connection** window.
Dial the telephone number shown on the window. DirectTalk/2 runs the entire voice program.

**Note:** Depending on how your system has been configured, this window may indicate a Line number rather than a specific telephone number.

b. Run the voice program step by step as follows:

1) Press `<Enter>` to run the first step in the voice program. DirectTalk/2 runs the step and displays the result.

   **Note:** The Make Phone Connection window is displayed when the Wait_for_Call action is run.

2) Press `<Enter>` to display the step highlighted in the result.

3) Make any changes you want to the step.

4) Press `<Enter>` to run the step.

5) Repeat this process for the remaining steps in the voice program.

**Note:** You may find it useful to have a printed listing of your application while you are trying to debug it. See “Printing the Voice Application” on page 56 for details of how to print out your application.

3. When you have successfully checked your application, press `<Esc>`, or `<F12>` to return to the Voice Application Developer window. If you made any changes to the voice program, DirectTalk/2 prompts you to save them.

   - Press `<F4>` to save the changes.
   - Press `<Enter>` to discard the changes.

---

**Preparing to Run the Application**

During development and debug of your application, it has been running in an environment which is defined by the default Application Developer Setup file. To successfully transfer your application to production mode, you need to:

1. Create a control file which defines the environment in which the application is to be run.

2. Configure your production system using the DirectTalk/2 Setup program.

This section tells you how to create the control file, but you should refer to the IBM CallPath DirectTalk/2 Administrator’s Guide for how to configure your system and manage your application in production mode.
Creating the Application Control File

This file sometimes referred to simply as the Control File, contains global variables which define, among other things, the servers and databases which your application will use and the name of the application that you want to run.

To create an application control file:

1. Select Application control files in the Options pull-down in the Voice Application Developer window to display the Application Control Files window.

2. Enter the name of the application control file you want to create. Your system administrator will normally specify naming conventions for the file. For the TIME application, enter TIMECF00.CTL.

3. Do not enter anything in the Optional template for control file field.

4. Press <Enter> to display the Application Control Files window.

5. The values used to create this file are the default Application Developer setup values. If you plan to run the application initially on your development machine, you do not need to make any changes to this file.
6. If the application is to be run on a different machine, you may need to change some of the variable values. Your System Administrator should provide these values.

   To change a value:
   a. Move the cursor to the variable in the Application Control Files window
   b. Press <Enter> to display the Change Data window.
   c. Type the new value in the Data field.
   d. Press <Enter> to close the Change Data window.

7. Press <F4> to save the changes and create the Application Control File for your application.

Notes:

1. The default application control file is set up to start an application on the next available telephone line. If you want to start an application on a specific line, change the voice_line parameter by repeating the previous step for voice_line instead of initial_appl. (You will need to page through the list of parameters to find voice_line.) Set voice_line to the line number on which you want the application to run. (A value of 0 for the parameter means that voice applications are started on the next free phone line.)

2. If you are moving your application to an established environment, the required parameter values may be contained in a 'Template' file. If you have such a file, you can enter its name in the Optional templates for control file field in the Application Control Files window (see picture and description on page 39). The file you create will then contain the values defined in the template, rather than the Application Developer setup values.
Part 2. More About Voice Application Development

This section describes functions and techniques that may be useful when you are creating your own applications.

Use the online help supplied with the Voice Application Developer if you need more assistance with finding and using the described functions.

Chapter 5. Variables for Voice Applications
- Creating and Changing Local Variables
  - Creating or Changing Application Variables
- Variables Created by Actions
- Creating Variables with the _Variable Actions
- Creating and Changing Global Variables
  - Creating and Changing the Application Control File
  - Using the _Variable Actions
  - Creating and Changing the Application Developer Control File
- System Variables
- User Variables

Chapter 6. More About Voice Applications
- Opening Another Voice Application
- Printing the Voice Application
- Using Different Languages
  - Using Multiple Languages in an Application

Chapter 7. More About Voice Programs
- Editing Voice Programs
  - Inserting Steps
  - Changing Steps
  - Renumbering Steps
  - Copying Steps
  - Moving Steps
  - Deleting Steps
- Setting Break Points
- Handling System Error Return Codes

Chapter 8. More About Voice Logic Modules
- Voice Logic Module Functions
  - Options for the PLAY Function
  - Options for the IF Function
- Editing Voice Logic Modules
  - Changing Module Parameters
  - Inserting Logic Statements
  - Changing Logic Statement Details
  - Deleting Logic Statements
- Adding Voice Logic Modules
- Deleting Voice Logic Modules
- Copying Voice Logic Modules
- Testing Voice Logic Modules
- Printing Voice Logic Modules
- Using Common Voice Logic Modules
### Voice Logic Module Currency Variable
- 69

### Minor Currency Processing
- 69

### Implied Currency Decimal
- 69

#### Chapter 9. More About Voice Segments
- 71
  - Adding and Changing Voice Segments
  - 71
  - Deleting Voice Segments
  - 71
  - Playing Voice Segments
  - 71
  - Copying Voice Segments
  - 71
  - Printing Voice Segments
  - 72
  - Using System-Supplied Voice Segments
  - 72
  - Using Common Voice Segments
  - 72
  - Editing Voice Segments
  - 73
  - Recording or Editing Multiple Voice Segments
  - 75
  - Recording Voice Segments Using a Tape Player or Microphone
  - 76

#### Chapter 10. Call Transfer and Referral
- 79
  - Call Extension on Analog and T1 Lines
  - 79
  - Call Extension on E1 Lines
  - 79
    - Referring a Call
    - 80
    - Transferring a Call
    - 81
    - Call Transfer Application Example
    - 81
    - E1 Extend Call System Variables
    - 81

#### Chapter 11. Password Security
- 85
  - Password Prompt
  - 87

#### Chapter 12. DirectTalk/2 REXX Environment
- 89
  - REXX Sample Voice Applications
  - 89
  - Command Line Processing
  - 90
    - The TREXX function
    - 91
    - DirectTalk/2 REXX Environment Debug Facilities
    - 92
  - Variable Processing
  - 92
    - Voice Programs
    - 92
    - User Actions
    - 93
    - REXX Variable Naming Standards
    - 94
    - DirectTalk/2 Global Variable Processing
    - 94
    - Global Variable Support for T-REXX Internal Procedures
    - 95
    - The TREXX-ENV Environment Variable
    - 95
  - REXX Subroutines and Functions
  - 95
  - Protecting REXX Voice Programs
  - 96
  - Handling System Error Return Codes
  - 97
Chapter 5. Variables for Voice Applications

Note: The information in this chapter relates particularly to voice programs that are written using the VAD. If you are writing REXX programs see “Variable Processing” on page 92.

DirectTalk/2 uses variables to store information that is required or created by voice applications. There are two types of variable data that can be used:

- Local variables
- Global variables

Some variables of each type are defined and used by DirectTalk/2 as system variables (see “System Variables” on page 47), and you can also create user variables (see “User Variables” on page 52) of both types to meet your own specific requirements.

The maximum amount of data that can be stored in a variable is 4Kb.

Local Variables

Local variables are available only to the application in which they are created. If your application links to another application, local variables created or set in the first application will not be seen or available to the linked application. If you need to make data available to another application, then you must create the variable as a global variable.

Sources of local variables are:

- The Application Setup file (<application_name>.SU)
- System local variables
- The Create_Variable action

Note: Local variables take precedence over any global variables defined with the same name.

Global Variables

Global variables are available to all applications running in an application session. If your application links to another application, global variables created or set in the first application are available and can be updated by any linked application. Global variables cannot be shared across the same application or different applications running in separate application sessions. If you need to share data across sessions, you must use DirectTalk/2 databases and actions.

Sources of global variables are:

- The Application Control file
- System global variables
- The Create_Variable action

Variable names must conform to the following rules:

- Be from 1 to 16 characters in length
- Contain any alphanumeric characters, including an underscore (_), dash (—), dollar sign ($), and number sign (#)
- Not the same as any of the System variable names (see “System Variables” on page 47)
The names of the variables are case-sensitive.

Creating and Changing Local Variables

Local variables can be created and changed using the following methods:

- Using the Application Setup file
- By DirectTalk/2 actions in the voice application
- Using Create_variable, Set_Variable and Delete_Variable actions

Creating or Changing Application Variables

Application variables are the local variables created for use by a specific voice application, and stored in the Application Setup file (application_name.SU). This file is automatically loaded whenever the application is run.

The Application Setup file can be created with an editor, or using the VAD. The VAD method is described below.

To Create an Application Variable:

1. Select Application variables from the Options pull-down in the Voice Application Developer window to display the Application Variables window.

   This window displays a list of the current application variables.

2. Press <F6> to display the Add window.
3. Type the name of the variable you want to add (see “rules for variable names” on page 43).

4. Press <Enter> to display the Change Data window.

5. Type the value for the new variable in the Data field. The maximum size of the new variable is 4,096 characters.

6. Press <Enter> to close the Change Data window.

**To Change an Application Variable**

1. Select the variable in the Application Variables window.
2. Press <Enter> to display the Change Data window.
3. Overtype the new value in the Data field.
4. Press <Enter> to close the Change Data window.

**To Delete an Application Variable**

1. Select the variable in the Application Variables window.
2. Press <F5> and confirm any messages you may see.

When you have made all your additions, changes, and deletions, you can:

- Press <F4> to save any additions or changes and return to the Voice Application Developer window.
- Press <Esc> or <F12> to return to the Voice Application Developer window without saving any additions or changes.

**Variables Created by Actions**

If your application uses actions that result in a system variable being created, DirectTalk/2 always creates the variable as local.
Creating Variables with the _Variable Actions
The Create_Variable action (see page 191) allows you to create local and global variables. The action requires a variable name to be supplied and, optionally a variable Type. You can create a local variable by specifying a Type of 'local', or by omitting the type information as it will default to 'local'.

Variables can be changed using the Set_Variable action (see page 272), and they can be deleted using the Delete_Variable action (see page 193).

Creating and Changing Global Variables
Global variables can be created by the following methods:

- Using the Developer Control file (development)
- Using the Application Control file (production)
- Using the Create_Variable, Set_Variable and Delete_Variable actions

Creating and Changing the Application Control File
DirectTalk/2 uses the Application control file to define the environment for your applications during production. The Application Control file should have an extension of .CTL and it can be created with an editor, or by using the VAD. The VAD method is described in “Creating the Application Control File” on page 39.

Using the _Variable Actions
The use of the Create_Variable, Set_Variable, and Delete_Variable actions is described in “Creating Variables with the _Variable Actions.” To create a global variable by this method, you must specify a Type of 'global'.

Creating and Changing the Application Developer Control File
DirectTalk/2 uses the Application Developer control file (VSVAD.CFG) to define the environment for your applications during development and testing. DirectTalk/2 also uses these variables as the default template for the Application control file after you complete the development of the voice application. The default values are specified during installation.

1. Select Application Developer setup from the Options pull-down in the Voice Application Developer window to display the Voice Application Developer Setup window.
2. To add, change, or delete variables, use the procedures described in “Creating the Application Control File” on page 39.

System Variables

DirectTalk/2 contains a number of system variables which it creates and maintains for you to use in your voice application. System variables can be local or global. These variables hold system information, such as the current date and time, the application name, or the last DTMF tone received. Voice applications can read a value from these variables, but should not modify them. Table 4 contains a list, and brief descriptions of these system variables. See Part 4, “Actions” on page 151, for information on where and how these variables are created and used.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>3270_server</td>
<td>The node name of the 3270 terminal emulator.</td>
</tr>
<tr>
<td>5250_server</td>
<td>The node name of the 5250 terminal emulator.</td>
</tr>
<tr>
<td>ADSI_cmn_appl</td>
<td>The name of an application which contains ADSI script files and their associated parameters.</td>
</tr>
<tr>
<td>ADSI_cmn_srvr</td>
<td>The server name of the DirectTalk/2 node that contains the ADSI scripts and their associated parameters.</td>
</tr>
<tr>
<td>ADSI_script_srvr</td>
<td>The node name of the ADSI server.</td>
</tr>
<tr>
<td>ANI_DNIS_data</td>
<td>Contains any Automatic Number Identification (ANI) and/or Dialed Number Identification Service (DNIS) data received.</td>
</tr>
<tr>
<td>application_name</td>
<td>Contains the name of the DirectTalk/2 application that is running.</td>
</tr>
<tr>
<td>ASCII_server</td>
<td>The node name of the ASCII terminal emulator.</td>
</tr>
<tr>
<td>case_sensitive</td>
<td>Used with Search_String to specify whether the search should be case sensitive.</td>
</tr>
<tr>
<td>close_database</td>
<td>If this variable is set to Y databases close after each access. Default is N.</td>
</tr>
<tr>
<td>CM32_server</td>
<td>The node name of the Communications Manager terminal emulator.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Variable Contents</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CM_key_reset</td>
<td>Set to 'NO' to disable automatic reset before Send_Keys_to_Scr.</td>
</tr>
<tr>
<td>cs_applname</td>
<td>The application name in the statistics log.</td>
</tr>
<tr>
<td>cs_calldate</td>
<td>The date an application answered an inbound call or dialed an outbound call. (Statistics log.)</td>
</tr>
<tr>
<td>cs_calltime</td>
<td>The time at which an application answered an inbound call or dialed an outbound call. (Statistics log.)</td>
</tr>
<tr>
<td>cs_calltype</td>
<td>The type of call the action processed. I=inbound, O=outbound. (Statistics log.)</td>
</tr>
<tr>
<td>cs_duration</td>
<td>The length of the call in seconds. (Statistics log.)</td>
</tr>
<tr>
<td>cs_hangstep</td>
<td>The step where the application detected that the caller hung up. (Statistics log.)</td>
</tr>
<tr>
<td>cs_lineno</td>
<td>The physical port number on which the application placed or received the call. (Statistics log.)</td>
</tr>
<tr>
<td>cs_number</td>
<td>The string the action dialed for an outbound call. Set to null for an inbound call. (Statistics log.)</td>
</tr>
<tr>
<td>cs_procedure</td>
<td>A string the application uses to identify the processing being performed when the caller hung up. (Statistics log.)</td>
</tr>
<tr>
<td>cs_record</td>
<td>Set to Y to log call detail statistics when the Log_Statistics action is called. See “Log_Statistics” on page 225 for a complete list of call detail variables.</td>
</tr>
<tr>
<td>cs_termtype</td>
<td>The termination type for a call. A=application hung up. C=caller hung up. T=application transferred call.</td>
</tr>
<tr>
<td>database_server</td>
<td>The node name of the Database Server node containing any general use local DirectTalk/2 databases.</td>
</tr>
<tr>
<td>db_record_lock</td>
<td>Tells the database Get_ actions to lock the record for update.</td>
</tr>
<tr>
<td>db_record_wait</td>
<td>Tells the database actions to wait until the target record is unlocked. Otherwise the action returns immediately without effect.</td>
</tr>
<tr>
<td>directory_server</td>
<td>The node name of Voice Messaging directory server.</td>
</tr>
<tr>
<td>dtlk_edit_x</td>
<td>A set of variables available to applications, where x is the language letter as defined in the TAMDLANG.NME file. See “Voice Logic Module Currency Variable” on page 69 for a full description.</td>
</tr>
<tr>
<td>dtmf_escape_char</td>
<td>Contains the key that caused an escape from the last Get_Tone_String action.</td>
</tr>
<tr>
<td>dtmf_term_char</td>
<td>Contains the key that terminated the last Get_Tone_String action.</td>
</tr>
<tr>
<td>entry_fail_flag</td>
<td>Set to 1 by the VR_Get_String action if it was unable to verify the caller’s response.</td>
</tr>
<tr>
<td>e_full_tran_pos</td>
<td>Indicates whether full call transfer is possible.</td>
</tr>
<tr>
<td>e_num_ext_lines</td>
<td>Number of extend call resources available.</td>
</tr>
<tr>
<td>e_lines_for_tran</td>
<td>Number of lines required for call transfer.</td>
</tr>
<tr>
<td>e_lines_for_ref</td>
<td>Number of lines required for call referral.</td>
</tr>
</tbody>
</table>
**Table 4 (Page 3 of 6). DirectTalk/2 System Variables**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_min_tran_scnn1</td>
<td>Minimum screen level for call transfer.</td>
</tr>
<tr>
<td>e_r_canhear_a1</td>
<td>Indicates whether it is possible to hear agent in extended calls.</td>
</tr>
<tr>
<td>e_r_canhear_c1</td>
<td>Indicates whether it is possible to hear caller in extended calls.</td>
</tr>
<tr>
<td>e_r_canplayto_a1</td>
<td>Indicates whether it is possible to talk to agent in extended calls.</td>
</tr>
<tr>
<td>e_r_canplayto_c1</td>
<td>Indicates whether it is possible to talk to caller in extended calls.</td>
</tr>
<tr>
<td>e_w_canhear_a1</td>
<td>Indicates whether it is possible to hear agent after Call_Extend_Init.</td>
</tr>
<tr>
<td>e_w_canhear_c1</td>
<td>Indicates whether it is possible to hear caller after Call_Extend_Init.</td>
</tr>
<tr>
<td>e_w_canplayto_a1</td>
<td>Indicates whether it is possible to talk to agent after Call_Extend_Init.</td>
</tr>
<tr>
<td>e_w_canplayto_c1</td>
<td>Indicates whether it is possible to talk to caller after Call_Extend_Init.</td>
</tr>
<tr>
<td>evr_beep_flag</td>
<td>Indicates whether beep will sound between character input for voice recognition.</td>
</tr>
<tr>
<td>evr_no_beep</td>
<td>Indicates whether beep should be sounded between character input for voice recognition.</td>
</tr>
<tr>
<td>ext_canhear_a2</td>
<td>Indicates whether agent output can be heard in extended calls.</td>
</tr>
<tr>
<td>ext_canhear_c2</td>
<td>Indicates whether caller output can be heard in extended calls.</td>
</tr>
<tr>
<td>ext_canplayto_a2</td>
<td>Indicates whether agent can hear voice system in extended calls.</td>
</tr>
<tr>
<td>ext_canplayto_c2</td>
<td>Indicates whether caller can hear voice system in extended calls.</td>
</tr>
<tr>
<td>ext_connect_type2</td>
<td>Type of agent connection detected in extended calls.</td>
</tr>
<tr>
<td>ext_fulltran_pos2</td>
<td>Indicates whether network connection allows full call transfer.</td>
</tr>
<tr>
<td>ext_line2_trunk2</td>
<td>Channel number being used for second line in extended calls.</td>
</tr>
<tr>
<td>ext_line2_ts2</td>
<td>Time slot being used for second line in extended calls.</td>
</tr>
<tr>
<td>ext_monagent_hup2</td>
<td>Indicates whether agent hangup can be detected in extended calls.</td>
</tr>
<tr>
<td>ext_moncall_hup2</td>
<td>Indicates whether caller hangup can be detected in extended calls.</td>
</tr>
<tr>
<td>ext_num_of_lines2</td>
<td>Indicates number of lines being used in an extended call.</td>
</tr>
<tr>
<td>ext_referral_poso2</td>
<td>Indicates whether the current network connection will allow call referral.</td>
</tr>
<tr>
<td>ext_switched_at2</td>
<td>Indicates where switching occurs in extended calls.</td>
</tr>
<tr>
<td>gmsg_msg_db</td>
<td>The database name used by the Get_Message action to store voice messages.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Variable Contents</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>gmsg_msg_server</td>
<td>The node name of the Database server node containing the voice messaging databases</td>
</tr>
<tr>
<td></td>
<td>to be used by the Get_Message action.</td>
</tr>
<tr>
<td>gmsg_notebook</td>
<td>The name of a notebook from which to retrieve a message.</td>
</tr>
<tr>
<td>gmsg_phone</td>
<td>The caller’s telephone number for the Get_Messages action.</td>
</tr>
<tr>
<td>handle_acterr</td>
<td>The error handling step number. 0=terminate application.</td>
</tr>
<tr>
<td>initial_appl</td>
<td>The application to be started by a voice application session.</td>
</tr>
<tr>
<td>key_immediate</td>
<td>If set to Yes, the system will return immediately after keys have been sent to the</td>
</tr>
<tr>
<td></td>
<td>host, without waiting for the host to unlock the keyboard.</td>
</tr>
<tr>
<td>last_cont_resp1</td>
<td>Contains the characters received from the Get_Voice_Cont action as the first choice. The value that DirectTalk/2 stores in the variable depends upon the vocabulary you specify for the action.</td>
</tr>
<tr>
<td>last_cont_resp2</td>
<td>Contains the characters received from the Get_Voice_Cont action as the second choice.</td>
</tr>
<tr>
<td>last_dtmf_data</td>
<td>Contains the keys the last Get_Tone_String action received.</td>
</tr>
<tr>
<td></td>
<td>If a caller terminates the input with a #, DirectTalk/2 does not store the # at the end of the string.</td>
</tr>
<tr>
<td>last_dtmf_tone</td>
<td>Contains the key the last Get_a_Tone action received.</td>
</tr>
<tr>
<td>last_found_column</td>
<td>The column number of the screen position of data found on a host screen.</td>
</tr>
<tr>
<td>last_found_row</td>
<td>The row number of the screen position of data found on a host screen.</td>
</tr>
<tr>
<td>last_repeat</td>
<td>If the value is ‘1’, the last repeat of a voice logic module is in process. You can use it to conditionally play a last chance message by using IF statements in a voice logic module.</td>
</tr>
<tr>
<td>last_voice_resp</td>
<td>Contains the characters received from the Get_Voice_Resp, Get_Voice_Cont, VR_Get_String, or VR_Get_Yes_No action. The value DirectTalk/2 stores in the variable depends on the vocabulary you specify for the action.</td>
</tr>
<tr>
<td>linked_from_appl</td>
<td>Contains the name of the voice application that linked to this voice application. For the main application, this variable is null.</td>
</tr>
<tr>
<td>mailbox_server</td>
<td>The node name of the Voice Messaging server.</td>
</tr>
<tr>
<td>oia_data_ascii</td>
<td>A 130-byte string of ASCII characters representing Operator Information Area (OIA) data for ASCII host emulators.</td>
</tr>
<tr>
<td>oia_data</td>
<td>Contains Operator Information Area (OIA) binary data for 3250 and 5250 host emulators.</td>
</tr>
<tr>
<td>previous_rc</td>
<td>Contains the return code of the step that ran prior to the current step.</td>
</tr>
<tr>
<td>previous_step</td>
<td>Contains the step number that ran prior to the current step.</td>
</tr>
<tr>
<td>previous_step2</td>
<td>Contains the step number that ran prior to the prior step.</td>
</tr>
<tr>
<td>recording_time</td>
<td>Contains the duration, in seconds, of the last voice recording played by the Record_Voice action.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Variable Contents</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>resp_escape_char</td>
<td>The response that caused the Escape return code to a Get_Voice_Resp action.</td>
</tr>
<tr>
<td>resp_term_char</td>
<td>Contains the reason a Get_Voice_Resp, Get_Voice_Cont, VR_Get_String, or VR_Get_Yes_No action was terminated.</td>
</tr>
<tr>
<td>stat_server</td>
<td>The node name of the Statistics server.</td>
</tr>
<tr>
<td>state_server</td>
<td>The node name of the Database server node containing the application Voice Programs.</td>
</tr>
<tr>
<td>sys_logic_srvr</td>
<td>The node name of the Database server node containing the system voice logic modules (SYSTM application).</td>
</tr>
<tr>
<td>sys_seg_server</td>
<td>The node name of the Database server node containing the system voice segments (SYSTM application).</td>
</tr>
<tr>
<td>system_date</td>
<td>Contains the current date in the format YYYYMMDD.</td>
</tr>
<tr>
<td>system_dow</td>
<td>Contains a number indicating the day of the week for the current date. For example, a value of 1 indicates Sunday, and 2 indicates Monday.</td>
</tr>
<tr>
<td>system_time</td>
<td>Contains the current time in hours, minutes, and seconds in the 24-hour format, HHMMSS.</td>
</tr>
<tr>
<td>telephone_number</td>
<td>Contains the telephone number associated with the current voice line. This value is set during system configuration.</td>
</tr>
<tr>
<td>term_application</td>
<td>This is set to 'yes' if the application is stopped by the Node Manager during execution.</td>
</tr>
<tr>
<td>text_segmt_srvr</td>
<td>The node name of the Database server node containing the text segment databases.</td>
</tr>
<tr>
<td>timeout_flag</td>
<td>If the value is 1, the caller did not respond in time for actions such as Get_a_Tone, Get_Tone_String or Get_Voice_Resp. You can use this flag to conditionally play a reinforcement or help message by using IF statements in a voice logic module.</td>
</tr>
<tr>
<td>timeout_value</td>
<td>Contains the time-out value, in seconds, specified on the current voice logic module. Use this value as the time allowed for a caller to respond to a Get_a_Tone action or a Get_Tone_String action. The value can be spoken to tell the caller how much time there is for the caller's response before the time you specify runs out. This variable is also used by the Wait_for_Hang action, to specify how long the application should continue to look for a hangup. The Set_Timeout action can be used to set this variable.</td>
</tr>
<tr>
<td>tmsg_g/n_db</td>
<td>The database name used by the Take_Message action to retrieve greetings and names.</td>
</tr>
<tr>
<td>tmsg_g/n_greet</td>
<td>The key of a personal greeting.</td>
</tr>
<tr>
<td>tmsg_g/n_name</td>
<td>The key for a recorded name.</td>
</tr>
<tr>
<td>tmsg_g/n_server</td>
<td>The node name of the Database server node containing the voice messaging greeting and name databases to be used by the Take_Message action.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Variable Contents</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tmsg_grtg_exit</td>
<td>The string of DTMF tones which, if received during a Take_a_Message greeting, result in the Greeting exit return code.</td>
</tr>
<tr>
<td>tmsg_msg_db</td>
<td>The database name used by the Take_Message action to retrieve voice messages.</td>
</tr>
<tr>
<td>tmsg_msg_server</td>
<td>The node name of the Database server node containing the voice messaging databases to be used by the Take_Message action.</td>
</tr>
<tr>
<td>tmsg_page_data</td>
<td>The digital information to send to a digital paging system.</td>
</tr>
<tr>
<td>tmsg_page_phone</td>
<td>The telephone number of the recipient’s paging system.</td>
</tr>
<tr>
<td>tmsg_page_type</td>
<td>The type of paging system the action uses when it stores a message.</td>
</tr>
<tr>
<td>tmsg_phone</td>
<td>The telephone number of the person receiving the message.</td>
</tr>
<tr>
<td>tmsg_rtime</td>
<td>The maximum recording time for a message.</td>
</tr>
<tr>
<td>voice_language</td>
<td>Contains the Language Name of the active language in the voice application; for example, US_English for U.S. English. DirectTalk2 uses this variable for database access. You should alter this variable only if the voice logic modules and voice segments exist for the new language. For further details, see IBM CallPath DirectTalk/2 National Language Information.</td>
</tr>
<tr>
<td>voice_line</td>
<td>The physical port number used by the application session. A value of 0 indicates the next available port. A value of −1 indicates that the application session does not require a physical port.</td>
</tr>
<tr>
<td>voice_logic_srvr</td>
<td>The node name of the Database server node containing the application voice logic modules.</td>
</tr>
<tr>
<td>voice_segmt_srvr</td>
<td>The node name of the Database server node containing the application voice segments.</td>
</tr>
<tr>
<td>voice_server</td>
<td>The node name of the Telephony Server, which contains the voice hardware and voice telephony functions.</td>
</tr>
</tbody>
</table>

**Notes:**
1. These variables are set when a Call_Extend_Cfg action is called.
2. These variables are set after any Extend Call action.

**User Variables**

User, or application variables, are defined by the user to meet specific voice application requirements and can be either local or global. For example, the variable could contain the user identifier the application should use to gain access to a host application. You can create and use as many user variables as necessary in a voice application and the applications are responsible for setting the values.

If you are using mailboxes, DirectTalk/2 also contains a number of user variables for which you supply part of the name. These variables are created and maintained by the system once you have supplied the necessary name prefix.
Table 5 on page 53 contains a list, and brief descriptions of the system maintained user variables. See Part 4, “Actions” on page 151, for information on where and how these variables can be used.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;prefix&gt;_dept</td>
<td>The department of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_fnm</td>
<td>The first name of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_node</td>
<td>The node address of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_grtgkey</td>
<td>Generated key for a personal greeting voice segment in the recorded names and greetings database.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_idno</td>
<td>A unique identification number for an item.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_lnm</td>
<td>The last name of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_loc</td>
<td>The location of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_mailbox</td>
<td>The assigned mailbox number of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_namekey</td>
<td>Generated key for the recorded name segment in the recorded names and greetings database.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_pagedata</td>
<td>The digital paging data.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_pagephon</td>
<td>The paging telephone number.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_pagetype</td>
<td>The paging type (V, D, or null).</td>
</tr>
<tr>
<td>&lt;prefix&gt;_phone</td>
<td>The telephone number of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_pswd</td>
<td>The password of the mailbox owner.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_rtime</td>
<td>The maximum recording time for a message.</td>
</tr>
<tr>
<td>&lt;prefix&gt;_userid</td>
<td>The userid of the mailbox owner.</td>
</tr>
</tbody>
</table>
Chapter 6. More About Voice Applications

In Chapter 4 we created a simple voice application. This chapter describes other functions that may be of use when you create your own applications.

Opening Another Voice Application

You may want to change from the current application to another application when, for example, you want to copy voice logic modules or voice segments from one application to another, or if you are using common voice logic modules or voice segments.

To change the current application:

1. Select Open in the File pull-down to display the Open Application window.

2. Select the application you want to open.

The following windows will now relate to the application you have selected:

- Voice Application Developer window
- Voice Logic Module Editor window
- Voice Segment Editor window
- Text Segment Editor window
- ADSI script database
Printing the Voice Application

You can use the Print function to generate a listing of your application that can be printed or saved in a file. The listing contains:

- All the steps and step details in the voice application
- The names, logic statements, and text descriptions of all the voice logic modules in the voice application
- The names and text of all the voice segments in the voice application
- Cross-references of steps and variables in the application

1. Select Print in the File pull-down in the Voice Application Developer window.

DirectTalk/2 displays the Print Application window.

2. Type 1 or 2 in the Output field depending on whether you want to print or save the listing.

   Note: The Voice Application Developer formats the output for the IBM Proprinter.

   If your printer is not Proprinter-compatible, it may be better to save the application listing to a file and then print the file using regular OS/2 commands.

3. If you choose to save the listing in a file, you can change the name of the list file by entering a name in the Store as text file field. DirectTalk/2 places the file in the OS/2 directory in which DirectTalk/2 is installed.

4. If you want to specify the name of a different printer, type the printer name in the Printer field.

5. If you do not want to print the entire voice application, specify the range of steps in the voice application you want to print in the Range field.

6. Press <Enter> to print or store the listing.
Using Different Languages

DirectTalk/2 can be configured to work with a number of different languages. When an application is started, DirectTalk/2 uses the `voice_language` variable to decide which voice logic module and voice logic databases to use. Table 6 shows a list of supported languages and the associated database file identifiers. These identifiers are the last letters of the associated voice logic module and voice segment database files. For example, for the menu application:

MENU.PRE and MENU.SGE are the US English databases
and
MENU.PRC and MENU.SGC are the Canadian French databases

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Traditional_Chinese</td>
</tr>
<tr>
<td>b</td>
<td>Brazilian_Portuguese</td>
</tr>
<tr>
<td>c</td>
<td>Canadian_French</td>
</tr>
<tr>
<td>d</td>
<td>Dutch</td>
</tr>
<tr>
<td>e</td>
<td>US_English</td>
</tr>
<tr>
<td>f</td>
<td>French</td>
</tr>
<tr>
<td>g</td>
<td>German</td>
</tr>
<tr>
<td>h</td>
<td>Danish</td>
</tr>
<tr>
<td>i</td>
<td>Italian</td>
</tr>
<tr>
<td>j</td>
<td>Japanese</td>
</tr>
<tr>
<td>k</td>
<td>Norwegian</td>
</tr>
<tr>
<td>l</td>
<td>Mexican_Spanish</td>
</tr>
<tr>
<td>m</td>
<td>Swiss_French</td>
</tr>
<tr>
<td>n</td>
<td>Swiss_German</td>
</tr>
<tr>
<td>o</td>
<td>Swiss_Italian</td>
</tr>
<tr>
<td>p</td>
<td>Portuguese</td>
</tr>
<tr>
<td>q</td>
<td>Afrikaans</td>
</tr>
<tr>
<td>r</td>
<td>Cantonese</td>
</tr>
<tr>
<td>s</td>
<td>Castilian_Spanish</td>
</tr>
<tr>
<td>t</td>
<td>Catalan</td>
</tr>
<tr>
<td>u</td>
<td>UK_English</td>
</tr>
<tr>
<td>v</td>
<td>Arabic</td>
</tr>
<tr>
<td>w</td>
<td>Swedish</td>
</tr>
<tr>
<td>x</td>
<td>Flemish</td>
</tr>
<tr>
<td>y</td>
<td>Belgian_French</td>
</tr>
<tr>
<td>z</td>
<td>Korean</td>
</tr>
<tr>
<td>0</td>
<td>Luxembourgish</td>
</tr>
<tr>
<td>1</td>
<td>Customer_Use</td>
</tr>
<tr>
<td>2</td>
<td>Customer_Use</td>
</tr>
<tr>
<td>3</td>
<td>Customer_Use</td>
</tr>
<tr>
<td>4</td>
<td>Customer_Use</td>
</tr>
<tr>
<td>5</td>
<td>Customer_Use</td>
</tr>
<tr>
<td>6</td>
<td>Turkish</td>
</tr>
<tr>
<td>7</td>
<td>Malaysian</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
</tr>
<tr>
<td>9</td>
<td>Finnish</td>
</tr>
</tbody>
</table>
To use a language other than US English you need to do the following:

1. Set the *voice_language* global variable to the specific language name. When you install DirectTalk/2 this variable is set to 'null' and the system defaults to US English. The way you set the variable depends on whether you are working in a development or production environment:

   **During Application Development**
   
   Set the variable in the Application Developer Control File as described in “Creating and Changing the Application Developer Control File” on page 46.

   **In the Production Environment**
   
   The variable needs to be set in the Application Control File as described in “Creating the Application Control File” on page 39.

2. Install the language specific voice segments, voice logic modules, and language DLLS, by selecting the language during installation (see the *IBM CallPath DirectTalk/2 Installation Guide*), or create new modules and segments.

   If you are creating new modules or segments, open the applications after setting the *voice_language* variable so that the new modules and segments are stored in the appropriate database.

   **Note:** You cannot copy voice logic modules and segments between language databases, but you can create a new language database by making a copy of existing language files and giving them extensions with the new country identifier. The voice module and segment names will then be available in the new language, but the voice segments will of course need to be rerecorded.

**Using Multiple Languages in an Application**

It is possible to write applications which use more than one language. To do this you must install or create the voice logic module and voice segment databases for the languages you want to use in the application. As described above, the databases that your application uses are determined by the setting of the *voice_language* global variable. You can change this setting from within your application, using the *Set_Variable* action (see “Set_Variable” on page 272) to enable the appropriate language at any particular time.
Chapter 7. More About Voice Programs

In Chapter 4, “Creating a simple voice application” on page 13 we created a simple voice program. This chapter describes other functions that may be of use when you create your own programs.

Editing Voice Programs

You can edit the voice program to:

- Insert steps
- Change step details
- Renumber steps
- Copy steps
- Move steps
- Delete steps
- Set or remove break points

The general procedure for editing a voice program is:

- Select Voice program editor in the Editors pull-down in the Voice Application Developer window to display the Voice Program Editor window with the voice program for the current application.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Comment</th>
<th>Break</th>
<th>Incomplete</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Wait_for_Call</td>
<td>Begin call loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Play_Module</td>
<td>Say hello</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Play_Module</td>
<td>Ask caller for choice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Get_a_Tone</td>
<td>Get the caller’s choice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Play_Module</td>
<td>Repeat invalid response</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Play_Module</td>
<td>Tell the time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Play_Module</td>
<td>Play the date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Play_Module</td>
<td>Play both time and date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Play_Module</td>
<td>Say goodbye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Hang_up_Phone</td>
<td>End of call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Return_from_App</td>
<td>Application shutdown</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

F1=Help  F2=Save  F4=File  F5=Delete  F6=Insert  F8=Break  F10=Menus  F12=Cancel
• Make the required changes to the voice program.
• Press F2 to save or F4 to file the changes you have made to the voice program.

  **Note:** You can also save or file changes to a voice program by selecting **Save** or **File** from the Program pull-down menu of the voice program editor window.
• Press Esc or F12 to cancel any changes you have made to the voice program. DirectTalk/2 prompts you before discarding the changes.
  – Press F4 to file the changes.
  – Press Enter to discard the changes.

If you want more information on any of the editing procedures refer to the online help supplied with the Voice Application Developer.

**Inserting Steps**

To insert a step:

1. Select the step that will precede the step you want to insert. To insert a step before the first step in the voice program, select the column title line.
2. Press `<F6>` or select **Insert** from the **Edit** pull-down menu.
3. Select the action you require and complete the step as described in “Creating the Voice Program” on page 18.

**Changing Steps**

To change a step:

1. Select the step you want to change.
2. Point to the step and press the right mouse button, or press `<Enter>`, or select **Open** from the **Edit** pull-down menu to display the Step Details window.
3. Complete the Step Details window and press `<Enter>`.

**Renumbering Steps**

If you make a large number of changes to a program, you may wish to renumber steps. You can renumber steps by doing one of the following:

• Select the steps to be renumbered and then Select **Renumber** from the **Edit** pull-down menu.
• Select **Renumber all** from the **Program** pull-down menu (to renumber all the steps in your program).

**Copying Steps**

To copy steps:

1. Select the steps that you wish to copy.
2. Select **Copy** from the **Edit** pull-down menu to display the **Copy Steps** window.
3. Refer to the online help if you need assistance to fill in the fields.
4. When the **Copy Steps** window is complete, press `<Enter>`
Moving Steps

To move steps:

1. In the Voice Program Editor window, select the steps that you wish to move.
2. Select Move from the Edit pull-down menu to display the Move Steps window.
3. Refer to the online help if you need assistance to fill in the fields.
4. When the Move Steps window is complete, press <Enter>.

Deleting Steps

To delete a step:

1. Select the step(s) you want to delete.
2. Press <F5>, or select Delete from the Edit pull-down menu to display the Delete Step window.
3. Press <F5> in the Delete Step window to confirm that you want to delete the step(s).

Setting Break Points

In “Checking and Testing the Voice Application” on page 36 we described how to run your complete program, or to run it step by step. While you are debugging your voice program, you may wish to pause at specific points, rather than at every step. You can indicate where you would like your program to pause by setting a Break point at the required step.

You can set a break point on any number of steps in a voice program. Break points will have no effect when an application is put into production mode. While you are testing an application using the debug option, execution will always pause on steps which have a break point set.

To set or remove break points in the voice program:

1. Display the voice program with the Voice Program Editor. (See page 59.)
2. Select the step at which you want to set or remove the break point.
3. Press F8, or else select Break on/off from the Edit pull-down menu.
   - If no break point currently exists at that step, DirectTalk/2 sets the break point and places an asterisk in the Break field.
   - If a break point currently exists at the step, DirectTalk/2 removes the break point and removes the asterisk from the Break field.
Handling System Error Return Codes

In a DirectTalk/2 voice program, the variable `handle_acterr` can be set either to 0 or to a valid step number. If it is set to 0 or an invalid step number, the DirectTalk/2 system handles the error.

If the DirectTalk/2 system is allowed to handle the system error return code, the voice program is terminated.

If the `handle_acterr` variable is set to a valid step number, the voice program branches to this step number and handles the error. For example:

```
10 Set_Variable Set variable handle_acterr to 900
20 Wait_for_Call
30 Get_Tone_String
   .
   .
   .
900 Comment Handle error
   .
   .
990 Return_from_Appl
```

User actions use the return code `EDGESYSERR` if they are returning a system error.
Chapter 8. More About Voice Logic Modules

In Chapter 4 we created a voice logic module to play some simple segments. This chapter gives more details of the functions that can be included in a voice logic module.

The window in which you enter the voice logic module details is the **Insert Line** window. (See “Creating the Voice Logic Modules” on page 26 for details of how to reach this window.)

![Insert Line Window](image)

### Voice Logic Module Functions

The **Function** field in the **Insert Line** window can contain either of two values:

- **PLAY**: To play a recorded voice segment or one of a number of special prerecorded segments.
- **IF**: To specify the condition under which to perform the next logic statement. If the condition is true, DirectTalk/2 performs the next logic statement; otherwise the next logic statement is omitted.

### Options for the PLAY Function

If you use the PLAY function you can specify a number of different segment types and the information required in **Operand 1** depends on the **Type**. Table 7 shows the types and corresponding operands you can specify.

<table>
<thead>
<tr>
<th>Type</th>
<th>Operand 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEG</td>
<td>Voice segment name</td>
</tr>
<tr>
<td>NUM</td>
<td>Number</td>
</tr>
<tr>
<td>STR</td>
<td>Character string</td>
</tr>
<tr>
<td>DOW</td>
<td>Date, play as: Monday</td>
</tr>
<tr>
<td>DMD</td>
<td>Date (play as: Monday June 5th)</td>
</tr>
<tr>
<td>MD</td>
<td>Date (play as: June 5th)</td>
</tr>
<tr>
<td>MDY</td>
<td>Date (play as: June 5th 1989)</td>
</tr>
<tr>
<td>MY</td>
<td>Date (play as: June 1989)</td>
</tr>
<tr>
<td>DAT</td>
<td>Date (play as: Monday June 5th 1989)</td>
</tr>
<tr>
<td>TIM</td>
<td>Time</td>
</tr>
</tbody>
</table>
Table 7 (Page 2 of 2). Types and Corresponding Operands for the PLAY Function

<table>
<thead>
<tr>
<th>Type</th>
<th>Operand 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUR</td>
<td>Number (play as: currency)</td>
</tr>
<tr>
<td>CTR</td>
<td>Number (play as: a counter)</td>
</tr>
<tr>
<td>TXT</td>
<td>Text segment name</td>
</tr>
<tr>
<td>TSR</td>
<td>Text string</td>
</tr>
</tbody>
</table>

Notes:

1. You must use single quotes when specifying a literal, and no quotes when specifying a variable.
2. The Condition and Operand 2 fields are not used when you specify a function of play.
3. For some languages, the CTR type will have no effect on spoken numbers.
4. You must have the Text-to-speech feature to use the TXT and TSR type options.

Options for the IF Function

If you use the IF function you must complete all the other fields. The Type field can contain either of the following values:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>when Operands 1 and 2 must contain a variable or a literal number</td>
</tr>
<tr>
<td>STR</td>
<td>when Operands 1 and 2 must contain a variable or a literal character</td>
</tr>
</tbody>
</table>

The Condition field can contain any of the values shown in Table 8.

Table 8. Conditions and Their Corresponding Meanings

<table>
<thead>
<tr>
<th>Condition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>Operand 1 equals Operand 2.</td>
</tr>
<tr>
<td>NE</td>
<td>Operand 1 is not equal to Operand 2.</td>
</tr>
<tr>
<td>GT</td>
<td>Operand 1 is greater than Operand 2.</td>
</tr>
<tr>
<td>GE</td>
<td>Operand 1 is greater than or equal to Operand 2.</td>
</tr>
<tr>
<td>LT</td>
<td>Operand 1 is less than Operand 2.</td>
</tr>
<tr>
<td>LE</td>
<td>Operand 1 is less than or equal to Operand 2.</td>
</tr>
</tbody>
</table>

Note: Use single quotes when specifying voice segment names or literal values.

You can also use the IF function with the last_repeat system variable to play certain warnings.

Editing Voice Logic Modules

You can edit voice logic modules to:

- Change module parameters
- Insert logic statements
- Change logic statement details
- Delete logic statements
The general procedure for editing a voice logic modules is:

1. Select **Voice logic module editor** in the **Editors** pull-down in the **Voice Application Developer** window to display the **Voice Logic Module Editor** window for the current application.

2. Select **Add or Change** from the **Edit** pull-down.

3. Display the module you want to edit by entering the name, or by pressing <F2> to select from the list of available modules.

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Operand 1</th>
<th>Cond</th>
<th>Operand 2</th>
<th>More:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAY</td>
<td>SEG</td>
<td>'choices'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>NUM</td>
<td>last_repeat</td>
<td>EQ</td>
<td>'1'</td>
<td></td>
</tr>
<tr>
<td>PLAY</td>
<td>SEG</td>
<td>'warning'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>NUM</td>
<td>last_repeat</td>
<td>EQ</td>
<td>'1'</td>
<td></td>
</tr>
<tr>
<td>PLAY</td>
<td>NUM</td>
<td>timeout_value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>NUM</td>
<td>last_repeat</td>
<td>EQ</td>
<td>'1'</td>
<td></td>
</tr>
</tbody>
</table>

4. Make the required changes to the voice logic module.

5. Press <F4> to file the changes you have made to the voice logic module.

6. Press <Esc> or <F12> to cancel any changes you have made to the voice logic module. DirectTalk/2 prompts you before discarding the changes:
   - Press <F4> to file the changes.
   - Press <Enter> to discard the changes.

If you want more information on any of the editing procedures refer to the online help supplied with the Voice Application Developer.

**Changing Module Parameters**

The module parameters are:

- Repeat value
- Time-out value
- Comment text

To change any of these:

1. Press <F2>.
2. Type in the new values.
3. Press <Enter>. 
Inserting Logic Statements
To insert a logic statement:
1. Select the statement that will precede the statement you want to insert.
3. Enter the values for the new statement.
4. Press <Enter>.

Changing Logic Statement Details
To change the details of a logic statement:
1. Select the statement you want to change.
2. Press <Enter>.
3. Type in the new values for the statement.
4. Press <Enter>.

Deleting Logic Statements
To delete a logic statement:
1. Select the statement(s) you want to delete.
2. Press <F5>.
3. Press <F5> in the Delete Line window to confirm that you want to delete the statement(s).

Adding Voice Logic Modules
The procedure for adding new voice logic modules is explained in detail in “Creating the Voice Logic Modules” on page 26.

Deleting Voice Logic Modules
To delete a voice logic module:
1. Select Delete from the Edit pull-down in the Voice Logic Module Editor window.
2. Display details of the module you want to delete by entering the name, or by pressing <F2> to select from the list of available modules.
3. Press <F5> to delete the module.

Copying Voice Logic Modules
Instead of writing new voice logic modules you can use one that exists and modify it if necessary. The Copy to and Copy from options on the Edit pull-down menu in the Voice Logic Module Editor window enable you to:
- Copy within the current application:
- Copy from another application to the current application:
- Copy to another application from the current application:
Testing Voice Logic Modules

DirectTalk/2 provides a facility for testing voice logic modules before you incorporate them into your voice logic program. This can be particularly useful if you are creating complex voice logic modules and programs.

To test a voice logic module:

1. Select Test in the Edit pull-down in the Voice Logic Module Editor window. DirectTalk/2 displays the Test Voice Logic Module window.

2. Type the name of the voice logic module you want to test, or press <F2> to select from a list of voice logic modules for the current voice application.

3. If you do not have a telephone connection, DirectTalk/2 displays the Make Phone Connection window. Dial the number (or line) that is indicated on the screen to make the telephone connection.

4. If there are any variables defined in the voice logic module, DirectTalk/2 prompts you for a value.

5. Specify a value for each variable DirectTalk/2 displays.

6. When the module runs correctly, press Esc or F12 until you return to the Voice Logic Module Editor window.

Printing Voice Logic Modules

You can print a listing of all the voice logic modules in your current application by selecting Print from the File pull-down of the Voice Logic Module Editor window.

All the general information about printing listings of voice applications (see “Printing the Voice Application” on page 56) applies to printing voice logic modules. The listing contains all the details of:

- The voice logic module parameters
- The logic statements contained in each module
- All voice segments called by the voice logic modules, including the node and database where the segment is stored
Using Common Voice Logic Modules

If you have more than one application, you may find that you need the same voice logic modules for more than one application. Rather than creating these voice logic modules for each application or copying them from one application to another, you can share these common voice logic modules between two or more applications by placing the voice logic modules in a common voice application.

Normally, DirectTalk/2 retrieves voice logic modules from the voice logic module database associated with the application. If the module is not found, DirectTalk/2 then searches for a Common application voice logic module database, and finally it searches the System voice logic module database.

You should create this common application as a new application using any name you choose. This application should contain common voice logic modules, but does not need to be a complete application as it is merely acting as a database.

**Note:** The same common application can also hold any common voice segments as well. See “Using Common Voice Segments” on page 72.

Once you have created the new application, copy or create all the common voice logic modules. A particular voice logic module should be in either a specific application or in the common application that is used by the application, but not in both.

To enable DirectTalk/2 to identify a common application to associate with your application, you must add and set the following variables in your Voice Application Developer setup and in the application control file:

- `cmn_logic_appl`: The name of the application which contains the common voice logic modules
- `cmn_logic_srvr`: The network name of the voice logic module server that the common application uses

For more information about setting these parameters, see Chapter 5, “Variables for Voice Applications” on page 43.
Voice Logic Module Currency Variable

Variations in the way numbers are interpreted as currency are controlled by the `dtlk_edit_x` variables, where `x` is the language letter as defined in the TAMDLANG.NME file. `dtlk_edit_x` has the following format:

```
Z D R C N
```

where:
- **Z**: Is the zero minor currency spoken character (Y,N).
- **D**: Is the implied decimal point (Y,N).
- **R**: Is reserved.
- **C**: Is the alternate calendar (Y,N).
- **N**: Is the number gender (F, M, N).

### Minor Currency Processing

<table>
<thead>
<tr>
<th>If Character 1(Z) is set to</th>
<th>This happens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y or y</td>
<td>The zero minor currencies are spoken for input strings of 'N', 'N.', 'N.0', and 'N.00', where N is an integer value of major currency.</td>
</tr>
<tr>
<td>N or n</td>
<td>The zero minor currencies are not spoken for input strings of 'N', 'N.', 'N.0', and 'N.00', where N is an integer value of major currency.</td>
</tr>
<tr>
<td>NULL or other than Y, y, N, n</td>
<td>The zero minor currencies are spoken or not spoken as defined for the language for input strings of 'N', 'N.', 'N.0', and 'N.00'.</td>
</tr>
</tbody>
</table>

### Implied Currency Decimal

<table>
<thead>
<tr>
<th>If Character 2(D) is set to</th>
<th>This happens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y or y</td>
<td>The last two digits of a currency integer string will be spoken as the minor currency.</td>
</tr>
<tr>
<td>Other than Y or y</td>
<td>A currency integer string will be treated as major currency only.</td>
</tr>
</tbody>
</table>

---

1. If you are using the Arabic language, you can enter a number from 1 through 7 corresponding to the day of the week to be spoken.
Chapter 9. More About Voice Segments

In the sample application we created and recorded a simple voice segment. This chapter describes some other ways of creating and improving voice segments.

Adding and Changing Voice Segments

The procedure for adding new voice segments is explained in detail in “Creating and Recording Voice Segments” on page 33.

You can also use this procedure to change an existing voice segment if you use the name of a segment that already exists, rather than supplying a new one.

Deleting Voice Segments

To delete a voice segment:

1. Select Delete from the Edit pull-down in the Voice Segment Editor window.

2. Display details of the segment you want to delete by entering the name, or by pressing <F2> to select from the list of available segments.

3. If you want to verify that you have the correct segment, you can press <F11> to listen to the segment as described in “Playing Voice Segments.”

4. Press <F5> to delete the segment.

Playing Voice Segments

To listen to voice segments that have already been recorded:

1. Select Play from the Edit pull-down in the Voice Segment Editor window.

   If you do not have a telephone connection, DirectTalk/2 displays the Make Phone Connection window.

2. Dial the number (or line) that is indicated on the screen to make the telephone connection.

3. Select the segment you want to delete by entering the name, or by pressing <F2> to select from the list of available segments.

4. DirectTalk/2 plays the voice segment through the telephone, and displays the text on the screen.

Copying Voice Segments

Instead of writing new voice segments you can use ones that exist and modify them if necessary. The Copy to and Copy from options on the Edit pull-down menu in the Voice Segment Editor window enable you to:

- Copy within the current application
- Copy from another application to the current application
- Copy to another application from the current application
Printing Voice Segments

You can print the text of all the voice segments in your current application by selecting **Print** from the **File** pull-down of the **Voice Segment Editor** window.

All the general information about printing listings of voice applications (see “Printing the Voice Application” on page 56) applies to printing voice segments.

Using System-Supplied Voice Segments

DirectTalk/2 includes a database of common voice segments that play numbers, letters, days of the week, and other common terms. These segments are stored in the system application (SYSTM).

To see a list of the voice segments that are available, select **SYSTM** from the **Open Application** window of the **Voice Segment Editor**.

The voice segments are stored in the same format as the segments that you record yourself, and you can work with them in the same way.

You can rerecord these voice segments to match the voice used in the rest of the voice application but be sure that you do not change their meaning or intent.

Using Common Voice Segments

If you have several applications, you may find that you need the same voice segment in more than one; for example, the initial greeting spoken by the applications or the hours of operation.

Rather than create these voice segments for each application or copy them from one application to another, you can share these common voice segments among applications by placing the voice segments in a common voice application.

Normally, DirectTalk/2 retrieves voice segments from the voice segment database associated with the application. If the segment is not found DirectTalk/2 then searches for a Common application voice segment database, and finally it searches the System voice segment database.

You should create this common application as a new application using any name you choose. This application should contain common voice segments, but does not need to be a complete application as it is merely acting as a database.

**Note:** The same common application can also hold any common voice logic modules as well. See “Using Common Voice Logic Modules” on page 68.

Once you have created the new application, copy or create all the common voice segments. A particular voice segment should be in either a specific application or in the common application that is used by the application, but not in both.

To enable DirectTalk/2 to identify a common application to associate with your application, you must add and set the following variables in your Voice Application Developer setup and in the application control file:
The name of the application which contains the common voice segments

The network name of the voice segment server that the common application uses

For more information about setting these parameters, see Chapter 5, “Variables for Voice Applications” on page 43.

Editing Voice Segments

Once you have recorded the voice segments for the application, you can use the Voice Pattern Editor to improve the way they sound by:

- Deleting silence and unwanted noise
- Inserting silence

You can start the voice pattern editor by any of the following three methods:

1. Select Change voice pattern from the Edit pull-down of the Voice Segment Editor window.
2. Press <F5> in the Add or Change Voice Segment window.
3. Press F5 from the Record Voice Segment window.

In the Voice segment field, type the name of the voice segment you want to display, or press F2 to select from a list of the voice segments for the current voice application.

DirectTalk/2 displays the voice segment text, plays the voice segment, and displays the voice segment as a waveform graph.
The height of the bars indicates the relative volume of the sound recorded. The shortest bars indicate silence. You may see noise spikes in the leading and trailing parts of a message. This often results from hitting the telephone handset against something as you are recording. Also, silence can appear as more than one block if you record the voice segment in a noisy room.

If the voice pattern is longer than the width of the window, DirectTalk/2 displays the word More and a right arrow in the upper right corner of the voice pattern window. You can scroll the window to the right (to the end of voice pattern) and to the left (to the beginning of the voice pattern).

A highlight block indicates the part of the voice pattern that any actions will be applied to. This block can be positioned using the left and right arrow keys or mouse button 2.

You can delete silence or noise from the beginning, end, or within voice segments. Silence can also be added at any point in the voice segment.

To delete a portion of the voice pattern:

1. Press <F7> (left) or <F8> (right) as many times as necessary to scroll to the part of the voice pattern you want to delete
2. Position the highlight block on the part of the voice pattern you want to delete.
3. Press:
   - <F5> to delete the highlighted block. DirectTalk/2 deletes 1/50 of a second each time you press F5.
   - <F9> to delete all the pattern to the left of the highlight block.
   - <F10> to delete all the pattern to the right of the highlight block.

Press <F11> to play the voice segment after you delete the portion of the pattern. While playing the voice segment, DirectTalk/2 adds a tone at the beginning and end of the voice segment to help you determine the amount of leading and trailing silence.
If you delete too much of the voice pattern you can recover in either of the following ways:

- Press <F12> to cancel the changes.
- Press F6 to insert silence back into the voice pattern. DirectTalk/2 inserts 1/50 second of silence into the voice pattern each time you press <F6>.

Press <F4> to save the changes to the voice segment.

Recording or Editing Multiple Voice Segments

You may want to record or edit a number of voice segments at one time, and DirectTalk/2 provides a facility to help with repeated recording and editing.

To record or edit a number of voice segments in succession:

1. Select Multiple Record from the Edit pull-down in the Voice Segment Editor window to display the Build Segment Name List window.

   Build Segment Name List
   
   Include in list:
   All Segments
   Text Only Segments
   
   F1=Help F12=Cancel

2. Select:
   - **All Segments**
     To record or edit all voice segments in the current application
   - **Text only Segments**
     To record or edit all segments in the current application that have no voice recorded

   DirectTalk/2 displays a list of all the requested segments.

3. Select the first segment you want to work with to display the Record Voice Segment window.

   Record Voice Segment: welcome
   
   Add/Change voice segment text:
   Welcome to the telephone calculator
   
   F1=Help F2=List F4=File Text F5=Edit Pattern F7=Previous F8=Next F10=Record F11=Play F12=Cancel

4. Follow the procedures described in “Creating and Recording Voice Segments” on page 33 to record and listen to the segment.
Follow the procedures described in “Editing Voice Segments” on page 73 to edit the segment.

5. When you have finished working with this segment, you can:
   - Press <F7> to move to the previous segment in your list.
   - Press <F8> to move to the next segment in your list.
   - Press <F2> to redisplay the list and select another segment.

6. When you have completed the recording or editing of all your segments, press <F12> or <Esc> to return to the Voice Segment Editor window.

---

**Recording Voice Segments Using a Tape Player or Microphone**

You can record voice segments for DirectTalk/2 using an audio coupler, such as the Dialogic AC/101 Audio Coupler or PromptMaster, with a tape player or a microphone. This feature enables you to exercise greater control over how your recordings sound, including better clarity and reduced background noise.

For example, you may record your voice segments at a studio, using the most advanced recording techniques. If you use languages other than English, you may have professionals record these voice segments in the appropriate countries, then install the voice segments on your system anywhere in the world.

You can use any new voice segments you create using this enhancement with any of your existing voice applications.

To record the voice segments, perform the following steps:

1. You must have the following equipment:
   - Audio coupler (AC/101, PromptMaster, or similar)
   - Analog telephone
   - Telephone cable
   - Tape player or microphone and amplifier

2. Connect the analog telephone to the telephone port on the audio coupler.

3. Connect one end of the cable to the Dialog/40 port on the coupler and the other end to one of the ports on the DirectTalk/2 system.

4. Start Telephony Server Configuration.

5. Open the Network Interface parameter set.

6. Divide the LSI channels to select the channel that has the AC/101 connected.

7. Open the channel being used with the AC/101.

8. Change the Connection Type parameter to “Permanent Connection”. (Remember to reset this parameter to “Switched Connection” when you have finished recording.)

9. Access the Voice Application Developer. Make sure that the Voice Application Developer is using the line that is connected to the audio coupler. You may have to update the voice_line parameter of the line to which the audio coupler is connected. Select **Application Developer setup** in the Options pull-down in the Node Manager window to update this parameter.

10. Access the Voice Segment Editor and select **Add** in the Edit pull-down. When you add the voice segment, DirectTalk/2 takes the line off-hook and does not
ask you to dial a number. Use the # key on the telephone attached to the audio coupler to start and stop recording.

You can use the audio coupler with all the functions of the Voice Segment editor. This feature works best with an amplified device (a tape player or a microphone and amplifier). You can record directly from the telephone attached to the audio coupler.
Chapter 10. Call Transfer and Referral

If you are connected to a public or private switch, DirectTalk/2 includes the capability to extend a call by connecting another party, referred to in this document as an agent, into the call. The call extension can take one of two forms:

**Referral** This takes place when the caller is connected to the agent, but the voice system remains active in the call and expects to take the call back from the agent.

**Transfer** This takes place when the voice system drops out of the call some time after attempting to connect the agent.

Call transfer can be carried out in two ways:

**Blind** In this case the new number is dialled and the voice system does not check whether the new connection is successful.

**Screened** In this case the voice system checks the progress of the connection to the agent.

Depending on your network and system configuration, you may be able to extend the call using just the original call line, or a second line may need to be used to enable the extension.

**Single Line**
This is done by the voice system first alerting the switch and then dialling the transfer number. The switch control is either through the original phone connection, or through an external computer or switch link.

**Dual Line**
The voice system uses a second line to alert the switch and dial the agent. When the agent answers, commands are sent to the switch to connect the two lines together.

### Call Extension on Analog and T1 Lines

On analog and T1 lines, single line transfers can be made using the Put_Tone_String and Place_a_Call system actions. In DirectTalk/2 Put_Tone_String provides a blind transfer, and a screened transfer can be effected using the Place_a_Call action. The correct control sequence must be sent to alert the switch before placing the extended call. See the call descriptions in Chapter 20, “List of Actions” on page 159 for details of how to perform the call extension.

### Call Extension on E1 Lines

There are a number of extend call actions to control transfer and referral of calls. In DirectTalk/2 Version 2.1 these actions can only be used on E1 lines controlled by 30 or 60 port Aculab cards. The Aculab matrix switch is used to connect the referral or transfer channels, and all the channels involved in a referral or transfer must be on a single card.

Currently only dual line E1 transfer and referral is supported. This means that a true transfer where the voice system is removed from the call is not possible. The
Call Transfer operation requires DirectTalk/2 to keep monitoring the call until hangup is detected. This may have implications for an application which has to respond to other resources, particularly a host session, which may time out if there is no input for an extended period.

In addition, standard E1 operation does not allow a caller to be put on hold at the switch. To simulate the on hold situation, DirectTalk/2 reroutes the connection to the caller so that the caller is unable to hear the output from the voice system.

The exact operation of the extend call actions depends on the system configuration. The system variables can be set by the extend call actions, to indicate the the state of the system, or you can use the `Call_Extend_Cfg` action to determine the current state.

Which actions you should use depends on whether you want to refer or transfer the call, and whether you want to speak to the agent before connecting the caller.

**Referring a Call**

If you want to refer the incoming call to an agent, use the following action:

**Call_Referral** To effect a single step referral.

When this action completes, the caller, agent, and voice system are all connected together. The voice system attempts to enable all three parties to talk to and hear each other, but the ability to do this is dependent on the switch and system configuration. The caller and agent are always able to talk to and hear each other and the voice system can always hear the agent.

If you want to talk to the agent before connecting the caller and agent together (this is known as whisper as the caller cannot normally hear the voice system/agent interaction), use the following sequence of actions:

**Call_Extend_Init** To start the call extension.

When this action completes, the caller should be on hold and the voice system and agent are connected together. (If the configuration does not allow the caller to be put on hold, the caller is left connected to the voice system.)

**Call_Extend_Ref** To complete a call referral.

When this action completes, the caller, agent, and voice system are all connected together. The voice system attempts to enable all three parties to talk to and hear each other, but the ability to do this is dependent on the switch and system configuration. The caller and agent are always able to talk to and hear each other and the voice system can always hear the agent.

There are two other actions that you can use with call referral:

**Call_Agent_Rel** To release the agent from the call and return to the original caller voice system connection.

**Wait_for_Hang** To detect when either the caller or agent hangs up a referred call.
Transferring a Call

If you want to transfer the incoming call directly to the agent use the following action:

**Call_Transfer** To effect a single step call transfer.

A full screened transfer is performed. Also, with DirectTalk/2 Version 2.1, this action will not result in a complete transfer as the voice system performs the switching.

If you want to whisper (see the description of whisper on page 80) to the agent before connecting the caller and agent together use the following sequence of actions:

**Call_Extend_Init** To start the call extension.

When this action completes, the caller should be on hold and the voice system and agent are connected together. (If the configuration does not allow the caller to be put on hold, the caller is left connected to the voice system.)

**Call_Release** can also be used to end the voice system call to the agent if the agent does not want to accept the call transfer.

**Call_Extend_Transfer** To complete a call transfer.

In DirectTalk/2 Version 2.1 the voice system performs the switching and this action will not complete until one of the parties hangs up.

Call Transfer Application Example

The following simple example shows a whisper to and response from the agent and options to complete the transfer or release the agent depending on the agent's response:

<table>
<thead>
<tr>
<th>Action</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>Call_Extend_Init</td>
<td>initiate transfer to agent.</td>
</tr>
<tr>
<td>Play_Module</td>
<td>whisper to agent</td>
</tr>
<tr>
<td>Get_a_Tone</td>
<td>get response from agent</td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>Call_Extend_Transfer</td>
<td>agent agrees, complete transfer</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>Call_Agent_Rel</td>
<td>agent refused, return to caller</td>
</tr>
</tbody>
</table>

E1 Extend Call System Variables

The Extend Call actions set and use the following system variables:

**ext_connect_type (Output)**

The type of agent connection that was detected. Possible values are PERSON, ANSWER, FAX, OPERATOR, or NONE. NONE is set if no connection is established.
ext_num_of_lines (Output)
The number of lines that are currently being used by the application.
Possible values are:

0 No caller
1 No extended call, or the extended call only requires a single line
2 The extended call is using two lines

ext_canplayto_a (Output)
Indicates whether output from the voice system can be heard by the agent.

ext_canhear_a (Output)
Indicates whether input from the agent can be heard by the voice system.

ext_canplayto_c (Output)
Indicates whether output from the voice system can be heard by the caller.

ext_canhear_c (Output)
Indicates whether input from the caller can be heard by the voice system.

ext_monagent_hup
Indicates whether it is possible to monitor the agent for hang-up.

Note: This is set according to the ability of the network interface to detect disconnection using signalling. It may be possible to detect hang-up using other noise based methods.

ext_moncall_hup (Output)
Indicates whether it is possible to monitor the caller for hang-up.

Note: This is set according to the ability of the network interface to detect disconnection using signalling. It may be possible to detect hang-up using other noise based methods.

ext_switched_at (Output)
Indicates where the switching for the current extend call took place, or where the switching for the previous extend call took place. Possible values are:

-1 No extended call established
1 PBX
2 Voice system
3 Adapter card

ext_fulltran_pos (Output)
Indicates whether the current extend call network connection will allow a transfer without the voice system monitoring until the caller or agent disconnects.

0=No (or no extended call), and 1=Yes.

ext_referral_pos (Output)
Indicates whether it is possible to move to a referral call with the current network connection.

0=No (or no extended call), and 1=Yes.
The expected state of the variables after each call, depends on the mode of operation. Table 9 shows the expected settings after each action for Standard E1 Operation.

<table>
<thead>
<tr>
<th>System variable</th>
<th>call extend init</th>
<th>call extend tran</th>
<th>call extend ref</th>
<th>call release</th>
<th>call transfer</th>
<th>call refer</th>
<th>wait for hang</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext_connect_type</td>
<td>type</td>
<td>NONE</td>
<td>n/c</td>
<td>NONE</td>
<td>type</td>
<td>n/c</td>
<td></td>
</tr>
<tr>
<td>ext_num_of_lines</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>n/c</td>
</tr>
<tr>
<td>ext_switched_at</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>-1</td>
<td>3</td>
<td>3</td>
<td>n/c</td>
</tr>
<tr>
<td>ext_fulltran_pos</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>n/c</td>
</tr>
<tr>
<td>ext_referral_pos</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>n/c</td>
</tr>
<tr>
<td>ext_canplayto_a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0/1</td>
</tr>
<tr>
<td>ext_canhear_a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0/1</td>
</tr>
<tr>
<td>ext_canplayto_c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0/1</td>
</tr>
<tr>
<td>ext_canhear_c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0/1</td>
</tr>
<tr>
<td>ext_monagent_hup</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>n/c</td>
</tr>
<tr>
<td>ext_moncall_hup</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>n/c</td>
</tr>
<tr>
<td>ext_line2_trunk</td>
<td>tr num</td>
<td>0</td>
<td>n/c</td>
<td>0</td>
<td>0</td>
<td>tr num</td>
<td>n/c</td>
</tr>
<tr>
<td>ext_line2_ts</td>
<td>ts num</td>
<td>0</td>
<td>n/c</td>
<td>0</td>
<td>0</td>
<td>ts num</td>
<td>n/c</td>
</tr>
</tbody>
</table>
Chapter 11. Password Security

If you are working in a multiuser environment, or plan to distribute your applications, you may want to restrict the access to all or some of your application files. The VAD password facility enables you to add a password to your application files. The password is then required before any of the following functions can be used with the password protected files:

- Voice Program Editor
- Voice Logic Module Editor
- Voice Segment Editor
- Text Segment Editor
- ADSI Script Database
- Voice Application Developer Check
- Voice Application Developer Debug
- Voice Application Developer Print
- Password Manager

To use the password facility, select Password manager from the Voice Application Developer Utilities pull-down menu to display the Manage Application Password window.

This window contains the following fields:

**Application**

The name of the application for which the password is being managed.

- Valid value: any defined application.
- Default: current application.

**File Type**

The type of application file to be password-protected.

- Valid Values: voice program, voice logic module, voice segment, text segment, ADSI script, All Types.
- Default: All Types.

**Language**

The language of the files to be password-protected. If the File Type field contains All Types, this field is ignored.

- Valid value: any defined language.
- Default: current language active in the Voice Application Developer.
**Current Password**  The existing password for the application. This field is required to delete or change a password. The password is not displayed.

Valid value: the valid password for the application.

Default: blank.

**New Password**  The new password for an application. This field is required to add or change a password. The password is not displayed.

Valid value: any character string.

Default: blank.

**Verify Password**  The password to be applied to the application. This field is used to verify that the new or changed password was typed correctly. The field is required to add or change a password. The password is not displayed.

Valid value: any character string.

Default: blank.

Password management performs the functions listed in Table 10, but only if the required prerequisites are in place:

<table>
<thead>
<tr>
<th>Function</th>
<th>Appl Name</th>
<th>Current PW</th>
<th>New PW</th>
<th>Verify PW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Password</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Delete Password</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>change password</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

If the **New Password** and **Verify Password** fields are used, they must be the same. If the **Current Password** field is used, it must be the same as the password currently defined for the application.

If you want to password protect databases created with versions of DirectTalk/2 earlier than Version 1.1.1, you must use the compress utility to enable the databases for password protection.

The application databases to be password-protected will be the voice program, voice logic module, voice segment, text segment and ADSI script databases.

When you want to delete a password, DirectTalk/2 only removes it from each application database if the password in the application databases matches the current password specified in the **Manage Application Password** window. When you want to add a password, DirectTalk/2 only adds it to all the application databases if the database does not have a password already defined. When you want to change a password, DirectTalk/2 only writes it to each of the application databases if the password in the application databases matches the current password specified in the **Manage Application Password** window.
**Note:** It is possible to assign different passwords to different files of a single application or only assign passwords to some of the files. You should take care in your password management and be aware of this situation.

**Password Prompt**

If the application is password-protected, and any of the protected functions are attempted, the **Application Password** window appears and DirectTalk/2 prompts you to enter the password. This occurs only once per active application.
Chapter 12. DirectTalk/2 REXX Environment

This chapter describes the DirectTalk/2 REXX Environment, which provides a REXX interface for developing voice programs.

**Note:** You must have OS/2 REXX installed on your system to use this part of DirectTalk/2. If you selected a full installation when you installed OS/2 then OS/2 REXX was automatically installed. You can check if OS/2 REXX is installed by verifying the presence of the file REXX.DLL in the OS2\DLL directory. If OS/2 REXX is not installed, you must perform an OS/2 Selective Install to install REXX before you can use this part of DirectTalk/2.

The voice program actions are provided for use within OS/2 REXX without modification to REXX or to the action source files. The REXX voice program actions accept the same parameters and generate the same return codes as the actions within a DirectTalk/2 voice language program. Any exceptions to this are limited to those imposed by the REXX language and any others documented in this book.

The DirectTalk/2 REXX Environment provides for the execution of a voice program written in the REXX language and allows that REXX voice program to use the voice program actions. The actions and supporting functions are used unchanged in the DirectTalk/2 REXX Environment. Voice Logic Modules and Voice Segments must still be created using the facilities of the Voice Application Developer.

REXX voice programs can be run in two ways:

- From the Session Monitor.
  
  The REXX environment supports REXX voice program execution from within the Session Monitor as currently supported by the Application Manager. This means that the Session Monitor is not required to treat either voice program environment any differently.

- From the OS/2 command line.

REXX voice programs can be run from the Application Manager as described in *IBM CallPath DirectTalk/2 Administrator’s Guide*, but they can also be run from the OS/2 command line. The following section describes how to run REXX voice programs from the OS/2 command line.

**REXX Sample Voice Applications**

The REXX source for the MENU and CALC sample applications which are referred to in this chapter, are provided on the DirectTalk/2 CD-ROM.
Command Line Processing

The REXX environment supports REXX voice program execution from the OS/2 command line using the following command syntax:

\[
\text{T-REXX control_file positional_options keyword_option}
\]

where control_file is the name of a control file which is input to the Session Monitor. It is the same control file as currently used for voice program execution.

The positional_options accepted by T-REXX are:

\[
\text{msg_lang mode client_name max_restarts action_trace}
\]

where:

- **msg_lang** is the message language character to use if a letter cannot be found in the global control file. The default is E.
- **mode** is the runtime mode of the REXX Voice Program. This value is either SESSION (invoked from the Session Monitor) or COMMAND (invoked from the command line). The default is COMMAND.
- **client_name** is the client name. The default value is based upon the node name and the process id.
- **max_restarts** is the maximum number of restarts that can occur when running under the Session Monitor. The default value is zero.
- **action_trace** enables the application trace (if set to 1) or no trace (if set to 0). The default is 0.

**Note:** The term positional_options implies that the meaning of an option is determined by its position on the command line. So if, for example, you want to set just client_name to something different from the default value, you would need to input the default values for msg_lang and for mode so as to correctly position the client_name option.

The keyword option supported is \(-l\), the log file name. The default is OS2ppp.LOG where ppp is the OS/2 process ID for the T-REXX. The file name can be any legal OS/2 FAT file system file name including drive and path specifications. The value "display" is also supported and causes all REXX Voice Program messages to be displayed on the screen. This only affects REXX voice program messages. REXX functions such as "SAY" remain unaffected.

T-REXX uses the Application Control File as supported by the Voice Application Executor. The control file can be created and edited using the Voice Application Developer.

The REXX voice program file name can have a maximum length of 5 characters. The file extension for any REXX voice program is either .TBT or .TRX. The full file name of the REXX voice program to be executed is the name specified by the initial_appl variable with an extension of either .TBT or .TRX. For the main application (specified by the init_appl variable in the Application Control File), T-REXX first attempts to load the binary .TBT file, and then the ASCII .TRX file. (See “Protecting REXX Voice Programs” on page 96.)
The TREXX function

The TREXX function provides the interface from the REXX Voice Program to the DirectTalk/2 actions. It takes the following form:

\[
\text{TREXX_RC} = \text{TREXX}(\text{action_name}, \text{parm1}, \text{parm2}, \text{parm3}, \text{parm4})
\]

where

- **TREXX_RC** is the return code from the DirectTalk/2 action. It has a value of 0 to 14. Some predefined values are created at initialization time to make the testing of the return code read more meaningful. These are:
  - TREXX_RC0 through TREXX_RC14
  - TREXX_STAR (same as TREXX_RC10)
  - TREXX_POUND (same as TREXX_RC11)
  - TREXX_HASH (same as TREXX_RC11)
  - TREXX_T1 (same as TREXX_RC12)
  - TREXX_T2 (same as TREXX_RC13)
  - TREXX_HUP (same as TREXX_RC14)

- **action_name** is the name of the DirectTalk/2 or user action; for example, Play_Module. The following actions are not supported and are treated as though the action did not exist:
  - Link_to_Appl Action (see “REXX Subroutines and Functions” on page 95)
  - Return_from_Appl Action (see “REXX Subroutines and Functions” on page 95)

- **parm1** is the first parameter for the action, if required.
- **parm2** is the second parameter for the action, if required.
- **parm3** is the third parameter for the action, if required.
- **parm4** is the fourth parameter for the action, if required.

For example:

\[
\text{TREXX_RC} = \text{TREXX}(\text{Play_Module}, \text{'goodbye'}, \text{'no'});
\]

\[
\text{if TREXX_RC} = \text{TREXX_HUP then}
\]

\[
\text{do}
\]

\[
\text{.}
\]

\[
\text{.}
\]

\[
\text{end}
\]

The syntax for literals and variables is the same as for a voice program; that is, literals are enclosed in either single quotes (') or double quotes ("'). The exception is that as a REXX program, the REXX Voice Program conforms to the rules of REXX. For example, if performing the Set_Variable action and the value variable has not been set in the REXX program, REXX will have assigned a value to the variable equal to the name. As such, the Set_Variable action will not return an error but, will be successful since it has assigned a value.
DirectTalk/2 REXX Environment Debug Facilities

The DirectTalk/2 REXX Environment supports the application trace as supported in the Voice Program Executor. The trace only traces TREXX actions. Intervening REXX functions and subroutines are not traced; to trace these, use the REXX trace facilities.

Variable Processing

This section provides important information for variable processing in REXX voice programs in general, and in user actions in particular.

Voice Programs

A REXX voice program should use REXX variable processing in preference to the corresponding DirectTalk/2 variable processing actions. REXX variable processing provides more function, and results in better performing and simpler programs.

This is demonstrated by the following, which shows the same program written using both methods. The program creates two variables (x and y), initializes them to integer values, stores the total in a third variable (z), and prints the value of this total variable.

We can achieve this using REXX variable processing, as follows:

```
x=4
y=7
z=x+y
say 'Result='z
```

On the other hand, using DirectTalk/2 variable processing, we have:

```
rcx=trexx(Set_Variable,'x','4')
rcy=trexx(Set_Variable,'y','7')
if (rcx¬=TREXX_RC/zerodot)
do
  say 'Set_Variable error, rc='rcx
end
else if (rcy¬=TREXX_RC/zerodot)
do
  say 'Set_Variable error, rc='rcy
end
else
rc=trexx('Calculate ','z',x,'+',y)
if (rc=TREXX_RC/zerodot)
do
  say 'Result='z
end
else
  do
    say 'Calculate error, rc='rc
  end
end
```

A variable in a REXX voice program is created implicitly by use and, unlike a DirectTalk/2 voice programming language variable, has a default value which is a string equal to the name of the variable.
If a variable is supplied as a parameter to the TREXX functions, REXX always passes the value of the variable. This means that, for an output parameter, where the variable (and not the value) is required, the name of the variable should be supplied as a literal string. This is done by supplying the name in quotes (without quotes the current value is passed). For an input parameter, where the value of the variable is required, the variable should be supplied; this is done by supplying the name without any quotes.

An example of the latter is as follows:

```plaintext
rc=trexx(Calculate,'total',x,'+',y)
```

where `total` is the name of the output variable to hold the result, variables `x` and `y` are input variables whose values are to be summed, and the literal `+` is the operation.

The DirectTalk/2 system variables `SYSTEM_DATE`, `SYSTEM_DOW`, and `SYSTEM_TIME` are not supported as the date, day of the week, or time respectively. Use the REXX functions `DATE` and `TIME` instead. The following examples show how you can set the variables `D`, `DOW`, and `T` to contain the date, day of the week, and time respectively in the format defined for the DirectTalk/2 system variables:

```plaintext
* SYSTEM_DATE (format YYYYMMDD)
  d=date('s')

* SYSTEM_DOW (format 1=Sunday, 2=Monday, etc)
  dow=(date('b')+1)//7+1

* SYSTEM_TIME (format HHMMSS)
  temp=time()
  t=substr(temp,1,2)substr(temp,4,2)substr(temp,7)
```

**User Actions**

Each parameter passed to a user action has a type associated with it. The type is either `PARMCHAR`, which indicates a literal string, or `PARMVAR`, which indicates a variable name. Since REXX always passes the value of a variable, the type will always be `PARMCHAR` for all parameters when the action is called from a REXX program.

This factor means that parameters should be processed as follows:

- **Input parameter**
  
  The type should be used to determine whether the parameter can be used directly (`PARMCHAR`), or is a variable (`PARMVAR`) where the value is required. Since REXX only supplies parameters of type `PARMCHAR`, there will be a small performance gain if this is checked and handled first.

- **Output parameter**
  
  The type should not be used, and the parameter should be treated as a variable; once the return value has been obtained, it is stored in the variable.
• Input/Output parameter

The type should not be used, and the parameter should be treated as a variable; the value of the variable is obtained for input, and once the return value is obtained, it is stored in the variable.

REXX Variable Naming Standards

REXX has standards governing the naming of variables and any variable created or accessed in a REXX program has to meet these standards.

However, it is also possible to create a DirectTalk/2 variable which does not meet these standards by either defining it in a CTL or SU file, or by using the action Create_Variable.

In this case, T-REXX modifies the variable name to make it acceptable to REXX as follows:

• If the name starts with a digit, the character '?' is appended to the front (for example, 3DOG becomes ?3DOG).

• If the name starts with '$$', the '$$' is replaced by '??' (for example, $$DOG becomes ??DOG). Normally, only certain DirectTalk/2 system names begin with '$$'.

• All other invalid characters are replaced by the character '!' (for example, FRED/JOE becomes FRED!JOE).

If a variable with an invalid REXX name is accessed using the DirectTalk/2 variable actions, T-REXX automatically changes the name to the modified name first, which means that the T-REXX program does not have to compensate.

If a variable with an invalid REXX name is accessed directly as a REXX variable then the variable name must be modified first, as described above. Using names which meet the REXX naming standards avoids having to make any name modification, but such modification is necessary for the DirectTalk/2 system variables which do not meet the REXX naming standard.

DirectTalk/2 Global Variable Processing

REXX does not provide global variable support for external subroutines and functions, since it provides subroutine and function parameter passing, with value return.

The DirectTalk/2 voice programming language does provide global variable support, because actions such as Link_to_Appl (which is the equivalent to a REXX subroutine call) do not provide parameter support.

T-REXX provides restricted DirectTalk/2 global variable support for variables defined in the Application Control file.

These variables, with their values as defined in the file, are available across REXX external subroutines and functions which are defined as a REXX Voice Program (the file name has the extension .TRX). However, any updates are purely local and are not carried across to an external subroutine or function: on entry, the variable has the value as defined in the control file.
All variables created using the Create_Variable action are treated as local (even when the type is global) and are not carried across to an external subroutine or function; on entry, the variable is not even defined.

**Global Variable Support for T-REXX Internal Procedures**

DirectTalk/2 has support for DirectTalk/2 global variables in T-REXX by including a new system variable called TREXX_VARS. This variable is maintained by T-REXX and must not be modified by the application. The variable contains the list of DirectTalk/2 global variables and can be used in a PROCEDURE statement in an internal REXX procedure as follows:

```rexx
proname: PROCEDURE expose (TREXX_VARS)
```

TREXX_VARS maintains the list of global variables by adding and deleting entries whenever DirectTalk/2 global variables are created and deleted.

**The TREXX-ENV Environment Variable**

A REXX program can determine how it was started by querying the TREXX environment variable. The name of the variable is:

```rexx
TREXX_ENV
```

and takes the value "COMMAND" if the program was started at the command line, and "SESSION" if the program was started as a session under the DirectTalk/2 Session Monitor.

For example:

```rexx
if TREXX_ENV = "COMMAND" then
    say "Please call "telephone_number;
```

**REXX Subroutines and Functions**

Both internal and external REXX subroutines and functions are generally treated as defined by the REXX language.

If an external call is made and the function or subroutine file name includes the .TRX extension and the file name is a valid voice program name, the function or subroutine is initialized as a REXX Voice Program with the DirectTalk/2 REXX Environment. This replaces the Link_to_Appl Action and allows one REXX voice program to link to another. If the called subroutine or function does not have the .TRX extension or the file name is not a valid voice program name, it is treated as an external call as defined by the REXX language. Its environment does not provide access to the TREXX function. For a REXX Voice Program, the DirectTalk/2 REXX Environment initializes the REXX environment with the global variables from the calling environment and local variables as defined in the setup file, if it exists. The setup file has the name `<applname>.SU`, where `<applname>` is the voice program name.

The following example is called from the MENU REXX voice program which is shipped with DirectTalk/2:

```rexx
if x=4 then
    call calc.trx(x)
else
    rc=resets.trx(x)
```
where the file name and extension of the file containing function "resets" is 
RESETS.TRX and for subroutine "calc" is CALC.TRX. In this example, "calc" is 
treated as a REXX Voice Program and "resets" is treated as a REXX function 
since its file name is too long to be a valid voice program file name. Any TREXX 
calls made in CALC use the environment of the CALC REXX voice program (for 
example, voice logic modules and voice segments come from CALC.PRx and 
CALC.SGx). Any TREXX calls made in RESETS use the MENU REXX voice 
program's environment. For example, voice segments come from MENU.PRx and 
MENU.SGx.

The REXX Voice Program must follow all the rules of the REXX language. This 
includes the termination of the REXX Voice Program. The REXX Voice Program 
should use a RETURN instruction when it is called from within another REXX Voice 
Program. (EXIT can be used from an external subroutine or function.) This is the 
equivalent of the voice program Return_from_Appl action.

Note: A REXX voice program cannot be run directly by the OS/2 program, 
because the file name extension is .TRX and not .CMD. This means that a REXX 
voice program cannot be run at the OS/2 command line, nor can it be run if it is 
passed by REXX as a command to the OS/2 program. Examples of the latter case 
are either enclosing the entire clause in quotes, or assigning the clause to a REXX 
variable and then having REXX execute the variable. Also, if a .CMD file is run in 
this manner from REXX, neither it nor any external subroutines or functions it calls 
can call the TREXX function.

Warning: If an internal subroutine or function uses the PROCEDURE, instruction 
it may not call the TREXX function either directly or indirectly using another internal 
subroutine or function. This is because the PROCEDURE instruction resets the 
REXX variables within the internal subroutine or function. This prevents access to 
the variables owned by its caller, which include the local and global variables pools. 
The TREXX function fails in these circumstances.

Protecting REXX Voice Programs

A REXX voice program is an ASCII command file, which means that there is no 
source protection. However, DirectTalk/2 provides source protection by using the 
DirectTalk/2 Binary Translator utility. The Binary Translator utility takes a T-REXX 
ASCII command file program as input and produces a T-REXX binary translated file 
as output. Both the ASCII and binary files are executable by the T-REXX 
command.

The Binary Translator utility is executed from the OS/2 command line as follows:

`T-REXXBT <infile> (<outfile>)`

where:

<infile> is the name of the input file.
<outfile> is the name of the output file.

If the output file name is not specified, the output file name generated is that of the 
input file with a file extension of .TBT.

The Binary Translator utility provides the following features:

• Comments and blanks are not translated.
• The binary output is random, so successive runs with the same input file produces different results.

• A .TBT file overrides a .TRX file at run time. T-REXX searches first for a file with extension .TBT and, if that is not found, it then searches for a file with extension .TRX.

Notes:
1. Only the main T-REXX can be replaced by an encoded file; any external functions or subroutines must exist as an ASCII T-REXX program.
2. The resulting encoded file must be 64Kb or less. If it is larger, it may not load (an error message will be produced).

Note that there are two alternative ways of providing protection to a REXX voice program:
• Have part of the application in a language such as C and produce from it an .EXE file which can be called from REXX. For heavy processing, this could result in a performance gain.
• Make up the program using user actions, requesters and servers.

Handling System Error Return Codes

In a REXX program, if handle_acterr is set to 1, the error is handled by the voice program. If handle_acterr is set to 0, the error is handled by the DirectTalk/2 system. No other values are permitted.

If the DirectTalk/2 system is allowed to handle the system error return code, the voice program is terminated.

If the REXX voice program is handling such errors, the return code from the action concerned is TREXX_SYSErr. Check for this return code and code the voice program accordingly. For example:

handle_acterr=1 /* Handle errors*/
rc=trexx(wait_for_Call,'1','0')
if rc=TREXX_RC0 then
  .
  .
else if rc=TREXX_RC1 then
  .
  .
else if rc=TREXX_SYSErr then
  .
  .
return
Part 3. Optional Features

This section describes the optional features that you can add to your DirectTalk/2 system.

Chapter 13. Voice Messaging .................................................. 101

Chapter 14. Voice Recognition .................................................. 103
Voice Recognition Configuration Parameters ......................... 104
Vocabulary Files ............................... 106
  vocabname.VBx ............................... 106
  vocabname.TPx ............................... 106
  vocabname.VTx ............................... 108
Application Notes ............................... 108
  Voice Cutthrough ......................... 111
  Cutthrough Voice Recognition Accuracy ......................... 112
  Dealing With Background Noise ................................. 112
Discrete Alphanumeric Recognition ................................. 113
Continuous Voice Recognition ................................. 114
Voice Recognition Testing ................................. 115
  Testing For Accuracy ....................... 115
  Testing The Voice Recognition Configuration ................. 115
Voice Recognition Sample Application ......................... 116
Voice Recognition Actions ................................. 116
VR_Get_String and VR_Get_Yes_No ................................. 117

Chapter 15. Communications .................................................. 123
Emulators Supported ............................... 123
  Emulation Session Groups ............................... 123
Terminal Emulation Applications ................................. 124
  Operator Information Area ............................... 124
Multiple Host Access ............................... 125
Programming Considerations ................................. 125
  Debugging Communications Applications ................. 126
Communications Actions ............................... 126

Chapter 16. Text-to-Speech .................................................. 129
Creating Text Segments ............................... 130
  Using the Text Segment Editor ............................... 130
  Listening to Your Text Segment ......................... 133
Valid Text Segment Characters ............................... 134
Copying Text Segments ............................... 135
Importing and Exporting Text Segments ......................... 135
Using Common Text Segments ............................... 135
Programming Considerations ............................... 136
Text-to-Speech Controls ............................... 137
  BeSTspeech Controls ............................... 137
  Lernout & Hauspie (L&H) Controls ......................... 138
Using Phonetic Input ............................... 140
  Phonetic Input for Lernout & Hauspie TTS Systems ............. 141
  Phonetic Input for BeST Speech TTS Systems ............... 141
  Using the Text-to-Speech Exception Dictionary ............. 142
Chapter 13. Voice Messaging

The optional Voice Messaging Feature enables you to create an application where callers can leave messages for individuals and those individuals can retrieve the messages.

The mailbox directory must be set up before a voice messaging application can be used. Use the Mailbox Manager to do this.

The individual tasks an application can perform are:

- Take messages from callers
- Retrieve the messages and play them to the recipient
- Record a personal greeting
- Update the user information

You usually need to perform the following steps to take messages from callers:

1. Prompt the caller to specify the recipient’s extension.
2. Get the recipient’s extension that the caller specifies.
3. Use the Search_Directory action to retrieve the entry for the recipient. The information in the entry includes:
   - Mailbox number
   - Telephone number
   - Greeting key
   - Recorded name key
4. Use the Take_a_Message action to record the message using the directory information as input.

See “TAKE Application” on page 305 for an example of this process.

To retrieve messages from mailboxes and play them to the mailbox owner:

1. Prompt the user for the extension.
2. Use the Search_Directory action to:
   - Save the mailbox number
   - Save the ID number
3. Prompt the user for the password.
4. Use the Check_Password action to verify the password.
5. Use the Get_Messages action to play the messages to the user if the password matches.

An example of this process is in Appendix B, “Sample Applications” on page 305.
To record a personal greeting:

1. Prompt the user for the mailbox number.

2. Use the Search_Directory action to:
   - Save the mailbox number
   - Save the ID number
   - Save the greeting key

3. Prompt the user for the password.

4. Use the Check_Password action to verify the password.

5. Use the Delete_Voice action to delete any prior greetings.

6. Prompt the user to record the greeting.

7. Use the Record_Voice action to record the message:
   - Use gmsg_msg_server or gmsg_g/n_db variables for server and database name.
   - Use the <prefix>_grtgkey variable for the key value.

8. Use the Clear_Tones action to clear the tone in the buffer after the user ends the recording by pressing a key.

You should consider making each of these tasks subprograms and using the Link_to_Appl action.

The actions provided by the Voice Messaging Feature are:

**Search_Directory**
Retrieve a directory entry based on phone number.

**Check_Password**
Check the password when it is entered.

**Put_User_Info**
Change information in the directory.

**Take_A_Message**
Record telephone messages and store them in a mailbox.

**Get_Messages**
Retrieve messages from a mailbox.
Chapter 14. Voice Recognition

You can use the optional DirectTalk/2 Voice Recognition Feature to specify a limited vocabulary of words the caller can speak and the voice application will recognize. This feature increases the number of potential callers that can use your voice application in areas where many of the callers still have rotary dial (pulse) telephones. A number of different voice recognition technologies are supported by DirectTalk/2:

Discrete Voice Recognition
This type of recognition operates on one word at a time. The caller must leave a gap between words to allow the system time to process each word. A beep is often used to indicate when the system is ready to receive the next word.

Voice Stop
The ability for a caller to interrupt a prompt with a spoken response. The presence of sufficient “noise” stops the prompt. No attempt is made to recognize the word spoken. This is sometimes known as “grunt detection”.

Cutthrough Word Recognition
The ability for a caller to interrupt a prompt with a spoken response that DirectTalk/2 attempts to recognize as a word. This feature allows voice recognition to work similarly to DTMF tone fastpath.

Continuous Voice Recognition
Allows a caller to speak a string of digits at a natural pace without prompt tones for each word.

The type of voice recognition supported on a system is determined by the hardware and the voice recognition vocabularies that are installed. The voice recognition vocabularies also determine which spoken language is recognized.

Three different hardware and software combinations are supported by DirectTalk/2 Version 2.1:

Scott Instruments with VR/41-MC, VR/81-MC, and VR/121
The Scott software is only supported in maintenance mode. Discrete voice recognition only is available with a limited number of vocabularies.

Voice Control Systems with VR/40 and VR/160
Between them, these combinations support a wide range of languages and all the technologies.

Voice Control Systems with Antares Cards
This combination supports a wide range of vocabularies and technologies.

IBM CallPath DirectTalk/2 General Information and Planning contains detailed information on the technology and vocabularies that are available for the various cards.

Regardless of the technology used, the voice recognition process follows the same steps:

1. Collect the utterance.
2. Quantify the utterance.
3. Compare the utterance with a known set of words.
It is important to bear in mind whilst working with voice recognition, that it is part art and part science.

The artistic part is at the beginning of the voice recognition process when we collect the caller's utterance. The utterance is affected by a large number of variables, such as the caller's state of mind, dialect, health, as well as the time familiarity with the system, of day. The caller's utterance is also affected by further external variables, such as the quality of the phone line, the quality of the caller's phone handset, and the amount of background noise.

The scientific part occurs whilst processing the collected utterance. This processing involves digitizing the utterance, examining it, quantifying it, and comparing it with a known set of word templates (the sub-vocabularies). This comparison achieves a very precise set of results where the utterance is rated against the known words, and the likelihood of a match is defined to a very fine degree for every known word in the set.

Voice Recognition is the art ofcountering all of the variables with science so that the end result is accurate, reliable, and repeatable.

The primary device for quantifying a response is the score assigned to each word template for a given utterance. The score is a rating which reflects how closely an utterance matches a word: the lower the score, the closer the match. Think of the score as a penalty point against the utterance each time it does not match the known word in some way. In this way, a word is ranked for a given utterance with the top ranked word (lowest score) being the best match.

Further, an examination of the list of word scores is made to test for an ambiguous response. If two words are scored very closely together, this is considered an ambiguous response. Ambiguous responses are less likely in a smaller sub-vocabulary.

**Voice Recognition Configuration Parameters**

The following three DirectTalk/2 Voice Recognition parameters affect the steps described above, and improve their performance:

- **Silence Threshold**

  **Note:** The Silence Threshold is a parameter specific to voice recognition vocabularies from Scott Instruments. The Noise Threshold has no effect with voice recognition from Voice Control Systems (VCS). Scott Instruments is available on the VR/x1-MC and VR/121 cards. VCS is available on the VR/40, VRP, and Antares cards.

  The Silence Threshold parameter affects the first step in the voice recognition process: collecting the caller's utterance. It determines the level of noise which is rejected as not being part of the utterance (that is, considered to be silence), and is a unitless value.

  The default value is 200 for analog phone lines (for example, for LSI/XXX cards). If you are configured to use a digital connection, such as T1 or E1, this value should be set initially to 400. This value should be changed only if responses are being consistently rejected as too soft (even though the caller is not speaking in a soft voice) or if line noise is triggering the response before a caller has a chance to speak a word. Higher values will filter more noise and
the caller may have to speak louder. However, setting this parameter too high will cause all utterances to be rejected as noise. Typically, this parameter should not change and should never need to go above 800.

For discrete digit vocabularies, this parameter is related to Silence Threshold Ceiling and Minimum Utterance Peak in the following way:

\[
\text{Min. Utt. Peak} = 2.5 \times \text{Silence Threshold} = 2.5 \times \text{Silence Threshold Ceiling}
\]

- **Acceptance Threshold**

  The Acceptance Threshold affects the evaluation of the results of comparing the utterance with the known words.

  Acceptance Threshold determines when an utterance is considered to be unrecognized due to the best matched word having so high a score that it cannot reasonably be considered a match. Think of this as an utterance quality threshold; that is, the quality of the utterance is so poor that no match can be determined.

  The default value for the Acceptance Threshold is 1000. This is sufficiently large to allow for poor connections or for callers with strong accents. You may set the value to a lower value to be more discriminating, or to a higher value if too many utterances are being rejected. The normal range of values is 700-1200. The maximum value is 9999, but it should not normally be set above 2000. The actual value used will be somewhat dependent on the language used and the calling conditions (ambient line noise, background noise, caller's voice volume, and so on).

  Every caller will have a different score range for their utterances within any given vocabulary, although it can be expected that the variations between callers will be relatively small (within 300 or 400). When setting the Acceptance Threshold value, take into account this variance.

  This should not be confused with the Marginal Quality Threshold, which is an internal Voice Recognition parameter.

- **Minimum Difference**

  The Minimum Difference parameter affects the determination of an ambiguous response. If the difference of the two best scores are within this value, the response will be considered ambiguous. The default value is 10, which is a very conservative value. The normal range of this value is 1-10. The maximum value is 999, but it should not normally go above 25. If the value is 0, ambiguous responses cannot be resolved. For most systems, accurate recognition is determined with the value set to either 1 or 2.

  A value of 1 is useful if the application accepts one utterance at a time and performs a verification of each word. The value 1 will allow, in this case, the application to resolve all utterances except those which are truly ambiguous. If you will be using the Enhanced Voice Recognition actions, VR_Get_Yes_No and VR_Get_String, the Minimum Difference should remain nonzero so that ambiguous responses can be resolved by the actions. For these actions, a value of 1 is recommended since the actions will perform verification of the string with the caller.
Vocabulary Files

The speech that can be recognized and decoded by a particular voice recognition card is defined in a set of vocabulary files. For Scott Instruments, and VCS with VR/40 and VR/160 technologies, these files are defined in the DirectTalk/2 configuration and are loaded at DirectTalk/2 startup. For Antares, the vocabularies are defined in the Antares configuration file and are loaded at OS/2 startup. Each language and each voice recognition card requires its own specific set of vocabularies. The Voice Recognition feature of DirectTalk/2 contains all the vocabularies required by the officially supported languages and hardware. If you decide to extend the vocabulary of an existing card, you need to install, and possibly create or edit the new vocabulary files.

Each vocabulary has three files associated with it:

- vocabname.VBx
- vocabname.TPx
- vocabname.VTx

**vocabname.VBx**

The _base vocabulary_ file which contains the data that is loaded onto the voice card to enable it to recognize and decode the speech. The base vocabulary filename extension indicates which type of voice card it supports:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Card(s) supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>vocabname.VB0</td>
<td>VR/40</td>
</tr>
<tr>
<td>vocabname.VB1</td>
<td>VR/121, VR/81-MC, VR/41-MC</td>
</tr>
<tr>
<td>vocabname.VB2</td>
<td>VRM/40</td>
</tr>
<tr>
<td>vocabname.VB3</td>
<td>VRM/2C</td>
</tr>
<tr>
<td>vocabname.VB4</td>
<td>Antares</td>
</tr>
</tbody>
</table>

**vocabname.TPx**

The _template_ file which defines the alphanumeric character that is to be sent to the application for each word in each vocabulary. This file may be supplied with the base vocabulary file but you may need to construct it yourself. This file can also be edited to customize your speech recognition. An example template file is shown in Figure 4 on page 108. This example file shows the standard DirectTalk/2 configuration file format. The values given must correctly indicate the features of the vocabulary as follows:

- The first line of the file is a comment describing the vocabulary. In the example this is *North American English Standard Digits*

- There is one TYPE=VOCAB statement in the file. This statement contains a number of values and substatements describing the vocabulary:

  **NUMCHAN=**
  Indicates the number of channels the vocabulary and hardware combination supports on a single daughter module or Digital Signal Processor (DSP). In the example, the number of supported channels is four.

  **DEF_CAPS=**
  This indicates the valid default capabilities of the vocabulary. The possible options are:
DISC     Discrete recognition is supported
CONT     Continuous recognition is supported
VSTOP    Voice stop is supported
CUTTHRU  Cutthrough is supported

The setting of this value is optional; an individual subvocabulary definition can override the default settings for the vocabulary. If more than one capability is specified, the set must be enclosed in brackets as in the example (DISC CUTTHRU VSTOP).

SUBVOCAB=
This is a substatement containing the values for a subvocabulary. Because it is a substatement, it must be terminated by a colon (:).

The value indicates which vocabulary is being defined and there must be a WORDS= value, indicating which letter is returned for each word in the subvocabulary (see “Specifying Subvocabulary Words” for a fuller definition. The capability values, as described in DEF_CAPS, can be defined for each subvocabulary. If any capability is defined, none of the defaults are used. The example shows five SUBVOCAB= substatements.

Note: The NUMCHAN capabilities values are not used for non-Antares implementations because these values can be determined from the card type.

Specifying Subvocabulary Words
The WORDS= value indicates the single character returned for each word that the subvocabulary can recognize. The returned characters are case sensitive. Numbers are returned as the number (1 for ONE for example), and lower case letters are returned for letters (a for A for example). Upper case letters are reserved for control words and for error conditions. Although you can modify the standard files, the voice recognition actions, VR_Get_String and VR_Get_Yes_No will not work if changes are made.

The standard control words are the same whichever system language you are using. The English spoken words are shown here:

Y  yes
N  no
H  help
C  cancel
T  stop

The reserved error words are:

U  unrecognized
S  spoken too soon
G  spoken too softly

Notes:
1. S and G are not entered in the template as they are returned from the voice recognition server.
2. U may be entered in the template to indicate an unexpected response.

Some subvocabularies contain synonyms for certain words. The UK English vocabulary for example has ‘zero’, ‘nought’, and ‘oh’. Depending on the
requirements of the application, you may want to return the same or different characters for the three synonyms.

**vocabname.VTx**

A text file which is provided by VCS and describes the content of the subvocabularies held within the base vocabulary file. This file is not used by DirectTalk/2. An example text file is shown in Figure 4.

The vocabulary files should be installed in the main DirectTalk/2 directory.

### Template File

```
+North American English Standard Digits

TYPE=VOCAB
NUMCHAN=4
DEF_CAPS=(DISC CUTTHRU VSTOP)
SUBVOCAB=1 WORDS="YNHCTUUUUUUUUUUU":
SUBVOCAB=2 WORDS="UUUUUUUUUUUUUUUU":
SUBVOCAB=3 WORDS="1234567890zerodotoTUUUU":
SUBVOCAB=4 WORDS="1234567890HCTUU":
SUBVOCAB=5 WORDS="1234YNHUUUUUUUUU":
```

### Text File

<table>
<thead>
<tr>
<th>VOCAB 1: (Regular talk triggering)</th>
<th>VOCAB 4: (Regular talk triggering)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 yes</td>
<td>0 one</td>
</tr>
<tr>
<td>1 no</td>
<td>1 two</td>
</tr>
<tr>
<td>2 help</td>
<td>2 three</td>
</tr>
<tr>
<td>3 cancel</td>
<td>3 four</td>
</tr>
<tr>
<td>4 stop</td>
<td>4 five</td>
</tr>
<tr>
<td>5 six</td>
<td>5 six</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VOCAB 2: (Reserved for future use)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 seven</td>
</tr>
<tr>
<td>7 eight</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VOCAB 3: (Quick talk triggering)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 nine</td>
</tr>
<tr>
<td>9 zero</td>
</tr>
<tr>
<td>10 oh</td>
</tr>
<tr>
<td>11 help</td>
</tr>
<tr>
<td>12 cancel</td>
</tr>
<tr>
<td>13 stop</td>
</tr>
<tr>
<td>5 six</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VOCAB 5: (Regular talk triggering)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 eight</td>
</tr>
<tr>
<td>8 nine</td>
</tr>
<tr>
<td>9 zero</td>
</tr>
<tr>
<td>10 oh</td>
</tr>
<tr>
<td>11 stop</td>
</tr>
<tr>
<td>4 yes</td>
</tr>
<tr>
<td>5 no</td>
</tr>
</tbody>
</table>

*Figure 4. Vocabulary File Formats*

### Application Notes

Voice recognition is not as accurate as DTMF tone input. However, voice input can be expected to yield very high accuracies and, with some forethought and consideration in your applications, voice recognition can be very nearly as accurate as DTMF input.
The most important factor in the success (acceptance by your callers/customers) of the Voice Recognition application is the perception by the callers that the application is user friendly. This means actively verifying each input and giving active responses rather than passive verification. For the purposes of these notes, passive verification means accepting a string of input and only performing verification on the entire string after it has been entered. If an error occurs, the application has very little choice but to ask for the entire string again. This results in much lower throughput and also caller frustration.

When designing an application based on voice recognition, it is preferable to structure the choices and options as a series of Yes/No questions. A caller will intuitively know the right word to answer the question. Also, they will not have to memorize the menu and be tempted to say unrecognizable words such as "what", "repeat", and so on. Additionally, the vocabularies containing the words Yes and No are quite small, which will significantly increase the accuracy of the recognition.

As a rule, multiple choice menus are appropriate for DTMF tone input and Yes/No questions are more appropriate for voice recognition applications. When the voice recognition application is taking input (either a string or menu choices), it should attempt to verify each word as it is input and obtain correction as necessary.

There are six possible errors in voice recognition, as follows:

- Time out: no spoken input detected.
- Unrecognized: utterance could not be matched to a word template.
- Too Soon: caller spoke too soon (before the beep).
- Too Soft: caller spoke too softly.
- Noise: too much background noise. (This is not reported directly by the system.)
- Substitution: word recognized is not the word spoken by the caller.

The first five errors can be dealt with immediately upon detection of each utterance. The last error, substitution, can only be dealt with by asking the caller to verify the word.

A voice recognition application will be more effective if it performs in the following way. For illustration, the example will be the input of a 9-digit number:

1. The application explains what is expected from the caller:
   
   "Please enter your 9-digit identification number by saying one digit after each tone."

2. For each digit, the application performs the following:
   
   a. Prompt with the tone (this is done by the Get_Voice_Resp action).
   
   b. Handle the first five errors listed above as follows:
      
      - Time out

      If expecting a fixed length string and the timeout occurs before the last digit, tell the caller the number of digits entered so far and prompt them to continue. Go back to step 2a.
If you are expecting a variable length string, use “Stop” as the termination word. If “Stop” has not yet been received, treat the timeout as follows:

“You have entered 4 digits so far. Please continue after the tone.”

Alternatively, you can play back the string so far and prompt the caller to either continue or, if the playback is incorrect, to start again:

“You have entered 1-2-3-5. Please continue the string after the tone or, if this is incorrect, say Stop after the tone.”

If the caller says “Stop”, reset the string and start with step 1 on page 109 above.

- **Unrecognized**
  
  Tell the caller the word was not understood and ask them to repeat the word at the tone:
  
  “I did not understand that digit. Please repeat the digit after the tone and speak more clearly.”

- **Too Soon**
  
  Tell the caller that they spoke too soon and to wait for the tone and then repeat the digit. Go back to step 2a on page 109:
  
  “You spoke too soon. Remember to wait for the tone before speaking. Please repeat the digit after the tone.”

- **Too Soft**
  
  Tell the caller that they spoke too softly and to wait for the tone and then repeat the digit. Go back to step 2a on page 109:
  
  “You spoke too softly. Please repeat the digit after the tone and speak more clearly.”

- **Repeated errors**
  
  If there are repeated errors, a noisy line or other problem can be assumed. If this occurs, the application should transfer the caller to be handled by a live person. Tell the caller that there is a problem and that they are being transferred to an operator or agent.
  
  “Sorry. There appears to be a problem. I am transferring you to an agent to handle your request. Please hold.”

- **c.** When the last digit has been received (either the last digit of a fixed string or the word “Stop”), the digit collection is complete and the application should continue with a full string verification.

  This process should handle no more than three of any of the above errors. If the fourth Time Out, Unrecognized, Too Soon, or Too Soft error occurs, transfer the caller to a live person as in the Noise error condition above.

3. Play back the entire string and ask the caller to verify. Any substitution errors will be caught in this verification:

“You entered 1-2-3-4-5-6-7-8-9. If this is correct say Yes. If it is not correct, say No.”

4. If the string is not correct, ask the caller to repeat the string and repeat the processing described in step 2.
"Please repeat the number. Remember to wait for the tone before speaking."

The smaller the string length, the more likely it is to be recognized without errors. If the string is long the application could attempt to recognize parts. Credit card numbers, which are written in groups, could be addressed this way.

If a caller has attempted to enter the string twice and has not been successful, they should be transferred to a live person:

"Sorry. There appears to be a problem. I am transferring you to an agent to handle your request. Please hold."

If the string is entered successfully, continue processing the call normally.

**Voice Cutthrough**

Voice cutthrough allows a caller to interrupt a voice prompt with an utterance intended for voice recognition. This capability is similar to the “fastpath” capability of DTMF tone input. The primary benefit of voice cutthrough is to increase the speed with which a caller, especially an experienced one, can interact with a voice application.

DirectTalk/2 offers two forms of voice cutthrough:

**Voice detection**

Voice detection (commonly referred to as “grunt detection”) allows a caller to interrupt a prompt by uttering any sound. There is no attempt to recognize the utterance as a word; it is simply a means by which the caller can interrupt a prompt. Any voice recognition follows the interruption.

**Word recognition**

Word recognition allows a caller to interrupt a prompt by uttering a word from the expected set of answers. The system performs voice recognition processing on the utterance and returns the word match.

Voice cutthrough is available with VRM/40 hardware modules on the VRP base board, and on the Antares platform. A special cutthrough vocabulary is included as part of the appropriate DirectTalk/2 language package.

During execution of a voice program, voice cutthrough is only available when a voice is being played on the phone line. As such, it is enabled by parameters on the Play_Module action or the Play_Voice action.

In the case of voice detection, the voice program can proceed with voice recognition processing in the same way as it would without voice cutthrough. That is, the Play_Module or Play_Voice actions are followed by one of the voice recognition actions:

- Get_Voice_Cont
- Get_Voice_Resp
- VR_Get_String
- VR_Get_Yes_No

and any further speech is captured and analyzed.

In the case of word recognition, the interrupting utterance is returned by the Play_Module or Play_Voice action in the last_voice Resp variable. This first word must be stored in another variable before continuing with the rest of the input,
because the other voice recognition actions will clear and then use the
last_voice_resp variable.

### Cutthrough Voice Recognition Accuracy

The accuracy of Cutthrough Voice Recognition is affected by:

1. The quality of the voice prompts
2. The caller's environment

By following the guidelines that follow, accuracy should be very similar to "normal" discrete voice recognition, because the recognition algorithms (which are used once the prompt has stopped) are the same.

The quality of the prompt is important because the voice recognition (VR) software has to "cancel out" the prompt from the received signal to leave any received noise. A higher quality prompt gives better cancellation. To get the best quality prompts:

- If possible, run the system with 64k bits per second voice recording and playback.
- Record the prompts with the VP parameter Automatic Gain Control set to OFF.

### Dealing With Background Noise

The caller's environment may have background noise or telephone line noise. The VR software needs to determine when it is the caller speaking and when it is background noise which it should ignore. There are two parameters that affect Cutthrough: Cutthrough Gain and Cutthrough Threshold. These parameters are described here, but they should only be modified in exceptional circumstances.

**Cutthrough Gain** controls the echo canceller. As this value is decreased, cutthrough sensitivity is decreased. This value should be set to 400 and **not** changed. This value is optimized for the Dialogic analog front-end cards. It has been set to cover the widest possible range of segments that can be played from the Dialogic cards. To change it may improve the sensitivity on a given segment, but make it worse for another segment which has different characteristics. So, do not change the Cutthrough Gain parameter.

**Cutthrough Threshold** is a measure of the energy on the line. You can change this parameter.

The default value of Cutthrough Threshold is 10000. As this value is increased, the sensitivity is decreased. One way to think of it is that the higher the number, the louder the caller must speak to interrupt the module. The following steps give help on how to choose a suitable value for this parameter for a given system and vocabulary:

1. With the Threshold at 10000, connect to the system and make the line as quiet as possible by covering the mouthpiece. Listen to the module to see if it completes all the way through. This will tell you whether the actual segments are contributing to the problem.

2. In any case, raise the Threshold to 20000 (this is very nearly the maximum effective value). Covering the mouthpiece, play the prompt through to completion again. It should complete.
3. Uncover the mouthpiece and play the prompt again, just listening with normal ambient noise. And then again, only this time making “incidental noises”. An “incidental noise” could be a sigh or a scrape against a beard, or any noise which does not involve the vocal cords (that is, not noises such as clearing your throat). Again, the prompt should play through to completion.

At this point, the Cutthrough is set up to ignore the incidental noise. It is now necessary to determine its responsiveness to normal voice.

Make several cutthrough attempts with normal voice and ever increasing voice loudness to determine at what level cutthrough occurs. At this point, accuracy is not a concern. If ten different people can do this part of the test, there will be a better basis.

If the only way to cause a cutthrough is to shout, then 20000 is too high. Set the value to 15000 and repeat this step. Adjustments to this parameter are meaningful in steps of 1000 or 2000.

The optimum value will be when the incidental noise does not cause cutthrough but the normal voice does. Also, it is reasonable that a caller may have to raise his or her voice somewhat to cause a cutthrough to occur. This is exactly the same as trying to interrupt a person during normal conversation.

Note that if segments contain music, this will be difficult for the cutthrough to handle due to the wider dynamics of the music.

---

**Discrete Alphanumeric Recognition**

DirectTalk/2 greatly enhances the usefulness of voice recognition by adding alphabetic recognition. This is primarily useful in processing catalogs, car license and car registration numbers, postal codes (such as those used in Canada and much of Europe), and order numbers.

**Note:** Alphanumeric recognition requires specially enabled hardware. See the *IBM CallPath DirectTalk/2 Installation Guide*.

Alphanumeric voice recognition increases the size of the set of words that can be recognized, to include letters of the alphabet. However, with the additional recognition capability comes further considerations. It is generally not recommended that alphanumeric recognition be used in a raw recognition application; that is, in an application where every utterance is taken discretely in the way that numeric voice recognition is usually done. This is mainly because, with an alphanumeric vocabulary, there are a significant number of letters that can be confused with each other. For example, D, T, and 3 have a very similar sound. Again, B, C, E, G, P, V, and Z also rhyme and will very likely be confused with each other.

For successful alphanumeric voice recognition, it is recommended that you use a more robust scheme. One such scheme is a string-based recognition algorithm developed by Voice Control Systems (VCS). The algorithm is detailed in a note called *How To Implement an Alphanumeric Voice Recognition Application*, and is available from Voice Control Systems or from Dialogic Corporation. This algorithm can be used when there is a known set of valid responses, catalog identifiers for example.
In summary, this algorithm involves setting up a database of the set of expected string inputs and then scoring each of the strings based upon a caller's input. In effect, rather than scoring each individual word, as is done with numeric recognition, the entire string is scored.

The process works as follows:

- As each letter is spoken, the score for all the words in the subvocabulary are stored.
- When all utterances are in, a score for each of the valid strings is generated by adding together the scores for each of the characters in the string.
- The string with the lowest score is repeated to the caller for confirmation.
- If the string is not correct, the second best string can be tried.

The strength of the algorithm is that, for a given input string of utterances, the scores of the letters that match the target correctly will be far lower than the scores of the letters that don’t match the target; that is, the score differences of correct matches will be significantly higher than the score differences of incorrect matches.

**Note:** The accuracy of string-based recognition may be compromised in a database of strings where a large number of the strings differ only in one position. This algorithm does require that the input strings are a well-defined set.

The DirectTalk/2 voice recognition APIs provide a function to return the scores of all words in a subvocabulary for a given utterance. The generation of string scores and the determination of the best strings must be implemented by a custom user action. The detailed design and implementation will depend on the characteristics of the input strings and those of the string databases.

**Continuous Voice Recognition**

Continuous digit recognition allows a caller to speak a string of selected words at normal or close to normal speech rate. This improves most callers’ comfort level with the voice recognition technology, because it allows them to speak more naturally. Continuous voice recognition does not prompt for each utterance, but instead prompts for a string of utterances.

Continuous voice recognition is designed to deal with a specific domain of inputs. It is not to be confused with what is generally known as “word spotting”. A caller cannot speak entire phrases and expect that the appropriate answer will be sifted out from among the many undefined words. The caller must still provide utterances from within a defined set of allowed input; these can, however, be spoken at normal rates. For example, the caller cannot say “two thousand four hundred and seventeen,” but must say “two four one seven.”

Continuous voice recognition is supported by the VRP and Antares hardware. On the VRP, you need to install the VRM/2C modules, which will each support two lines of continuous voice recognition. On Antares, you must install a Dongle which enables the required number of continuous voice recognition channels. Additionally, continuous voice recognition requires the use of a special vocabulary which in included as part of the appropriate DirectTalk/2 language package.

Continuous voice recognition is enabled by the use of the Get_Voice_Cont action. This action can be used wherever other voice recognition actions would be used.
The action will return the two highest-rated strings. The highest-rated string should be played back to the caller for confirmation. If the caller indicates that the string is incorrect, the voice program can either play the second string or else simply prompt again for input. Parameters on the Get_Voice_Cont action indicate the minimum and maximum expected string length. The action determines the maximum response time from the maximum string length. Best results are obtained when the string length is low and the difference between the minimum and maximum is no more than two digits.

A voice program should monitor the success of a given caller with continuous voice recognition, and be prepared to offer the use of discrete voice recognition. This applies particularly when a caller cannot properly input a string after two attempts. The advantage of discrete voice recognition is that it forces callers to pace their responses. This will most likely be necessary for callers new to the system and inexperienced in the use of voice recognition. Offering discrete voice recognition as an option will avoid new users becoming frustrated with the technology. Experienced users will become comfortable with continuous voice recognition.

Voice Recognition Testing

At the earliest opportunity (and certainly before the system goes into production), you should test your Voice Recognition configuration and accuracy.

Testing For Accuracy

Voice Recognition accuracy testing requires a large number of samples in order to be useful. That is, a valid accuracy test must be performed with a statistically significant group. This group should comprise at least 25 callers. Each caller should say each word at least 10 times. Additionally, the callers should be representative of the expected end users. Each caller should be given a list of the words to be spoken and asked to record the result. The testing application should not prompt for a specific word; rather, it should prompt generically for the next word and the caller should read it from the list. These tally sheets should be used to calculate the accuracy. Caller errors (such as speaking too soon or too softly) should not be counted in the accuracy measurement because the voice recognition technology did not have a chance to recognize the utterance.

An alternative method for recording Voice Recognition scores is to use the Telephony Server Voice Recognition Scores trace. The advantage of this method is that it allows you to store all the scores in an ASCII file for later reference. To start the Voice Recognition scores trace, add the statement LOG_SCORES to the Voice Recognition board definition in the Telephony Server configuration file, VSTS.CFG. The trace is written to an ASCII file called VRANTSCR.OUT if you are using Antares, and to VRSCORES.OUT if you are using other voice recognition cards. These files are in the DirectTalk/2 main directory (usually \DTALK).

Testing The Voice Recognition Configuration

To test the functional configuration of the Voice Recognition Feature, you can use one caller and ten utterances per word to get a fair indication (although this is not a good basis for accuracy testing: see “Testing For Accuracy”).

The call should be performed in a situation which resembles as closely as possible an actual caller line (external call through the network versus internal to the PBX,
and so on). The test will give the three top choices for each utterance and the scores associated with the match. You should look at the top two choices. You should note the range of the top choice of all the words and the difference between the first and second choice. You will want to determine the range of scores which cover the first choices for all of the utterances and the range of differences between the first and second choices.

The range of scores will give an indication for the values of the Acceptance Threshold (upper limit). When looking at the range, it can be treated statistically and the highest 5% can be ignored. The Acceptance Threshold should be at least greater than the high end of the range plus 300. The differences will give an indication for the value of the Minimum Difference. If you are attempting to be as discriminating as possible, the lowest 5% can be ignored.

Ideally, this test should be performed for each sub-vocabulary as the range of scores for a given word in different sub-vocabularies may be slightly different. However, if the test is run against the largest sub-vocabulary, the resulting scores should be fairly representative. The parameter values should be chosen so that the scores from all sub-vocabularies are included.

Voice Recognition Sample Application

The RMENU sample application contains a sample voice recognition application. This sample application allocates separate telephone lines for voice response, and tests the responses for all valid values using a series of Compare_Chars actions. Alternatively, the Branch action could be used.

Voice Recognition Actions

The actions provided by the Voice Recognition feature are:

- **Get_Voice_Cont**: Allows your application to recognize one or more words spoken by a caller with a given vocabulary set and return the words to the application. This action allows a caller to speak naturally and does not prompt between words in a string.

- **Get_Voice_Resp**: Allows your application to recognize voice responses from a caller. This action is similar to the Get_Tone_String action. The caller must pause after each utterance.

- **VR_Get_String**: Enables the application to recognize one or more single words spoken by the caller, and verify that the recognized words are the ones that the caller actually said. It manages the interaction and provides context-sensitive help where needed. For example, if the user makes a mistake, it will provide information specific to the particular mistake that was made. When it returns, the response data has been verified by the caller.

- **VR_Get_Yes_No**: Allows the application to initiate a dialog with the caller to receive a response of “yes” or “no”. It provides the same type of interaction and context-sensitive help as VR_Get_String. If the response from the caller is
ambiguous, it attempts to determine what the correct response should be.

**VR_Get_String and VR_Get_Yes_No**

The voice recognition actions VR_Get_String and VR_Get_Yes_No (described in the previous section) require that you add additional voice segments and voice logic modules to your system voice segment and system voice logic module databases. To make this process easier, the segment text and logic modules have been provided on the Voice Recognition Feature disk in the form of a DirectTalk/2 application. You need to copy the segments and voice logic modules from these applications to the SYSTM application and then record the segments. Figure 5 on page 121 shows a simplified flow of the VR_Get_String action indicating where the various voice segments are used.

Some of the voice segments are used to play tones, and cannot readily be reproduced. Therefore, these segments have been recorded for you and are included in these applications. Before copying these segments to your SYSTM application, you must know which voice sampling rate and which language you use. The tone segments have been provided for all of the possible voice sampling rates. You need to copy the tone segments from the application that corresponds to the sampling rate you use on your system, as indicated in Table 11.

<table>
<thead>
<tr>
<th>Rate</th>
<th>Application Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 Kbps</td>
<td>EVR24</td>
</tr>
<tr>
<td>32 Kbps</td>
<td>EVR32</td>
</tr>
<tr>
<td>48 Kbps</td>
<td>EVR48</td>
</tr>
<tr>
<td>64 Kbps</td>
<td>EVR64</td>
</tr>
</tbody>
</table>

**Note:** These applications were recorded into databases using the U.S. English language code. Before copying the segments rename the files EVRxx.SGE and EVRxx.PRE so that the language character matches the language which you are using for your applications.

(For further details, see “Copying Voice Segments” on page 71 and “Copying Voice Logic Modules” on page 66.) The prerecorded voice segments are listed in Table 12.

<table>
<thead>
<tr>
<th>Voice Segment</th>
<th>Prerecorded tone</th>
</tr>
</thead>
<tbody>
<tr>
<td>vr_desc_pair</td>
<td>Descending tone pair used in VR_Get_String</td>
</tr>
<tr>
<td>vr_buzzer</td>
<td>Buzzer used for VR_Get_String</td>
</tr>
<tr>
<td>vr_asc_pair</td>
<td>Ascending tone pair used in VR_Get_String</td>
</tr>
</tbody>
</table>

Table 13 on page 118 lists the voice segments for the EVRxx applications which are provided as text only. These segments need to be recorded.
<table>
<thead>
<tr>
<th>Segment</th>
<th>U.S. English Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>vr_beginning</td>
<td>Would you like to hear what you said from the beginning?</td>
</tr>
<tr>
<td>vr_buzz_means</td>
<td>The buzzer means “repeat your answer more clearly”.</td>
</tr>
<tr>
<td>vr_choices</td>
<td>Do you want to hear your choices again?</td>
</tr>
<tr>
<td>vr_confusion</td>
<td>Sorry for the confusion ...</td>
</tr>
<tr>
<td>vr_correct</td>
<td>Are the characters correct?</td>
</tr>
<tr>
<td>vr_did_say</td>
<td>Did you say ...</td>
</tr>
<tr>
<td>vr_digit</td>
<td>... character</td>
</tr>
<tr>
<td>vr_digits</td>
<td>... characters</td>
</tr>
<tr>
<td>vr_example</td>
<td>Eight &lt;BEEP&gt; zero &lt;BEEP&gt; zero &lt;BEEP&gt;.</td>
</tr>
<tr>
<td>vr_hear_buzz</td>
<td>Whenever you hear this buzzer ...</td>
</tr>
<tr>
<td>vr_hello</td>
<td>Hello ...are you there?</td>
</tr>
<tr>
<td>vr_if_problem</td>
<td>If there’s a problem ...</td>
</tr>
<tr>
<td>vr_is_first</td>
<td>Is the first character ...</td>
</tr>
<tr>
<td>vr_is_it</td>
<td>Is it ...</td>
</tr>
<tr>
<td>vr_is_next</td>
<td>Is the next character ...</td>
</tr>
<tr>
<td>vr_like_this</td>
<td>It should sound like this ...</td>
</tr>
<tr>
<td>vr_listen</td>
<td>Listen for the tone after each character.</td>
</tr>
<tr>
<td>vr_louder</td>
<td>Please speak a little louder.</td>
</tr>
<tr>
<td>vr_may_stop</td>
<td>You may say the word “stop” at any time.</td>
</tr>
<tr>
<td>vr_mistake</td>
<td>If you make a mistake ...</td>
</tr>
<tr>
<td>vr_more_choice</td>
<td>Here are more choices ...</td>
</tr>
<tr>
<td>vr_once_again</td>
<td>Now once again ...</td>
</tr>
<tr>
<td>vr_one_digit</td>
<td>You should say only one character after each beep.</td>
</tr>
<tr>
<td>vr_pause</td>
<td>Simply pause just a moment ...</td>
</tr>
<tr>
<td>vr_problem</td>
<td>There seems to be some problem.</td>
</tr>
<tr>
<td>vr_remember</td>
<td>Please remember ...</td>
</tr>
<tr>
<td>vr_repeat</td>
<td>... and clearly repeat what you just said.</td>
</tr>
<tr>
<td>vr_repeat_digit</td>
<td>Please repeat that character.</td>
</tr>
<tr>
<td>vr_repetition</td>
<td>Listen to the repetition.</td>
</tr>
<tr>
<td>vr_rpt_choice</td>
<td>As soon as you hear the choice you want, please repeat it.</td>
</tr>
<tr>
<td>vr_say_number</td>
<td>Would you like to say your characters again?</td>
</tr>
<tr>
<td>vr_say_stop</td>
<td>Say the word “stop”.</td>
</tr>
<tr>
<td>vr_start_over</td>
<td>Let’s start over.</td>
</tr>
<tr>
<td>vr_that_only</td>
<td>That was only ...</td>
</tr>
<tr>
<td>vr_that_was</td>
<td>That was ...</td>
</tr>
<tr>
<td>vr_this_time</td>
<td>This time ...</td>
</tr>
<tr>
<td>vr_try</td>
<td>Let’s try once more.</td>
</tr>
<tr>
<td>vr_wait_tone</td>
<td>Remember to wait for the tone before speaking.</td>
</tr>
</tbody>
</table>
The following Voice Logic Modules are provided in the ERVxx applications. Copy these modules to the SYSTM application.

$$vr\_did\_say–Play 'Did you say $$vr\_digit'$$

```
PLAY SEG 'vr\_did\_say'
PLAY STR $$vr\_digit
```

+----------------------------------------------------------+
| Did you say ... |
+----------------------------------------------------------+

$$vr\_is\_first–Play 'Is the first digit $$vr\_digit'$$

```
PLAY SEG 'vr\_is\_first'
PLAY STR $$vr\_digit
```

+----------------------------------------------------------+
| Is the first character ... |
+----------------------------------------------------------+

$$vr\_is\_it–Play 'Is it $$vr\_digit'$$

```
PLAY SEG 'vr\_is\_it'
PLAY STR $$vr\_digit
```

+----------------------------------------------------------+
| Is it ... |
+----------------------------------------------------------+

$$vr\_is\_next–Play 'Is the next digit $$vr\_digit'$$

```
PLAY SEG 'vr\_is\_next'
PLAY STR $$vr\_digit
```

+----------------------------------------------------------+
| Is the next character ... |
+----------------------------------------------------------+

$$vr\_play\_dcf11–Play 'That was $$vr\_digit\_digit'$$ ($$vr\_digit$$ equals 1)

```
PLAY SEG 'vr\_that\_was'
```

+----------------------------------------------------------+
| That was ... |
+----------------------------------------------------------+

```
PLAY NUM $$vr\_digit
```

+----------------------------------------------------------+
| | character |
+----------------------------------------------------------+
$$vr\_play\_dcf12$$—Play 'That was \(<$$vr\_digit>\) digits' (\(<$$vr\_digit>\) greater than 1)

```
PLAY SEG 'vr\_that\_was'
|That was ... |
PLAY NUM $$vr\_digit$$
PLAY SEG 'vr\_digits'
```

$$vr\_play\_dcf21$$—Play 'That was only \(<$$vr\_digit>\) digit.' (\(<$$vr\_digit>\) equal to 1)

```
PLAY SEG 'vr\_that\_only'
|That was only ... |
PLAY NUM $$vr\_digit$$
PLAY SEG 'vr\_digit'
```

$$vr\_play\_dcf22$$—Play 'That was only \(<$$vr\_digit>\) digits.' (\(<$$vr\_digit>\) greater than 1)

```
PLAY SEG 'vr\_that\_only'
|That was only ... |
PLAY NUM $$vr\_digit$$
PLAY SEG 'vr\_digits'
```

$$vr\_play\_seg$$—Desc: Play the segment indicated by $$vr\_segment$$ variable

```
PLAY SEG $$vr\_segment$$
```

**Note:** $$vr\_segment$$ is a variable which the application changes as required to play a specified segment. In this way, the need for multiple Voice Logic Modules is avoided.

$$vr\_say\_digit$$—Play \(<$$vr\_digit>\)

```
PLAY STR $$vr\_digit$$
```

120 DirectTalk/2 Application Development User's Guide
Figure 5. Process Flow for VR_Get_String
Chapter 15. Communications

The DirectTalk/2 Communications Feature consists of hardware and software which allows you to communicate with mainframe computers using a variety of communication types and terminal emulators.

Emulators Supported

The DirectTalk/2 Communications feature supports 3270, 5250, and ASCII terminal emulation. The combinations of terminal types and Communications products that you can use are listed in Table 14.

Table 14. DirectTalk/2 Terminal Emulation Support

<table>
<thead>
<tr>
<th>Emulation Monitor</th>
<th>Emulation Terminal Type</th>
<th>Communications Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS3270EH</td>
<td>IBM 3270</td>
<td>Communications Manager/2 (3270 feature) or Personal Communications/3270 for OS/2 or Personal Communications AS400 and 3270 for OS/2</td>
</tr>
<tr>
<td>VS5250EH</td>
<td>IBM 5250</td>
<td>Communications Manager/2 (5250 feature) or Personal Communications AS400 for OS/2 or Personal Communications AS400 and 3270 for OS/2</td>
</tr>
<tr>
<td>VS3270LU</td>
<td>IBM 3270</td>
<td>Communications Manager/2 (LUA feature) Communications Server/2 (LUA feature)</td>
</tr>
<tr>
<td>VS3270MX</td>
<td>IBM 3270</td>
<td>Supported directly by DirectTalk/2 when using Portmaster Adapter/A or Multiport Model 2 Realtime Interface Coprocessor Adapters (ARTIC)</td>
</tr>
<tr>
<td>VSASCII</td>
<td>DEC VT100</td>
<td>TCP/IP for OS/2</td>
</tr>
<tr>
<td></td>
<td>DEC VT220</td>
<td>TCP/IP for OS/2</td>
</tr>
<tr>
<td></td>
<td>IBM 3151</td>
<td>TCP/IP for OS/2</td>
</tr>
</tbody>
</table>

The Communications actions provided with DirectTalk/2 perform the same functions on all the supported emulation terminal types, unless otherwise stated in the descriptions of the individual actions.

Emulation Session Groups

DirectTalk/2 allows terminal emulator sessions to be configured into groups with all the sessions in a group sharing the same session name.

Grouping can be useful if, for instance, you have a number of emulation sessions connected to a single host computer. If you group these sessions together, an application does not have to be aware of different names for each of the emulator sessions.

An emulator session group can contain one or many sessions and the groups are specified on the communications panels of the DirectTalk/2 Setup program. If you do not specify emulator session groups, all your sessions belong to a default group. The name of the default group is the name of your emulation monitor server and these names are stored in the following system variables:

- **CM32_server** for VS3270EH
- **5250_server** for VS5250EH
LUA32_server for VS3270LU
3270_server for VS3270MX
Ascii_server for VSASCII

You specify the type of emulator you want to use by specifying the session group name in the Connect_Screen action.

Terminal Emulation Applications

Before beginning to implement a terminal emulation application, you should understand the operation of the host application and its use of the terminal. You need the following information about the displays used in the host application:

- The location (row and column) of text that can uniquely identify each host screen
- The screen flow for the normal processing case
- Which keystrokes are required to get back to the initial screen from any other application screen
- The screen flow required to log on to the application

In general, ASCII terminal emulation presents a tty-type interface. There is no concept of a static screen because the text scrolls up the screen (in a similar way to the text in an OS/2 window). Actions that retrieve text, by specifying a row and column, retrieve the text from the specified area of the emulated screen at the time the call is made. Similarly, the Send_Keys_to_Scr action inserts text at the current position. You need to design your applications accordingly.

Operator Information Area

The Operator Information Area (OIA) is the bottom line of some screen types. DirectTalk/2 provides access to this part of the screen for 3270 and 5250 terminal types through the Refresh_OIA action. OIAs in VSASCII terminal emulations are not supported by DirectTalk/2.

The Refresh_OIA action has two associated system variables and the use of these variables depends on the emulator type you are using:

**OIA_data**
This variable can only be used for VS3270EH and VS5250EH emulation sessions.

OIA_data stores an ASCII representation of 22 bytes of binary OIA data as described in *IBM Communications Manager/2 EHLLAPI Programming Reference* (SC31-6163).

**OIA_data_ascii**
This variable can be used by all the supported 3270 and 5250 terminal types.
The data is stored as a string of 80 ASCII characters as displayed on the screen. See “Refresh_OIA” on page 250 for more details of the data format.
Multiple Host Access

The Communications feature also enables you to access up to four host sessions without logging off one session before logging on to another. This not only increases the efficiency of your voice applications, but also increases their effectiveness. For example, you can program a DirectTalk/2 application to first access an IBM 3270 Information Display System terminal emulation host session and then a 5250 terminal emulation host session, using the Switch_Screen action. For a description of this action, see “Switch_Screen” on page 275.

Programming Considerations

The basic structure of a DirectTalk/2 application using terminal emulation is:

1. Log on to the host application with the Connect_Screen action and go to the base screen for the application. Since the process of logging on could be the same for different host programs, you may want to define this process as a subapplication that you can invoke by more than one voice application. Perform a logon for each host session (up to four) that is required.

2. Wait for a call.

3. Answer the call and guide the caller through the transaction. Use the Switch_Screen action to change to the host session that contains the data to satisfy the caller’s request. The caller hangs up at the end of a transaction.

4. Send the keystrokes that the application requires to return to the base screen, to the host application and loop back to wait for another call.

You should keep in mind several considerations as you design a DirectTalk/2 application that uses terminal emulation. You will need to:

- Ensure that the host interaction has completed successfully. Issue Wait_Scr_Update after a Send_Keys_to_Scr action, and before retrieving screen data or issuing another Send_Keys_to_Scr.
- Provide the logic to handle situations when the host application becomes unavailable.
- Provide the logic to handle an unexpected host screen response.
- Provide the logic to handle a caller hanging up in the middle of a transaction.

You can determine if the host computer has gone down in several different ways. These are:

- If DirectTalk/2 cannot establish a physical connection to the host, the initial Connect_Screen action fails. If the action loses the connection during processing, processing should continue with the Session Disconnected Return Code.
- If the action experiences a communications error with the host, processing should continue with the Session Error return code.
- If the Wait_Scr_Update action returns a time out, this can indicate that the host computer is unavailable or that an unexpected screen was displayed.

If the host system goes down during call processing, you will need to play a message to the caller to hang up and try again later. The program should try to
reestablish a connection with the host and inform the system administrator of the
problem with the host connection. If this retry processing fails, you must specify
that subsequent calls need to be answered with a message that the application is
unavailable. After each call, you should specify that the application tried to
establish a host connection again. After reconnecting to the host computer, the
application can log on and resume normal processing.

If the action encounters an unexpected host screen, this can indicate an application
error or that you did not identify a screen when designing the application. If the
Wait_Scr_Update action returns a time out (the screen the action expected was not
displayed within the specified time period), you must identify the screen to the
application. To identify the screen, you can use a series of Search_Screen actions
until the application matches the screen with a known screen identifier.

When the application determines which screen is displayed, it must recover in
some way. The recovery can be playing an error message and returning to the
base screen, or continuing with application processing. In either case, you should
specify that the action logs a message to note the occurrence and indicate which
screen was displayed.

If the caller hangs up in the middle of a transaction, you must specify that the
application returns to the base screen before waiting for another telephone call. In
many applications, you can accomplish this by sending a PF key to the application.
For more complex host applications, this recovery processing may be more difficult.
You may need to set up a scheme where your voice application identifies the
current screen, and issues the command required to go back one screen. The
application would then repeat this until it reaches the base screen.

### Debugging Communications Applications

The host emulation monitor that forms part of the VAD debug screen, is displayed
in either color or monochrome, depending on the emulator type:

- **Color display**
  - 3270 with ARTIC adapters

- **Monochrome display**
  - 3270 with Personal Communications, OS/2 Communications Manager/2
    and Communications Server/2
  - ASCII

### Communications Actions

The actions that provide an interface with a host application are:

- **Connect_Screen**
  - Connect to the host application.

- **Disconnect_Scr**
  - Disconnect from the host application.

- **Get_Screen_Data**
  - Get information from the simulated screen to use in a DirectTalk/2
    variable.
Refresh_OIA

Obtain the information area data from the emulation screen and store it in the IOA_data_ascii and IOA_data system variables.

Search_Screen

Perform a single search each time you specify the action.

Send_Keys_to_Scr

Send information from a caller or program through the simulated display keyboard to the host application.

Switch_Screen

Change the active host session.

Wait_Scr_Update

Perform one search per second for the number of seconds you specify.
Chapter 16. Text-to-Speech

You can use the optional DirectTalk/2 Speech Synthesis Feature to provide more flexible spoken output capabilities than those provided by prerecorded voice segments. The Text-to-Speech (TTS) feature synthesizes speech from text. Speech Synthesis is available in English, Dutch, German, French, Italian, and Spanish.

**Note:** You must install either the BeSTspeech** (BeST) Text-to-Speech software and hardware, or the Lernout and Hauspie (L&H) Text-to-Speech software and the Dialogic Antares card to use this option.

The following are examples of the uses to which you can put the Text-to-Speech feature:

- Read electronic mail to a caller.
- Read names and addresses of required calls to a service engineer.
- Read the description of a part ordered by catalog number to confirm that it is the required item.
- Provide an information service for rapidly-changing information, such as road conditions, weather, or stock market prices.

In these situations, it would not be feasible to record all the text segments required to meet all the different cases. Text-to-Speech provides the solution to this problem.

The Text-to-Speech feature analyzes the text it is playing to determine how it is to be spoken. It can:

- Change the voice tone depending on the position of a word in a sentence
- Recognize abbreviations and expand them to the full word
- Recognize numbers and speak them appropriately

It is therefore important to insert correct pronunciation in any text spoken using Text-to-Speech. It is also possible to add ‘reset codes’ to the text to more fully control the way the text is spoken.

The actions provided by the Text-to-Speech feature are:

**Play_Text**
To synthesize voice from records that are in a DirectTalk/2 database

**Play_Text_String**
To synthesize voice from data contained in a DirectTalk/2 variable

In addition, a voice logic module can include PLAY statements with either of the two types:

**TXT**
To play a text segment

**TSR**
To play a text string
Creating Text Segments

The Text segments and strings that are to be played by the Text-to-Speech feature are created and modified using the Text Segment Editor which is provided as part of the Voice Application Developer. With the Text Segment Editor you can create, modify, delete, copy, or print text segments. You can also import text that has been written with another editor.

The Text Segment Editor also provides the ability to play the text segments. This allows you not only to check how the segments created will be played, but also to experiment with the reset codes to get the best rendering of the text that will be spoken by the application.

**Note:** You can use the Text Segment Editor to create text segments even if you do not have the Text-to-Speech feature installed, but it must be installed if you want to listen to or play the segments.

Using the Text Segment Editor

![Text Segment Editor screenshot]

To start the Text Segment Editor, select **Text segment editor** from the **Editors** pull-down in the **Voice Application Developer** window.
1. Select **Add or Change** from the **Edit** pull-down in the **Text Segment Editor** window, to display the **Add or Change Text Segment** window.

2. Type the name of the text segment you want to add or change, or press `<F2>` to select from a list of the existing text segments for the current voice application.

DirectTalk/2 displays the Add or Change Text Segment window.
The text for the segment may be entered or modified in this window. While entering or modifying the text you can use the following special keys:

**Key**  | **Use**
---|---
**Ins**  | Toggle between insert and replace mode
**End**  | Move the cursor to the end of the line
**Home**  | Move the cursor to the start of the line
**Alt-S**  | Split the line at the cursor position
**Alt-J**  | Join the next line to the current line
**Alt-L**  | Mark the current line, and all lines between the current line and any other marked line
**Alt-U**  | Unmark any marked area
**Alt-D**  | Delete any marked lines
**Alt-C**  | Copy any marked line to immediately after the current line
**Alt-M**  | Move any marked line to immediately after the current line
**Alt-Del**  | Delete current line
**Alt-Ins**  | Insert line after the current line
**Alt-End**  | Delete from the cursor position to the end of the line

If you want to enter some Text-to-Speech control codes (see “Text-to-Speech Controls” on page 137), press <F5> if you are using L&H TTS, or <F6> if you are using BeST Speech TTS, to display the appropriate Control Codes window.

| BeST Text-to-Speech Control Codes |
|---|---|---|---|
| Set default values | **ON** |
| Alpha literal | **ON** | **OFF** |
| Digit literal | **ON** | **OFF** |
| Prosodic punctuation literal | **ON** | **OFF** |
| Arithmetic pronunciation | **ON** | **OFF** |
| Full number | **ON** | **OFF** |
| Stop time pronunciation | **ON** | **OFF** |
| Stop abbreviation pronunciation | **ON** | **OFF** |
| Stop capital letter pronunciation | **ON** | **OFF** |
| Phoneme reading mode | **ON** | **OFF** |
| Alternate pronunciation | **ON** |

Set speaking rate
Set speech loudness
Set acronym pronunciation length...
Special punctuation marks for changing prosody...

F1=Help  F12=Cancel
In these windows you can set modes on and off and select specific controls. Some of the options lead to further windows and selections. If you need assistance with making your selections press <F1> to display the online help associated with a window or choice.

**Note:** Control codes can also be entered and edited without using the menus. This is particularly useful if the control you want is not included in the lists but you know the characters it uses, you can enter it directly into the text.

If you want to move, copy, or delete one or more lines of text, press <F7> to display a menu of block operations that you can use.

You can also split and join lines using the options provided when you press <F8>, and search for specific text by pressing <F9>.

3. When you have completed your text, save it by pressing <F2> or <F4>.

### Listening to Your Text Segment

Once you have written a text segment, you can listen to it by pressing <F11> when the text is displayed in the **Add or Change Text Segment** window.

DirectTalk/2 displays the Make Phone Connection window if there is not already a phone connection available.

<table>
<thead>
<tr>
<th>Make Phone Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make phone connection by dialing 555-1212.</td>
</tr>
<tr>
<td>To cancel, press F12.</td>
</tr>
<tr>
<td>Canceling may take a few seconds.</td>
</tr>
<tr>
<td>F12=Cancel</td>
</tr>
</tbody>
</table>

Make the phone connection by dialing the number shown.

When the connection is made, DirectTalk/2 will generate the speech for the segment and play it to you over the phone.
Valid Text Segment Characters

The Text Segment Editor monitors all characters you either type into a text segment using the Add or Change option, or copy into a text segment using the import option. The characters are checked against a list of valid characters. The default setting of this list is normally sufficient. This includes all the normal characters with ASCII codes between hex 20 and hex 127, carriage return, line feed, and the accented characters with ASCII codes above hex 127.

If you are in the Add or Change option and type a character that is not in the list of valid characters, DirectTalk/2 signals with a warning beep and does not change the text segment. If you are importing a file that contains a character that is not the list of valid characters, a window is displayed. You can cancel the import or continue the import while changing all invalid characters to spaces.

You may add or remove valid characters from the list. To do this, use an editor to create the file TEXTSEG.CHx in the DirectTalk/2 development directory, where x is the application language letter. For example, text segments being used by a U.S English application have the file name TEXTSEG.CHE. See the language documentation for the language you are using for the list of application language specifiers. Each line in the file is either a comment, an instruction to add a character to the list of valid characters, or an instruction to remove a character from the list of valid characters. The meaning of the line is determined from the first character in the line:

<table>
<thead>
<tr>
<th>Char</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Add a character to the list of valid characters. The character to be added immediately follows the +, and is represented by its decimal ASCII value.</td>
</tr>
<tr>
<td>-</td>
<td>Remove a character from the list of valid characters. The character to be removed immediately follows the -, and is represented by its decimal ASCII value.</td>
</tr>
<tr>
<td>*</td>
<td>A comment line. The line is ignored. (Empty lines are also ignored.)</td>
</tr>
</tbody>
</table>

Here’s an example file:

* This is an example of a TEXTSEG.CHx file.
* Remove lower and upper case E acute from the list
  -130
  -144
* Add vertical bar to the list
+179

The list of valid characters is set each time the Text Segment Editor is selected from the voice Application Developer window. If you are using the Text Segment Editor and need to change the valid character list:

1. Go to an OS/2 window and use an editor to create or modify the file TEXTSEG.CHx. This file should only contains your modification to the default list.
2. Return to the DirectTalk/2 screen and exit from the Text Segment Editor.
3. Reselect the Text Segment Editor. The changes you made to the valid character list will become active at this point.
Copying Text Segments

Instead of writing new text segments you can use one that exists and modify it if necessary. The Copy to and Copy from options on the Edit pull-down menu in the Text Segment Editor window enable you to:

- Copy within the current application
- Copy from another application to the current application
- Copy to another application from the current application

Importing and Exporting Text Segments

If you have text files created by an OS/2 editor or another program and you wish to use them in a voice application, you can import them by selecting Import from the File pull-down in the Text Segment Editor window. The text is stored as a text segment which can be used by the current application.

Similarly, Text segments can be exported as ASCII text files by selecting Export from the File pull-down in the Text Segment Editor window.

Using Common Text Segments

If you have more than one application, you may find that you need the same text segments for more than one application. Rather than creating these text segments for each application or copying them from one application to another, you can share these text segments between two or more applications by placing the text segments in a common voice application.

Normally, DirectTalk/2 retrieves text segments from the voice logic module database associated with the application. If the segment is not found, DirectTalk/2 then searches for a Common application voice logic module database, and finally it searches the System voice logic module database.

You should create this common application as a new application using any name you choose. This application should contain common text segments, but does not need to be a complete application as it is merely acting as a database.

Note: The same common application can also hold any common voice segments and voice logic modules as well. See “Using Common Voice Segments” on page 72 and “Using Common Voice Logic Modules” on page 68.

Once you have created the new application, copy or create all the common text segments. A particular text segment should be in either a specific application or in the common application that is used by the application, but not in both.

To enable DirectTalk/2 to identify a common application to associate with your application, you must add and set the following variables in your Voice Application Developer setup and in the application control file:

- cmn_segmt_appl
  The name of the application which contains the common text segments

- cmn_segmt_srvr
  The network name of the database server the common text segment application uses
Programming Considerations

The primary use of Text-to-Speech is in applications which require variable data to be read to a caller. The larger the amount of data and the more variable it is, the more useful Text-to-Speech will be. Synthesized speech is not a substitute for recorded human voice and should never be used as such. However, if you need to deliver information which, for any reason, is difficult to record then Text-to-Speech is the best way in which to deliver it.

There are two primary considerations when using Text-to-Speech: the user interface, and intelligibility. The user interface concerns itself with the caller. Intelligibility concerns itself with the presentation and recovery when something is not understood.

When designing an application which uses Text-to-Speech, consider the following recommendations:

- Do not mix recorded voice and synthesized voice in the same sentence or phrase.
  - Human voice should be used for all prompts, and for fixed or small vocabulary information. Text-to-Speech should be used for highly variable information, especially if it requires a large vocabulary.
- Do not try to fool the caller into thinking a synthesized voice is a human voice; it will not work.
  - Callers respond much more favorably if your application warns them that they are about to hear a synthesized voice, either by prompt or by natural breaks in the application.
- Remember that a caller receiving unexpected information is likely to have trouble understanding it.
  - If someone is verifying their own name, they will always understand it. If, however, they are hearing an unfamiliar company name or restaurant name, then they may have trouble understanding it. Often, this difficulty is blamed on the Text-to-Speech technology, but anything can be difficult to understand if you are not expecting it.
  - Pacing the speech will help. If words are clearly marked with a pause, then the listener does not have to concentrate on separating the words, but only on understanding them.
- Always offer listeners playback options.
  - Playback options should include one of more of the following:
    - Repeat
    - Repeat slowly
    - Spell
    - Clear spell
  - Repeat slowly can be either slowing the actual speech rate, or adding pauses between words. Clear spell uses a phonetic for capital letters. For example, “Mary” is read as “capital emm as in Michael ay arr wye.”
Text-to-Speech works best with full use of grammar, punctuation, and mixed case.

The Text-to-Speech processing functions best from properly written text in complete sentences. So, as much as possible, use this format. It is especially important to terminate text with punctuation.

**Text-to-Speech Controls**

The DirectTalk/2 Speech Synthesis Feature is highly configurable. There are many controls, or resets, which will affect the way words are pronounced. The controls you can use and the characters used to invoke them are different depending on the software and hardware you are using for Text-to-Speech.

Most of the controls can be set when you configure your system, (see *IBM CallPath DirectTalk/2 Installation Guide* for details of how to configure Text-to-Speech), but they can also be added to your applications to modify the text when it is being played. The Voice Application Developer menus provide assistance for the insertion of the control characters.

The intelligibility and clarity of a synthesized voice can be affected by a number of factors, including volume, speed, voice pitch, and pacing. When judging a synthesized voice, consider that it is in many ways like a human voice but with a regional accent. Once you are used to the accent, understanding greatly improves.

Work with the default synthesized voice before making changes. If you are having trouble understanding the default voice, there are a number of controls or resets which affect the way the voice sounds. However, before experimenting with these and changing the sound of the voice, work on the user interface first. It is often the case that difficulties in understanding have more to do with the data and its presentation than with the way the voice sounds.

**BeSTspeech Controls**

Some of the most useful BeSTspeech controls are listed here. For complete details, together with a complete list of the resets, see the *Understanding BeSTspeech T-T-S* document, which is delivered with your Text-to-Speech hardware. In addition, there are a number of application notes available from BeST and Dialogic. BeSTspeech controls remain in effect on the channel on which they are defined until they are explicitly reset. If Reset control sequence is set in the Telephony Server TTS configuration, that sequence will only be played when the TTS channel is assigned to an application.

**Pause, \texttt{\textasciitilde}2,x\]**

Adds fixed pauses between words.

\[x\] signifies the longest pause, and can take values 1 to 6. A value of \(x=4\) is recommended for repeating information slowly. This is the most effective way to slow down information delivery; it reduces the rate, but does not distort the actual words.

**Speech rate, \texttt{\textasciitilde}r\]**

Changes the speaking rate of the actual words.

\[x\] signifies the percentage change from the default, with positive values slowing speech. This is not as effective because it distorts the spoken words.
Alpha literal or spelling mode, \texttt{\~n1,x}

If \( x=1 \), words will be spelled out normally. If \( x=2 \), words will be spelled using aids to clarify spelling: ‘S for sugar’ for example.

Upper case processing, \texttt{\~n7,x}

Controls how uppercase words are processed.

If \( x \) is a positive number, uppercase words less than \( x \) length are treated as abbreviations and, possibly, spelled out. If \( x = 0 \), this is equivalent to \texttt{\~n7,5}.

If \( x = 1 \), no words will be treated as abbreviations, effectively turning this off.

Volume, \texttt{\~gx}

Overall gain or volume of speech.

The value of \( x \) is in the range 00-25. See “Using Tone Interruption” on page 144 for an example of where this can be useful.

Digit literal mode, \texttt{\~n2,x}

If \( x=1 \), all numbers are spoken as a string of digits. If \( x=0 \), all numbers are spoken as groups; for example, 1024 as “ten twenty-four”.

Full number mode, \texttt{\~n6,x}

If \( x=1 \) then four-digit numbers without commas will have the words “hundreds” or “thousands” added.

Phonetic mode, \texttt{\~p}

This indicates that the following text uses special phonetic spelling, rather than normal spelling, to improve pronunciation.

Table 16 on page 142 shows the characters that can be used for phonetic spelling on a US English TTS system.

Normal text mode, \texttt{\~t}

Return to normal text mode from one of the other modes.

Exception dictionary entry, \texttt{\~x}

Define or use an exception dictionary entry.

See “Using the Text-to-Speech Exception Dictionary” on page 142.

Other resets control the handling of abbreviations, acronyms, voice characteristics, and other, more subtle aspects of Text-to-Speech.

**Lernout & Hauspie (L&H) Controls**

The L&H controls are briefly described here. For more details and further information consult the information supplied with your Text-to-Speech hardware and software. L&H controls are set until the end of a single play when they revert to the L&H default values. If a sequence is specified in the Telephony Server TTS configuration, that sequence is prefixed to any test segment or string that is played.

**Notes:**

1. It is not possible for an application to play a segment or string that sets a play mode and then to play a further segment or string in that mode.

2. L&H software normally expects the \texttt{<ESC>} code as the first character of a control sequence, but this is difficult to insert from a keyboard. To overcome this problem, the \texttt{\~} character should be used as the start of all L&H control codes. DirectTalk/2 converts the \texttt{\~} to \texttt{<ESC>} before passing it to the Text-to-Speech software.
Volume, \textbackslash Vx
Sets the volume of the speech output.
\textbackslash x can take a value between 1 (low volume) and 9 (high volume).
The default is 8.
See “Using Tone Interruption” on page 144 for an example of where this can be useful.

Pitch level, \textbackslash Ix
Determines the pitch level of the spoken text.
\textbackslash x can take a value between 1 (lower pitch) and 9 (higher pitch).
The default is 5

Speech rate, \textbackslash Rx
Determines how fast the text is spoken.
\textbackslash x can be in the range 1 (slow) to 9 (fast).
The default is 5.

Speaker, \textbackslash Sx
Selects a speaker from the range of speakers supported by your system.
\textbackslash x can take a range of values from 1 (low voice) to 9 (high voice).
The default is 1.

Read mode, \textbackslash Mx
Determines how the system will split the input text up into message units.
Each message unit is separately processed and pronounced by the Text-to-Speech system.
\textbackslash x can take the following values:
0 the text is spelled letter by letter.
1 the text is read word by word.
2 the text is read sentence by sentence. The system uses punctuation marks and capitalization to recognize beginning and end of sentences.
3 the system breaks the text when it finds a terminator. Terminators are carriage return, line feed, and all control sequences except \textbackslash p,\textbackslash " , \textbackslash C, and \textbackslash /.
The default is 2.

Wait, \textbackslash Wx
Inserts an extra pause between message units (see “Read mode, \textbackslash Mx” on page 139).
\textbackslash x can take a value from 0 (no additional pause) to 9 (long pause).
The default is 0.
(See also “Pause, \textbackslash Px” on page 140.)

Default values, \textbackslash F
Resets all control parameters to their default values.

Generate a Beep Tone, \textbackslash Bx
Generates a beep tone with a frequency determined by \textbackslash x.
The range for \textbackslash x is 0 (low tone) to 9 (high tone).
Pause, `Px

Inserts a pause between words in a message.

x can be in the range 1 (short pause) to 9 (long pause). A value of x=6 is recommended for repeating information slowly. This is the most effective way to slow down information delivery; it reduces the rate, but does not distort the actual words.

(See also "Wait, "Wx" on page 139.)

Sentence accent

This is used to indicate words which should be emphasized with special duration and intonation.

The control characters should be put in front of the word which is to be accented.

If there are no sentence accent controls in the text, sentence accents are automatically assigned using linguistic rules.

Force continuation, `C

This prevents Read mode 2 (see "Read mode, "Mx" on page 139) from creating sentence ends in the wrong places, as in the following example.

The U.S. `C Federal Court is a respected institution.

Without the forced continuation the TTS system would pause before the word 'Federal' as it looks like the start of a new sentence.

`C must be followed by at least one space character.

Force end of message, `E

This is the opposite of `C and can be used in sentence mode or terminator mode (see "Read mode, "Mx" on page 139) to create an extra break point in the text.

`E must be followed by at least one space character.

Enter phonetic input, `/

This allows text to be input with special phonetic spellings rather than normal spelling to improve pronunciation.

Table 15 on page 141 shows the characters that can be used for phonetic spelling on a US English TTS system.

---

Using Phonetic Input

Text-to-Speech software supports phonetic input, so that words whose spelling deviates from the normal language pronunciation rules can still be pronounced correctly.

To use this facility, the input text should be written using the phonetic alphabet listed in Table 15 on page 141 or Table 16 on page 142, depending on which TTS software you are using.
Phonetic Input for Lernout & Hauspie TTS Systems

For L&H systems the text must be preceded by ~/ control characters and input is interpreted phonetically until the end of the message unit (see "Read mode, ~Mx" on page 139), or until ~/ is entered again.

The following characters are used in place of some of the normal control characters when using phonetic input with L&H TTS software:

- ` ' to accent the following syllable
- `" to accent the following word
- `- to create syllable boundaries
- `# to insert a short pause

<table>
<thead>
<tr>
<th>char</th>
<th>sound</th>
<th>as in</th>
<th>char</th>
<th>sound</th>
<th>as in</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a</td>
<td>fall</td>
<td>p</td>
<td>p</td>
<td>pit</td>
</tr>
<tr>
<td>a</td>
<td>o</td>
<td>got</td>
<td>b</td>
<td>b</td>
<td>bit</td>
</tr>
<tr>
<td>æ</td>
<td>a</td>
<td>cat</td>
<td>t</td>
<td>t</td>
<td>tap</td>
</tr>
<tr>
<td>l</td>
<td>i</td>
<td>fill</td>
<td>d</td>
<td>d</td>
<td>dig</td>
</tr>
<tr>
<td>i</td>
<td>ee</td>
<td>feel</td>
<td>k</td>
<td>c</td>
<td>cat</td>
</tr>
<tr>
<td>U</td>
<td>u</td>
<td>full</td>
<td>g</td>
<td>g</td>
<td>got</td>
</tr>
<tr>
<td>u</td>
<td>oo</td>
<td>fool</td>
<td>f</td>
<td>f</td>
<td>fat</td>
</tr>
<tr>
<td>o</td>
<td>oo</td>
<td>door</td>
<td>v</td>
<td>v</td>
<td>vat</td>
</tr>
<tr>
<td>E</td>
<td>e</td>
<td>fell</td>
<td>e</td>
<td>u</td>
<td>curt</td>
</tr>
<tr>
<td>^</td>
<td>u</td>
<td>cut</td>
<td>T</td>
<td>th</td>
<td>thin</td>
</tr>
<tr>
<td>$</td>
<td>ye</td>
<td>yellow</td>
<td>D</td>
<td>th</td>
<td>then</td>
</tr>
<tr>
<td>J</td>
<td>ai</td>
<td>fail</td>
<td>S</td>
<td>sh</td>
<td>ship</td>
</tr>
<tr>
<td>@</td>
<td>ou</td>
<td>foul</td>
<td>Z</td>
<td>s</td>
<td>leisure</td>
</tr>
<tr>
<td>Y</td>
<td>i</td>
<td>file</td>
<td>l</td>
<td>l</td>
<td>lip</td>
</tr>
<tr>
<td>y</td>
<td>oi</td>
<td>foil</td>
<td>r</td>
<td>r</td>
<td>rip</td>
</tr>
<tr>
<td>O</td>
<td>oa</td>
<td>goal</td>
<td>j</td>
<td>y</td>
<td>yes</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>why</td>
<td>h</td>
<td>h</td>
<td>hat</td>
</tr>
<tr>
<td>m</td>
<td>m</td>
<td>man</td>
<td>n</td>
<td>n</td>
<td>nut</td>
</tr>
<tr>
<td>N</td>
<td>ng</td>
<td>ring</td>
<td>?</td>
<td>glottal stop</td>
<td>eat</td>
</tr>
</tbody>
</table>

Phonetic Input for BeST Speech TTS Systems

For BeST Speech systems, phonetic text must be preceded by `p] control characters. Input is then interpreted phonetically until the `t] control characters are entered to indicate a return to normal text input. When using BeST TTS software, stress symbols are placed after the vowel of the stressed syllable. Primary stress is a single quote `', secondary stress is double quotes "." Boundaries between words are marked by the characters $W. Stress marks and phonetic symbols must always be preceded by a space.

For example:

Three bedroom = `p] th r i `$W b E `d r u m `t]
### Table 16. American English Phonetic Alphabet for BeST speech TTS Systems

<table>
<thead>
<tr>
<th>char</th>
<th>as in</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Consonants</strong></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>watt, wet, woo, quit, Duane, wham</td>
</tr>
<tr>
<td>y</td>
<td>yacht, yet, you, use, argue, yam</td>
</tr>
<tr>
<td>h</td>
<td>hot, heard, who, hi, ahoy, ham</td>
</tr>
<tr>
<td>m</td>
<td>sum, ramp, my, limb, ample, moose</td>
</tr>
<tr>
<td>n</td>
<td>sun, rant, nigh, Lynn, handle, noose</td>
</tr>
<tr>
<td>ng</td>
<td>sung, rank, drunk, long, ankle, ping</td>
</tr>
<tr>
<td>l</td>
<td>lots, stole, feel, sold, lily, led</td>
</tr>
<tr>
<td>r</td>
<td>roster, store, fear, soared, rare, Fred</td>
</tr>
<tr>
<td>f</td>
<td>fat, half, rough, lift, phase, off</td>
</tr>
<tr>
<td>v</td>
<td>vat, have, shove, lived, cover, vivid</td>
</tr>
<tr>
<td>th</td>
<td>booth, author, ether, anthem, thesis, therapy</td>
</tr>
<tr>
<td>dh</td>
<td>smooth, other, either, rhythm, these, there</td>
</tr>
<tr>
<td>s</td>
<td>sue, bus, lace, recent, city, oxen</td>
</tr>
<tr>
<td>z</td>
<td>zoo, buzz, lays, resent, zitty, exact</td>
</tr>
<tr>
<td>ch</td>
<td>batch, chin, hitch, nature, virtual, church</td>
</tr>
<tr>
<td>jh</td>
<td>badge, gin, Jeff, soldier, gradual, judge</td>
</tr>
<tr>
<td>sh</td>
<td>bash, shin, chef, nation, racial, mission</td>
</tr>
<tr>
<td>zh</td>
<td>beige, measure, vision, fusion, casual, seizure</td>
</tr>
<tr>
<td>b</td>
<td>bats, robe, baby, beak, abey, amble</td>
</tr>
<tr>
<td>p</td>
<td>pats, rope, puppy, speak, opaque, ample</td>
</tr>
<tr>
<td>d</td>
<td>door, mad, dime, did, buzzed, road</td>
</tr>
<tr>
<td>t</td>
<td>tore, mat, time, strut, bussed, wrote</td>
</tr>
<tr>
<td>g</td>
<td>got, rag, ogre, Greg, agog, figs</td>
</tr>
<tr>
<td>k</td>
<td>cot, rack, ocher, quake, pique, fix</td>
</tr>
<tr>
<td><strong>Vowels in stressed and unstressed syllables</strong></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>beet, leak, ease, we, ski, eel</td>
</tr>
<tr>
<td>I</td>
<td>bit, lick, is, spirit, hear, ill</td>
</tr>
<tr>
<td>e</td>
<td>bait, lake, came, way, steak, ale</td>
</tr>
<tr>
<td>E</td>
<td>bet, Lech, desk, merry, head, el</td>
</tr>
<tr>
<td>æ</td>
<td>bat, lack, ask, graph, had, Al</td>
</tr>
<tr>
<td>u</td>
<td>boot, Luke, dune, move, stew, coed</td>
</tr>
<tr>
<td>U</td>
<td>put, look, bush, lure, tour, could</td>
</tr>
<tr>
<td>o</td>
<td>boat, choke, flow, woe, oboe, code</td>
</tr>
<tr>
<td>O</td>
<td>bought, chalk, flaw, store, long, cawed</td>
</tr>
<tr>
<td>a</td>
<td>pot, mock, spa, mark, starry, cod</td>
</tr>
<tr>
<td>^</td>
<td>but, luck, done, just, hull, cud</td>
</tr>
<tr>
<td>R</td>
<td>Bert, lurk, earn, mirth, journey, curd</td>
</tr>
<tr>
<td>ay</td>
<td>bite, like, hire, why, eyes, aisle</td>
</tr>
<tr>
<td>oy</td>
<td>boy, join, hoist, coy, oink, oil</td>
</tr>
<tr>
<td>aw</td>
<td>bout, pound, house, cow, ouch, owl</td>
</tr>
</tbody>
</table>

### Using the Text-to-Speech Exception Dictionary

The exception dictionary allows you to define a specific pronunciation for a word or acronym to reflect a local usage that is different from the Text-to-Speech default. For acronyms, the Text-to-Speech feature either tries to pronounce the acronym as a word or, for common acronyms such as ACCT, expands the acronym to the word that it stands for. For example, if you do not want ACCT expanded to Account (the
default), you could use the exception dictionary to specify ACCT as Accounting (or
Banana, if you so choose).

The exception dictionary is a file which contains Text-to-Speech resets or controls,
and is loaded onto the Dialogic Text-to-Speech card when DirectTalk/2 starts up. It
is active for all Text-to-Speech lines in the system. The exception dictionary
filename is specified in the Telephony Server TTS configuration, and the file you
specify should exist in the main DirectTalk/2 directory.

Creating an Exception Dictionary for BeST Speech TTS Systems
To use the exception dictionary with DirectTalk/2, create it as an ASCII file named
EXCEPT.DIC. This file should contain one exception definition per line. It can also
include any other Text-to-Speech controls. For further information see the section
"Understanding BeSTspeech T-T-S" in the Dialogic Text to Speech for OS/2
manual.

For exceptions, the syntax is as follows:

\[ \text{ASST = s I s ' t = n t } \text{t} \]

where ASST is the acronym or word to be excepted, and the string following is a
set of phonemes (defined in Appendix B of the Dialogic documentation) which will
be spoken whenever the Text-to-Speech feature encounters that word.

The exception dictionary will take any of the defined resets and is one way to set
defaults for the Text-to-Speech function. For details of the use of resets, refer to
the Dialogic Text-to-Speech documentation.

Any exceptions should only be loaded using the exception dictionary once per
board during startup. Using the EXCEPT.DIC file will do this. It is not
recommended that each application attempt to modify the exception dictionary by
using any of the Play_Text functions or that the exception dictionary be modified
once multiple TTS channels (lines) are active. You can use the Text-to-Speech
editor in the Voice Application Developer to experiment with the exceptions during
development. However, if you have applications in production, you should not
modify the exception dictionary on the board on the same system.

Creating an Exception Dictionary for Lernout and Hauspie TTS
Systems
The creation of an exception dictionary for L&H is a two stage process:

1. Create a text file with an extension .DCT.
   Each line in the file has the format:
   
   Word 1 translation 1
   Word 2 translation 2
   ...

   The maximum number of entries is 400 words. The translation may consist of
   a number of words, phonetic spellings, and so on.

2. Run the DOS program:

   MKDCT <filename>

   to create a binary file named <filename>.dat from <filename.dct>. The
   MKDCT program is in the directory in which the L&H software is installed.
Using Tone Interruption

If you are using tone interruption with Text-to-Speech, the interruption may not be detected if the configuration is not optimum. This problem can be resolved by adding a gain control string at the front of the message which is expecting a tone interrupt. For example:

`˜g -15]Hello (BeST)`

or

`˜V6 Hello (L&H)`

reduces the volume and improves the detection rate of the interrupt tone.

TTS output can also be falsely recognized as tones and it therefore may be necessary to increase the time period specified for hang-up tones. A minimum of half a second is recommended. This problem is most likely to occur if the tone definition is only a duration and not a cadence.
Chapter 17. Telecommunication Devices for the Deaf

The Telecommunication Devices for the Deaf (TDD) option allows your application to interact with hearing-impaired callers. You do not require any additional hardware to allow DirectTalk/2 to interact with TDD devices if you use this software option. You can use the TDD option with any DirectTalk/2 application. It accepts DTMF input from your caller as well as TDD characters.

No protocol exists to identify that a caller is using a TDD device and so it is recommended that you provide dedicated lines if you are using this feature.

Use the Assign_Resource action to assign a TDD resource to the application for the duration of each call. Use the Free_Resource action to unassign it. Assign the TDD resource for each call so that you ensure that the TDD state is correctly initialized and maintained for the call.

The actions provided by the TDD Feature are:

Receive_TDD
   Receive TDD data from a caller.

Send_TDD
   Send TDD data from a database.

Send_TDD_String
   Send TDD data from a variable or literal.
Chapter 18. Analog Display Service Interface (ADSI)

Analog Display Services Interface (ADSI) is a Bell Communications Research (Bellcore) standard defining a protocol for data transmission over a voice grade telephony channel. ADSI generating devices such as a Central Office or Interactive Voice Response (IVR) system can communicate with ADSI compatible telephones. Such telephones have some processing capability, an area of read/write memory, a screen, navigation keys (to scroll information up, down, left and right), and softkeys. The softkeys can be programmed to perform different functions at different times.

The data transmitted to an ADSI telephone can be of two different types:

**Server Display Control**

This type of data is sent when an IVR voice application involves a continuing connection between the ADSI generating device, such as DirectTalk/2 and the ADSI telephone.

**Feature Download Management**

This allows a number of alternative display and key setups (overlays) to be downloaded and stored in the ADSI telephone. These overlays can be selected by SDC actions or by various signals (events) occurring at the telephone.

Adding ADSI generating support to Interactive Voice Response (IVR) systems means users, who are able to upgrade from standard to ADSI compatible telephones, can be provided with voice applications that are easier to use and richer in function, and with stand alone ADSI telephone programs.

ADSI telephones cannot pass control data directly to each other, the control data stream can only be generated by a switch (PBX/CO) or by an IVR such as DirectTalk/2.

---

**ADSI Support in DirectTalk/2**

DirectTalk/2 contains the following components to support ADSI:

- A script language to enable you to write the control information and programs. This language also allows for parameter substitution points at which parameters passed from DirectTalk/2 can be included in the downloaded data.

- Online help and the *IBM CallPath DirectTalk/2 ADSI Programmer’s Guide* which describe how to create ADSI Script source files using the language provided, and how to convert them into ADSI scripts which can be interpreted by an ADSI telephone.

- A user interface to assist with the following tasks:
  - The addition of ADSI Script source files to a Voice Application Database
  - Compilation of the script source files into a form that is recognized by ADSI telephones
  - Specification of the parameter data that is to be included in the ADSI scripts

- Two user actions to enable the ADSI data to be sent to and received from an ADSI telephone:
– Send_ADSI
– Get_ADSI

(The existing Get_a_Tone and Get_Tone_String actions can be used to receive DTMF.)

Note: Sending ADSI data to an on-hook telephone is not supported.

---

Creating a DirectTalk/2 ADSI Application

The following is an outline of the steps you need to perform to add ADSI support to existing applications, or to create new applications:

1. Use the Voice Application Developer (VAD) to create a Voice Application in which you want to incorporate ADSI function.

Notes:

a. The voice program should test for errors that may be passed back from the Send_ADSI action and then continue as normal.

b. ADSI telephones can transmit alphanumeric data in the form of encoded DTMF tones.

(See “Creating the Voice Program” on page 18 for details of how to create a voice application using the VAD.)

2. Decide whether SDC, or FDM scripts, or both are required.

3. Design the ADSI screen appearance and the expected responses at each stage in the application.

4. Design the content of the various overlays to be included in each FDM script.

5. Write the ADSI script source files, using an ASCII text editor, and paying attention to the following:

For SDC scripts

• Several SDC Functions that are called by the Send_ADSI actions in a Voice Program can be defined in a single file.

• Break the operations into discrete functions that can be individually executed by the application. Using a voice application as an example, where you would speak a new prompt or voice segment to the caller, define a new ADSI function.

• Include definitions of the soft keys that are used by the functions included in the script.

• In each function statement, include any substitution parameters used in the function.

For FDM scripts

• Create a separate FDM script source file for each main statement.

• Define all the softkeys that are used by the main and other overlays that are included in the script.

• List all the substitution parameters that are used by the softkeys in a softkeyparms statement. Insert this statement before the softkey definitions.
Include, in the main statement, all the substitution parameters that are used by the main and overlay statements included in the script.

For both SDC and FDM scripts

- Write the script source files using the ADSI language statements which are described in the IBM CallPath DirectTalk/2 ADSI Programmer’s Guide.
- Save the files to the main voice system directory.

6. Use the VAD to create, in the ADSI Script Database of the Voice Application, a Script for each of the source files.

7. Use the VAD to compile each Script.
   
   Compilation converts Script source file Function and Overlay definitions into ADSI Statements that can be sent by the Send_ADSI action.
   
   A successful compile will also automatically create:
   - A Parameter List for each Function in an SDC script source file.
   - A single Parameter List for an FDM script source file.

8. Use the VAD to access the Parameters Lists and specify the data to be substituted when the associated SDC Function or FDM script is sent by the Send_ADSI action.

A sample ADSI script source file is included with DirectTalk/2. See “Sample ADSI Application” on page 311.
Part 4. Actions

This section describes the user actions that are provided with DirectTalk/2. In Chapter 19, “Using Actions” on page 153 the actions are listed according to the tasks they are used for and in Chapter 20, “List of Actions” on page 159 all the actions are described in detail and are listed alphabetically.

Chapter 19. Using Actions

- Telephony Actions ........................................ 153
- Interacting with the Caller ................................ 154
- Voice-Related Actions .................................... 155
- Creating and Using Variables ............................... 155
- Structuring the Voice Application .......................... 156
- Interfacing with a Host Application ........................ 156
- Collecting Statistics and Logging Messages ............... 157
- Using Voice Messaging .................................... 157
- Working with a Database .................................. 157
- Telecommunication Devices for the Deaf ..................... 158
- Analog Display Services Interface (ADSI) Actions ........ 158
- Creating New Actions .................................... 158

Chapter 20. List of Actions

- Add_Record .................................................. 160
- Add_Stat_Item ............................................... 161
- Assign_Resource ............................................ 162
- Branch ....................................................... 164
- Calculate ...................................................... 166
- Call_Agent_Rel ............................................. 167
- Call_Extend_Cfg ............................................ 168
- Call_Extend_Init ............................................ 170
- Call_Extend_Ref ............................................. 173
- Call_Extend_Tran ............................................ 175
- Call_Referral ............................................... 177
- Call_Transfer ............................................... 179
- Check_Password ............................................. 182
- Clear_Tones .................................................. 183
- Close_Database ............................................. 184
- Comment ....................................................... 185
- Compare_Chars .............................................. 186
- Compare_Numbers .......................................... 187
- Concatenate .................................................. 188
- Connect_Screen ............................................. 189
- Create_Variable ............................................ 191
- Delete_Record .............................................. 192
- Delete_Variable ............................................ 193
- Delete_Voice .................................................. 194
- Disconnect_Scr .............................................. 195
- Free_Resource .............................................. 196
- Get_ADSI .................................................... 197
- Get_Length ................................................... 199
- Get_Messages ................................................ 200
- Get_Next_Record ............................................ 204

© Copyright IBM Corp. 1992, 1996
Chapter 19. Using Actions

This chapter lists the actions you can use depending on the task you are trying to perform. Some actions can be used in more than one task, but each action is listed only under its prime task. The detailed descriptions of the actions are listed alphabetically by action name in Chapter 20, “List of Actions” on page 159.

Telephony Actions

Use DirectTalk/2 telephony actions to dial, answer, transfer, or hang up calls to or from callers. These actions also control the telephony resource of DirectTalk/2.

You can use:

Assign_Resource
To permanently assign a Voice Recognition or Speech Synthesis channel to a voice program channel.

Call_Agent_Rel
To release the agent from a referred call and return to a standard call setup.

Call_Extend_Cfg
To indicate which Extend Call functions are possible and how they will operate.

Call_Extend_Init
To initiate a call to an agent with whisper.

Call_Extend_Tran
To complete the transfer of a call that was started by Call_Extend_Init.

Call_Extend_Ref
To complete the referral of a call that was started by Call_Extend_Init.

Call_Transfer
To make a simple call transfer with no interaction with the agent before transfer.

Call_Referral
To make a simple call referral with no interaction with the agent before referral.

Free_Resource
To free an assigned Voice Recognition or Speech Synthesis channel from the voice program channel.

Hang_up_Phone
To hang up the phone and end the incoming or outgoing call.

Place_a_Call
To dial an outgoing call with call progress monitoring.

Put_Tone_String
To specify a string of numbers to dial and use these numbers to dial an outgoing call without call progress monitoring.

Set_Timeout
To set the time an action should wait for the caller to respond.
Wait_For_Call
To wait for and answer an incoming call.

Wait_For_Hang
To detect when a party in a call hangs up the phone.

Interacting with the Caller

Use the DirectTalk/2 caller interaction actions to play voice segments to callers, determine and receive tone input from callers, clear tone input from the buffer, and recognize voice responses from a caller.

You can use:

Clear_Tones
To clear any buffered keys that the caller pressed, but the application has not received.

Get_a_Tone
To determine the key the caller pressed on the telephone key pad.

Get_Tone_String
To receive the keys the caller pressed on the telephone key pad.

Get_Voice_Cont
To recognize a continuous set of voice responses from a caller within a given vocabulary set and return the response to the application. You must have the optional Voice Recognition Feature installed to use this action.

Get_Voice_Resp
To recognize one or more voice responses from a caller within a given vocabulary set and return the response to the application. You must have the optional Voice Recognition Feature to use this action.

Play_Module
To run a voice logic module to play voice segments to callers.

Play_Text
To synthesize speech from a text record in a DirectTalk/2 database and then play that speech to the caller.

Play_Text_String
To synthesize speech from a text string, such as the value of a variable, and then play that speech to the caller.

VR_Get_String
To recognize one or more single words spoken by the caller, and verify that the recognized words are the ones that the caller actually said. You must have the optional Voice Recognition Feature to use this action.

VR_Get_Yes_No
To initiate a dialog with the caller to receive a response of ‘yes’ or ‘no’. You must have the optional Voice Recognition Feature to use this action.
Voice-Related Actions

In addition to using a voice logic module to play voice segments, you can use DirectTalk/2 voice-related actions to play recorded messages to callers, to record and save messages from callers, and to organize messages by record key in a database. See the DirectTalk/2 Administrator’s Guide for more information on creating the database.

You can use:

Delete_Voice
To delete a voice record from a database.

Play_Voice
To play voice records that are recorded from callers and saved in a database using the Record_Voice action.

Record_Voice
To record a caller's voice and save it in a database.

Creating and Using Variables

You can use the following actions to create and use local and global variables:

Calculate
To perform simple integer arithmetic on variables or literals and store the result in a variable.

Compare_Chars
To compare two values as characters and indicate the result.

Compare_Numbers
To compare two values as numbers and specify the result.

Concatenate
To concatenate two variables or literals and store the result in a variable.

Create_Variable
To create and add variables to the local and global variable pools.

Delete_Variable
To delete variables from the local and global variable pools.

Get_Length
To determine the length of a string and store the length in a variable.

Get_Substring
To extract a portion of a string and copy it into a variable.

Search_String
To determine the position of the first occurrence of a literal or variable string from a starting point within the source string.

Set_Variable
To assign new value to a variable.
Structuring the Voice Application

You can use the following actions to structure your voice application:

- **Branch**
  To branch on a variable value.

- **Comment**
  For a comment or a place holder within the application.

- **Link_to_Appl**
  To run a subapplication.

- **Pause**
  To specify the number of seconds to suspend running the application.

- **Return_from_Appl**
  To return processing from a subapplication to the linking application.

Interfacing with a Host Application

You must have the optional Communications Feature to use actions that provide an interface with a host application. The actions provide 3270, 5250, and ASCII via terminal emulation. The actions are:

- **Connect_Screen**
  Connect to the host application.

- **Disconnect_Scr**
  Disconnect from the host application.

- **Get_Screen_Data**
  Get information from the virtual screen.

- **Refresh_OIA_Action**
  Refresh the Operator Information Area.

- **Search_Screen**
  Perform a single search for the text you specify on the terminal screen each time you specify the action.

- **Send_Keys_to_Scr**
  Simulate keystrokes to type data on the screen and press function keys that the host application is to process.

- **Send_Alert**
  Send a Netview alert.

- **Switch_Screen**
  Switches the active host screen.

- **Wait_Scr_Update**
  To perform one search per second for the number of seconds you specify until the text you specify appears in the expected location on the screen.
Collecting Statistics and Logging Messages

You can use statistics and message actions in a DirectTalk/2 application to:

- Define the statistical records for DirectTalk/2 to collect
- Write the statistical record to the DirectTalk/2 Statistics Server
- Delete variables from the statistical record
- Write a message to the message log file
- Start and stop clocks to record durations of events

The actions you can use are:

**Add_Stat_Item**
Define a variable to be a statistics item, such as part of a statistics record.

**Log_a_Message**
Read a user-defined message with a single text substitution from the application message file, format it, and write it to the message log file.

**Log_Statistics**
Write the statistics record you specify to the DirectTalk/2 Statistics Server.

**Remove_Stat_Item**
Delete the statistics item from the statistics record.

**Start_Clock**
Start one of eight clocks to time events.

**Stop_Clock**
Stop one of eight clocks and store the duration from the last Start_Clock action in the variable you specify.

Using Voice Messaging

You must have the optional Voice Messaging Feature to use the directory and mailbox actions. These actions are:

**Check_Password**
Check an input password against a directory entry.

**Get_Messages**
Retrieve messages from a mailbox.

**Put_User_Info**
Change information in the directory.

**Search_Directory**
Retrieve a directory entry based on telephone number.

**Take_A_Message**
Record telephone messages and store them in a mailbox.

Working with a Database

You can use the following actions to work with a database:

**Add_Record**
To add a new record to the application database.

**Close_Database**
To close a DirectTalk/2 database.
Delete_Record
To delete an existing record from an application database.

Get_Next_Record
To perform sequential access through the database.

Get_Record
To read a record from an application database.

Update_Record
To replace an existing record in the application database with new record data.

Telecommunication Devices for the Deaf
These actions can only be used with the optional Telecommunication Devices for the Deaf feature.

The following actions can be used to communicate with TDD devices:

Receive_TDD
Receive TDD data from a caller.

Send_TDD
Send TDD data from a database.

Send_TDD_String
Send TDD data from a variable or literal.

Analog Display Services Interface (ADSI) Actions
These actions can only be used with the optional Analog Display Services Interface (ADSI) feature.

ADSI requires ADSI information to be sent to, and received from an ADSI telephone. The following actions can be used in your voice programs to send and receive the ADSI data.

Get_ADSI
To receive ASCII data from an ADSI telephone.

Send_ADSI
To send ADSI data to an ADSI telephone.

Creating New Actions
In addition to using the actions sent with DirectTalk/2, you can also create your own actions. The instructions for creating your own actions are in the IBM CallPath DirectTalk/2 Application Programmer’s Guide.
Chapter 20. List of Actions

This chapter contains detailed information on all the actions referred to in the previous chapter. This information is arranged in alphabetical order based on the action names.
Add_Record

Use the Add_Record action to add a new record to the DirectTalk/2 application database.

Parameters

Database server name (Input, Required, Default=none)
The network name of the database server that the action uses to access the application database.

File name (Input, Required, Default=none)
The name of the application database in which to add the record.

Record key (Input, Required, Default=none)
The key for the record. This value must be the same length as the key length defined when the database was created.

Record data (Input, Required, Default=none)
The data associated with the key. The data can be of variable length. The maximum length of the data is the record length defined when the database was created. If the length of the data exceeds the defined record length, DirectTalk/2 will log an error message.

Return Codes

Record added (RC=0)
The action added the record to the application database.

Record already exists (RC=1)
The action did not add the record to the application database because the record already exists in the database.

Error (RC=2)
An error condition occurred. You specified an incorrect database server name or file name, the action could not write the data to the file, or the record data is too long.

System Variables

close_database (Input)
To close all databases after adding a record with this action, set the close_database variable to Y or y. Note that this option may cause severe performance degradation. The default setting for close_database is N or n.

Comments

The Add_Record action:

- Locks the record so that only one application can access the record at a time.
- Searches the database for the record before adding it to the database. If the record already exists, the action does not add the record to the database.
- Unlocks the record before returning to the application.
Add_Stat_Item

Use the Add_Stat_Item action to define variable names to store in a statistics record. You can define a maximum of 32 such user variables. The voice application writes the statistics record to the statistics log file when you use the Log_Statistics action.

Parameter

Variable (Input, Required, Default=None)
The name of the variable to add to the statistics record.

Return Codes

Added (RC=0)
The action added the statistics item.

Too many items (RC=1)
The statistics record already has 32 entries.

Comments

- The voice application updates the values of the variables in the statistics record.
- These values can include timing events and the results of clock actions, counts of occurrences of events or any information required to help manage the DirectTalk/2 application. The statistics record is written when you use the Log_Statistics action.
- When the Log_Statistics action is executed, it picks up items added with the Add_Stat_Item action only if they were added in the same application, and not in a linked-to application.
Assign_Resource

Use the Assign_Resource action to permanently assign a Voice Recognition, Speech Synthesis, Telecommunication Devices for the Deaf (TDD), or ADSI channel to a voice application session.

Parameters

Device (Input, Required, Default=none)
The name of the resource to be assigned. Specify 'VR' for Voice Recognition, 'TTS' for Speech Synthesis, 'TDD' for Telecommunication Devices for the Deaf, or 'ADSI' for ADSI resource.

Requirements (Input, Optional, Default=none)
The specific requirements to be specified for the channel being assigned. If no free channel meets the requirements, or the requirements are not recognized by the installed version of the device, the 'Not Available' action return is taken.

You can specify one or more Requirements. Normally only one is required to obtain a suitable resource channel. If you do specify more than one, join them together with a + sign.

The Requirements parameter is either just a keyword indicating a feature that must be supported by the channel being assigned, or a keyword and a value indicating that the feature indicated must have the value specified. The key words differ depending on the device type and are case independent.

VR Keywords

CONT Select a channel that supports Continuous VR.
DISC Select a channel that supports Discrete VR.
VSTOP Select a channel that supports prompt voice stop.
CUTTHRU Select a channel that supports prompt voice stop with word recognition.

VOCAB=value Select a channel that has the vocabulary specified by "value" loaded. The vocabulary name is given without a file extension.

TTS Keywords

DICT=value Select a channel that has the exception dictionary specified by "value" loaded. The exception dictionary filename is given with a file extension.

The device channel assigned is also limited by the voice line Ports Linked To selected during the Telephony Server configuration.
Return Codes

Resource added (RC=0)
   The requested resource has been permanently added (or is already assigned) to the voice application session.

Not available (RC=1)
   The type of resource requested is not available.

Error (RC=2)
   An error occurred while trying to assign the resource.
   For example, the input parameter might have been incorrectly specified: only “ADSI”, “TDD”, “TTS”, and “VR” are valid.

Comments

- The Assign_Resource action is not a prerequisite to using any of the resources available to a given channel.
- The action will remove a specified resource from the resource pool, and assign it permanently to the voice channel. In some situations, this may improve performance. This will also guarantee that the resource is available to the voice channel at all times. While the action effectively reduces the number of resources available to other voice channels, it also reduces by one the number of voice channels requiring a service. You should carefully consider the use of this action.
- Only one instance of a resource type can be assigned to an application at one time.
Branch

Use the Branch action for branching on a variable value. The Branch action evaluates the first character of the variable and returns its value to the corresponding return code.

Parameters

Variable (Input, Required, Default=none)
The name of a defined variable.

Case sensitive (Input, Optional, Default=none)
Specifies if the evaluation of the first character of the variable parameter is case sensitive.

Specify 'Y' in the first byte of the case sensitive parameter to make the evaluation case sensitive. The case sensitive parameter can be a literal or a variable.

Override (Input, Optional, Default=none)
A string which defines the association of characters with a return code. The override parameter can be a literal or a variable. The string is made up of one or more definitions, where a definition has the following syntax:

\[ a=n \]

where \( a \) is one or more characters and \( n \) is the return code to which it is assigned. A colon divides successive definitions. If \( a \) is one or more characters, it indicates an OR condition; that is, if the variable parameter starts with any one of the characters, then the specified return code will be returned. If \( a \) is not included, that return code will never be returned.

\( n \) can be 0-9, *, or #. The existence of an override string will completely replace the default. The default is defined as follows:

\[ 0=0;1=1;2=2;3=3;4=4;5=5;6=6;7=7;8=8;9=9;CH=*=T=\# \]

Spaces are ignored. For example:

\[ Y=1;N=0;CH=9;=8 \]

would indicate that a value string starting with “Y” returns a 1, “N” returns a 0, “C” or H returns a 9, and nothing returns an 8.

Return Codes

Branch 0 (RC=0)
Branch 0 return code

Branch 1 (RC=1)
Branch 1 return code

Branch 2 (RC=2)
Branch 2 return code

Branch 3 (RC=3)
Branch 3 return code

Branch 4 (RC=4)
Branch 4 return code
Branch 5 (RC=5)
  Branch 5 return code

Branch 6 (RC=6)
  Branch 6 return code

Branch 7 (RC=7)
  Branch 7 return code

Branch 8 (RC=8)
  Branch 8 return code

Branch 9 (RC=9)
  Branch 9 return code

Branch * (RC=10)
  Branch 10 return code

Branch # (RC=11)
  Branch # return code

Not in list (RC=T1)
  Not in list return code

Error (RC=T2)
  Error return code

System Variables

case_sensitive (Input)
  See note below.

Comments

It is recommended that the system variable case_sensitive is used for the case sensitive parameter, so that case sensitivity is consistent throughout your application, if required.
Calculate

Use the Calculate action to perform integer arithmetic on variables or literals and store the result in a variable.

Parameters

Variable for result (Output, Required, Default=none)
The variable to receive the result. If the result of the calculation is less than \(-2147483646\) or greater than \(2147483646\), the action will return the Error return code.

Value 1 (Input, Required, Default=none)
The value of the first operand. You must specify an integer value. You can specify a variable or a literal. This value must fall between \(-2147483646\) and \(2147483646\).

'+' , '-' , '*' , '/' , '%' (operation) (Input, Required, Default=none)
The operation to perform on the operands. You can only specify the symbols for addition, subtraction, multiplication, division, and remainder respectively.

Value 2 (Input, Required, Default=none)
The value of the second operand. You must specify an integer value. You can specify a variable or a literal. This value must fall between \(-2147483646\) and \(2147483646\).

Return Codes

Done (RC=0)
The action performed the operation on the operands and placed the result into the result variable.

Error (RC=1)
An error condition occurred. The action may have:

- Tried to divide by zero
- Specified a nonnumeric value
- Specified a source variable that did not exist
- Specified an invalid operation
- Given a result less than the minimum allowed
- Given a result greater than the maximum allowed

Comments

- If the result variable does not exist, the Calculate action creates it as a local variable.
- An example of Calculate action appears in the CALC sample application.
Call_Agent_Rel

Note: This action is currently implemented only on E1 configurations.

Use the Call_Agent_Rel action when an application is in a referral situation and the agent needs to be released

Return Codes

Agent released (RC=0)
The agent is successfully released from the call. Any resources used to connect the agent are freed. If the caller was on hold he is returned to the attached state.

ext_canhear_c, ext_canplayto_c, ext_canhear_a, and ext_canplayto_a system variables are set to indicate how much involvement the voice system has in the call.

No agent (RC=1)
There is no valid agent to release. Call_Extend_Init did not precede this action, or the preceding action did not return successfully. The caller remains in his current state.

No caller (RC=2)
There is no existing call.

Caller hung up (RC=HUP)
The voice system detected that the caller disconnected. The agent is released from the call.

The ext_canhear_c, ext_canplayto_c, ext_canhear_a, and ext_canplayto_a system variables are updated.

It is possible for more than one return code to be valid and in this case the return code precedence is:
1. No caller (RC=2)
2. No agent (RC=1)
3. Caller hung up (RC=HUP)
4. Agent released (RC=0)

System Variables
The system variables and their settings after this action are discussed in “E1 Extend Call System Variables” on page 81.

Comments
The values returned in the system variables are shown in Table 9 on page 83.
Call_Extend_Cfg

Note: This action is currently implemented only on E1 configurations.

Use the Call_Extend_Cfg action to determine what Extend Call actions are possible on the installed system configuration, and how they will operate.

Return Codes

OK (RC=0)
The capabilities of the phone line the application is using have been collected and stored in the system variables.

Not feasible (RC=1)
The configuration is not capable of performing extend call actions.

Resource unavailable (RC=2)
The voice session is not able to attach an Extend Call resource. There may either be none configured or they may all be being used.

System Variables

e_num_ext_lines
The number of Extend Call resources available (not necessarily free) in the system.

e_lines_for_tran
The number of lines required to make a transfer.

e_lines_for_ref
The number of lines required to make a referral.

e_tran_switch_at
Where the switching/connection takes place when a transfer is made:

1 at switch
2 on voice system voice bus (e.g. SCbus)
3 on voice system adapter card (e.g. Aculab)

e_full_tran_pos
Indicates whether a transfer with the voice system completely removed from the call is possible. (1=yes, 0=no.)

e_min_tran_scrrn
The minimum screening level that will be accepted by the call transfer action. Can be set to NONE, DIALED, RINGING, or FULL

e_w_caplayto_a
Indicates whether it is possible to play to the agent after a successful Call_Extend_Init action.

e_w_canhear_a
Indicates whether it is possible to hear the agent after a successful Call_Extend_Init action.

e_w_canplayto_c
Indicates whether it is possible to play to the caller after a successful Call_Extend_Init action.
`e_w_canhear_c`
Indicates whether it is possible to hear the caller after a successful Call_Extend_Init action.

`e_r_canplayto_a`
Indicates whether it is possible to play to the agent after a successful Call_Extend_Referral or Call_Referral action.

`e_r_canhear_a`
Indicates whether it is possible to hear the agent after a successful Call_Extend_Referral or Call_Referral action.

`e_r_canplayto_c`
Indicates whether it is possible to play to the caller after a successful Call_Extend_Referral or Call_Referral action.

`e_r_canhear_c`
Indicates whether it is possible to hear the caller after a successful Call_Extend_Referral or Call_Referral action.

**Comments**

- Only the system variables listed above are set, the standard set of Extend Call system variables are not set by this action.
- The values returned in a standard E1 environment are shown in Table 17.

<table>
<thead>
<tr>
<th>System variable</th>
<th>Standard E1 Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>e_num_ext_lines</td>
<td>depends on config</td>
</tr>
<tr>
<td>e_lines_for_tran</td>
<td>2</td>
</tr>
<tr>
<td>e_lines_for_ref</td>
<td>2</td>
</tr>
<tr>
<td>e_tran_switch_at</td>
<td>3</td>
</tr>
<tr>
<td>e_full_tran_pos</td>
<td>0</td>
</tr>
<tr>
<td>e_min_tran_scrn</td>
<td>FULL</td>
</tr>
<tr>
<td>e_w_canplayto_a</td>
<td>1</td>
</tr>
<tr>
<td>e_w_canhear_a</td>
<td>1</td>
</tr>
<tr>
<td>e_w_canplayto_c</td>
<td>0</td>
</tr>
<tr>
<td>e_w_canhear_c</td>
<td>0</td>
</tr>
<tr>
<td>e_r_canplayto_a</td>
<td>0</td>
</tr>
<tr>
<td>e_r_canhear_a</td>
<td>1</td>
</tr>
<tr>
<td>e_r_canplayto_c</td>
<td>0</td>
</tr>
<tr>
<td>e_r_canhear_c</td>
<td>0</td>
</tr>
<tr>
<td>e_r_canhear_c</td>
<td>1</td>
</tr>
</tbody>
</table>
Call_Extend_Init

Note: This action is currently implemented only on E1 configurations.

Use the Call_Extend_Init action to initiate a call to the agent on the specified number. The optimum successful completion is for the caller to be on hold and the agent to be connected to the voice system. An acceptable completion is for the agent and voice system to be connected with the caller still connected.

Parameters

Phone Number (Input, Required, Default=none)
The telephone number of the agent.

Final State (Input, Optional, Default='EITHER')
How the extend call is probably going to complete, either to a referral or to a transfer. If the application is not sure at this stage, for example it could depend on a response from the agent, this parameter can be specified as EITHER. Possible values are 'TRANSFER', 'REFERRAL', or 'EITHER'.

Valid Connections (Input, Optional, Default=none)
This parameter indicates that the Connected return code should also be given if the call is answered by an answering machine, fax/modem, or operator SIT tones, depending on the keywords used.

The possible keywords are ANSWER, FAX, and OPERATOR, and they can be combined using +, &, or a space.

Return Codes

Connected (RC=0)
The action completed successfully and the agent is connected through a screened call. The completion may involve the allocation of extra resource to the telephony session. The system variable ext_num_of_lines indicates if an extension phone line is being used.

Callers State
If possible the caller is put in a state such that he is unable to hear or play ('on hold'). If this is not possible the actions will proceed with the caller attached. The ext_canplayto_c and ext_canhear_c system variables are set to indicate the caller’s state.

Agents State
The agent is attached to the call and is able to speak to and hear from the voice system. The ext_canplayto_a and ext_canhear_a variables are set to 1 (=Yes).

Busy (RC=1)
The agent’s phone was busy.

No answer (RC=2)
No answer from the agent’s phone.

No ring (RC=3)
The agent’s phone did not ring.
Operator Intercept (RC=4)
The Special Intercept Tones (SIT) indicating, for instance, that the number is no longer in service, or the number has changed, were received.

Note: If the Valid Connections parameter is set to include OPERATOR, Connected will be returned instead.

Answering Machine (RC=5)
An answering machine was detected.

Note: If the Valid Connections parameter is set to include ANSWER, Connected will be returned instead.

No dial tone (RC=6)
The voice system could not get a dial tone and could not dial.

FAX/Modem (RC=7)
A FAX or modem was detected.

Note: If the Valid Connections parameter is set to include FAX, Connected will be returned instead.

No resource (RC=8)
Not enough resources are available to make this extended call. This return is generated if a second line is required and no free extend call lines are available.

No caller (RC=9)
There is no caller attached to the session and the voice system is therefore not in the correct state to perform this action.

Completion not possible (RC=10)
The configuration does not support the final state specified.

Already have agent (RC=11)
The agent is already attached to the call. The state of the agent and caller remains the same.

Agent hung up (RC=T2)
Agent disconnect detected after an initial connection was established.

Caller hung up (HUP)
The caller has disconnected from the call (hung-up). If this occurs before the agent answers, the call to the agent is terminated. If the agent has already answered the call, the agent is not disconnected and this is indicated by the ext_canplayto_a and ext_canhear_a variables being set to ‘yes’.

Agent hung up (T2)
The agent hung up after an initial connection was established.

After unsuccessful completions (RC=1,2,3,4,5,6,7,8,10), the system state is as follows:

Agent’s State
The call to the agent is disconnected and any resources allocated to this action are freed.

Caller’s State
The caller is still connected to the voice system. The ext_canplayto_c and ext_canhear_c system variables are set to 1 (Yes).
It is possible for more than one return code to be valid and in this case the return code precedence is:

1. The return codes:
   - RC=11 Agent already attached
   - RC=9 No caller
   - RC=8 No resource

2. Caller hung up (RC=HUP)

3. Other returns

System Variables

The system variables and their settings after this action are discussed in “E1 Extend Call System Variables” on page 81.

Comments

- This action will always provide a fully screened agent attachment.
- If this action returns successfully, the application will be connected to the agent and the action should be followed by a Call_Extend_Tran action, or a Call_Extend_Ref action. Call to other actions can be made before these calls, for example, Play_Voice, or Get_Tones.
- Switch dependent information such as dual or single line and the operation to get the attention of the switch, are set at configuration time.
- The values returned in the system variables are shown in Table 9 on page 83.
Call_Extend_Ref

Note: This action is currently implemented only on E1 configurations.

Use the Call_Extend_Ref action to setup a referral call that has been started with a successful Call_Extend_Init action. The optimum successful completion of this call is for the agent and caller and voice system to be connected so that they can all hear from and speak to each other. If this is not possible, the caller and agent should be able to hear and speak to each other and the voice system should be able to hear either both caller and agent, or just the agent.

Return Codes

Referral made (RC=0)
The agent and caller are connected and the voice system is monitoring the call. If the caller was on hold he is returned to the attached state.

ext_canhear_c, ext_canplayto_c, ext_canhear_a, and ext_canplayto_a indicate how much involvement the voice system has in the call.

The system variable ext_switched_at records where the referral took place:

1 PBX  
2 voice system  
3 adapter card

No agent (RC=1)
There is no valid agent to connect to. Call_Extend_Init did not precede this action, or the preceding action did not return successfully. The caller remains in his current state.

No caller (RC=2)
There is no existing call.

Agent hung up (RC=T2)
The voice system detected that the agent disconnected before the referral completed. The caller is taken off hold and is reconnected to the voice system.

If the caller also hangs up, Caller hung up is returned instead.

Caller hung up (RC=HUP)
The voice system detected that the caller disconnected before the referral completed. In this case the agent remains connected to the voice system.

Note: It is not always possible to detect the caller disconnect.

The ext_canhear_c, ext_canplayto_c, ext_canhear_a, and ext_canplayto_a system variables are updated.
It is possible for more than one return code to be valid and in this case the return code precedence is:

1. No caller (RC=2)
2. No agent (RC=1)
3. Caller hung up (RC=HUP)
4. Agent hung up (RC=T2)
5. Referral made (RC=0)

System Variables

The system variables and their settings after this action are discussed in “E1 Extend Call System Variables” on page 81.

Comments

- Switch dependent information, such as dual or single line and the operation to get the attention of the switch, is set at configuration time.
- The values returned in the system variables are shown in Table 9 on page 83.
Call_Extend_Tran

Note:  This action is currently implemented only on E1 configurations.

Use the Call_Extend_Tran action to complete a call transfer that has been started with a successful Call_Extend_Init action, or to transfer a previously referred call. The optimum successful completion of this call is for the agent and caller to be connected and the voice system to be removed from the call. If this is not possible, the voice system connects the caller to the agent using a second phone line and monitors the call until either the caller or agent disconnects.

Return Codes

Transferred (RC=0)
The agent and the caller are connected together and are both disconnected from the voice system.

The system variable ext_switched_at records where the transfer took place.

1  PBX
2  voice system
3  adapter card

No agent (RC=1)
There is no valid agent to connect to. Call_Extend_Init, or a referred call, did not precede this action, or the preceding action did not return successfully.

No caller (RC=2)
There is no caller attached to the session and the voice system is therefore not in the correct state to perform this action.

Agent hung up (RC=T2)
The voice system detected that the agent disconnected before the transfer completed. The caller is taken off hold and is reconnected to the voice system.

If the caller also hangs up, Caller hung up is returned instead.

Caller hung up (RC=HUP)
The voice system detected that the caller disconnected before the transfer completed. In this case the agent remains connected to the voice system.

Note:  It is not always possible to detect the caller disconnect.

The ext_canhear_c, ext_canplayto_c, ext_canhear_a, and ext_canplayto_a system variables are updated.

It is possible for more than one return code to be valid and in this case the return code precedence is:

1. No caller (RC=2)
2. No agent (RC=1)
3. Caller hung up (RC=HUP)
4. Agent hung up (RC=T2)
5. Transferred (RC=0)
System Variables

The system variables and their settings after this action are discussed in “E1 Extend Call System Variables” on page 81.

Comments

- Switch dependent information such as dual or single line and the operation to get the attention of the switch, are set at configuration time.
- The values returned in the system variables are shown in Table 9 on page 83.
Call_Referral

**Note:** *This action is currently implemented only on E1 configurations.*

Use the Call_Referral action to do a single step referral with no interaction with the agent. The optimum successful completion of this call is for the agent and caller and voice system to be connected so that they can all hear from and speak to each other. If this is not possible, the caller and agent should be able to hear and speak to each other and the voice system should be able to hear either both caller and agent, or just the agent.

**Parameters**

**Phone Number (Input, Required, Default=none)**
The telephone number of the agent.

**Valid Connections (Input, Optional, Default=none)**
This parameter indicates that the Connected return code should also be given if the call is answered by an answering machine, fax/modem, or operator SIT tones, depending on the keywords used.

The possible keywords are ANSWER, FAX, and OPERATOR, and they can be combined using +, &, or a space.

**Return Codes**

**Referral made (RC=0)**
The referral to the requested number is complete. The variables `ext_canhear_c`, `ext_canplayto_c`, `ext_canhear_a`, and `ext_canplayto_a` indicate how much involvement the voice system has in the call.

**Busy (RC=1)**
The agent's phone was busy.

**No answer (RC=2)**
No answer from the agent’s phone.

**No ring (RC=3)**
The agent’s phone did not ring.

**Operator Intercept (RC=4)**
The Special Intercept Tones (SIT) indicating, for instance, that the number is no longer in service, or the number has changed, were received.

**Note:** If the Valid Connections parameter is set to include OPERATOR, Referral made will be returned instead.

**Answering Machine (RC=5)**
An answering machine was detected.

**Note:** If the Valid Connections parameter is set to include ANSWER, Referral made will be returned instead.

**No dial tone (RC=6)**
The voice system could not get a dial tone and could not dial.
FAX/Modem (RC=7)
A FAX or modem was detected.

Note: If the Valid Connections parameter is set to include FAX, Referral made will be returned instead.

No resource (RC=8)
Not enough resources are available to make this extended call. This return is generated if a second line is required and no free extend call lines are available.

No caller (RC=9)
There is no caller attached to the session and the voice system is therefore not in the correct state to perform this action.

Referral not poss (RC=10)
It is not possible to complete this action with the configuration being used.

Too many agents (RC=11)
An agent is already attached to this session and the voice system is not in the correct state to perform the action.

Agent hung up (RC=T2)
The agent could disconnect (hang-up) between being contacted and the referral completing. If this is detected, this return code is generated, any resource allocated to the agent is released, and, if the caller has been placed on hold, he or she is reconnected to the voice system.

Caller hung up (RC=HUP)
The caller has disconnected from the call (hung-up). If this occurs before the agent answers, the call to the agent is terminated. If the agent has already answered the call, the agent is not disconnected and this is indicated by the ext_canplayto_a and ext_canhear_a variables being set to 'yes'.

It is possible for more than one return code to be valid and in this case the return code precedence is:

1. Referral not poss (RC=10)
2. Too many agents (RC=11)
3. No caller (RC=9)
4. No resource (RC=8)
5. Caller hung up (RC=HUP)
6. Agent hung up (RC=T2)
7. Other returns

System Variables
The system variables and their settings after this action are discussed in “E1 Extend Call System Variables” on page 81.
Call_Transfer

Note:  This action is currently implemented only on E1 configurations.

Use the Call_Transfer action to do a single step transfer with no interaction with the agent.

This action provides a variety of levels of monitoring by the voice system (screening levels). The voice system can stop monitoring as soon as the number is dialled, or it can continue to monitor until the call is answered, and there are also intermediate levels. The screening levels affect the time the voice program is involved in the call and therefore the time the line is unavailable to take another call.

Note:  With the current DirectTalk/2 implementation, screening must be set to FULL, because a completed call to the agent must be established before the transfer can take place.

Parameters

Phone Number (Input, Required, Default=none)
The telephone number of the agent.

Screen Level (Input, Required, Default=FULL)
The minimum level of screening required. If FULL is specified, this is equivalent to a screened transfer. Valid values are:

'NONE'  Return immediately after the agent’s number has been dialled. A successful return will always be received, even if a valid line has not been obtained.

'FULL'  The action will only return successfully if the agent has been contacted.

'DIALED'  The action checks that a valid line is obtained and then dials the number.

'RINGING'  The action returns successfully when a normal ringing tone is detected on the agent’s line.

Valid Connections (Input, Optional, Default=none)
This parameter indicates that the Connected return code should be used if the call is answered by an answering machine, fax/modem, or operator SIT tones, depending on the keywords used.

The possible keywords are ANSWER, FAX, and OPERATOR, and they can be combined using +, &, or a space.

This parameter is used only if full screening is selected.

Return Codes

Transferred (RC=0)
The action completed to at least the level requested. The caller has been released and the application is free to receive another call. Any additional resource required to contact the agent have been freed.

Busy (RC=1)
The agent’s phone was busy.
No answer (RC=2)
No answer from the agent’s phone.

No ring (RC=3)
The agent’s phone did not ring.

Operator Intercept (RC=4)
The Special Intercept Tones (SIT) indicating, for instance, that the number is no longer in service, or the number has changed, were received.

Note: If the Valid Connections parameter is set to include OPERATOR, Transferred will be returned instead.

Answering Machine (RC=5)
An answering machine was detected.

Note: If the Valid Connections parameter is set to include ANSWER, Transferred will be returned instead.

No dial tone (RC=6)
The voice system could not get a dial tone and could not dial.

FAX/Modem (RC=7)
A FAX or modem was detected.

Note: If the Valid Connections parameter is set to include FAX, Transferred will be returned instead.

No resource (RC=8)
Not enough resources are available to make this extended call. This return is generated if a second line is required and no free extend call lines are available.

No caller (RC=9)
There is no caller attached to the session and the voice system is therefore not in the correct state to perform this action.

Transfer not poss (RC=10)
It is not possible to complete this action with the configuration being used.

Invalid screen level (RC=11)
The screen level parameter is invalid for this configuration, or possibly for all configurations.

Application quiesce (RC=T1)
The transfer completed successfully and a request to stop the application has been received.

Agent hung up (RC=T2)
The agent has hung up between being contacted and the transfer completing. Any call extension resource used will be released, and the caller, if on hold, will be reconnected to the voice application session.

Caller hung up (HUP)
The caller has disconnected from the call (hung-up). The agent is disconnected and the call terminated, whether or not the agent has already answered.

After unsuccessful completions (RC=1,2,3,4,5,6,7,8,10,11), the system state is as follows:
**Agent’s State**

The call to the agent is disconnected and any resources allocated to this action are freed.

**Caller’s State**

The caller is still connected to the voice system. The `ext_canplayto_c` and `ext_canhear_c` system variables are set to 1 (Yes).

It is possible for more than one return code to be valid and in this case the return code precedence is:

1. Transfer not poss (RC=10)
2. Invalid screen level (RC=11)
3. No caller (RC=9)
4. No resource (RC=8)
5. Caller hung up (RC=HUP)
6. Agent hung up (RC=HUP)
7. Application quiesce (RC=T1)
8. Other returns

**System Variables**

The system variables and their settings after this action are discussed in “E1 Extend Call System Variables” on page 81.

**Comments**

- If a screen level of NONE is specified, this action will return successfully even if a dial tone is not achieved.
- In some configurations, the level of feedback to the caller may be restricted if you request a low screen level. For instance if a screen level of NONE is requested, and the agent’s phone is engaged, the caller will be cut off without even hearing the busy signal. This should be taken into account when designing applications.
Check_Password

Use the Check_Password action to check the password the caller specifies against the password in the directory entry. You must have the optional Voice Messaging Feature to use this action.

Parameters

Directory ID (Input, Required, Default=none)
The unique directory identification value. Use the <prefix>_idno variable that is returned by the Search_Directory action.

Password (Input, Required, Default=none)
The input password to test against the directory entry.

Return Codes

Match found (RC=0)
The action found a match for the password you specified.

Match not found (RC=1)
The action could not find a match for the password you specified.

Error (RC=2)
A severe directory error occurred while the action was trying to check the password.

System Variables

directory_server (Input)
The name of the remote server on which the directory is held.

Comments

• You can use the <prefix>_idno variable returned from the Search_Directory action as the directory identification value.

• An example of the Check_Password action is in the sample get application (see Appendix B, “Sample Applications” on page 305).
Clear_Tones

Use the Clear_Tones action to clear any buffered keys that the caller pushed, but which the application has not yet received. The Clear_Tones action returns the hardware buffer to an empty state.

Return Codes

Buffer cleared (RC=0)
The Clear_Tones action cleared any keys the caller pressed which the application had not yet received.

Caller hung up (RC=HUP)
The caller hung up the telephone.

Comments

• A caller can fast-path through voice menus by pressing the correct keys prior to the prompt for the key (voice menu).

• If the caller makes an invalid selection, the buffer will contain tones other than the invalid selection. The results of these tones may cause the action to go to a known point in the application and confuse the caller. You may want to use this action to clear the buffer of all tones. The voice application can then prompt the caller again for input.

• Some DirectTalk/2 actions, such as Play_Module, detect tones but do not retrieve the keys. If those actions detect tones, you may want to follow them with this action.
Close_Database

Use the Close_Database action to close a DirectTalk/2 database.

Parameters

**Database server name (Input, Required, Default=none)**

The network name of the database server that the action uses to access the application database.

**File name (Input, Required, Default=none)**

The name of the application database to close.

Return Codes

**Database closed (RC=0)**

The action closed the database successfully.

**Database in use (RC=1)**

The database is currently being accessed by another application, session, server, or requester. The database is not closed.

**Error (RC=2)**

An error condition occurred. You specified an incorrect database server name or file name. See the application session log for additional information.

Comments

In general, the DirectTalk/2 database actions leave the database files open for maximum performance. There are two drawbacks to this. First, the open databases cannot be copied or accessed via OS/2 because OS/2 protects files opened for access. Second, in the event of a power failure or abnormal system shutdown, you may lose data.

Use **Close_Database** only when necessary especially for a commonly-used database. You might want to use it, for example, periodically, to allow for backups.

The **Get_Record** and **Get_Next_Record** actions open the database only for read access. You do not need to close the database if it is only used by these actions.
Comment

Use the Comment action for a comment or a place holder within the application.

Return Codes

Done (RC=0)
The action did not perform any operation.

Comments

- You can use the Comment action to add descriptive information to help you understand the structure of the application. For example, you can delimit processing for each option on a menu with a comment.

- Executing large blocks of consecutive Comment actions (say, 20 or more) will begin to affect performance. Comment blocks should be structured so that the step number for the Done return code of all comment steps points to the last comment in the block. In this way, regardless of the size of the block, no more than two comments will ever be executed.
Compare_Chars

Use the Compare_Chars action to compare two values as characters and indicate the result. You can specify variables or literal values.

Parameters

Value 1 (Input, Required, Default=none)
The first value to compare. You can specify a variable or literal.

Value 2 (Input, Required, Default=none)
The second value to compare. You can specify a variable or literal.

Case sensitive (Input, Optional, Default='N')
Specify 'Y' or 'y' to make the comparison of the input values case sensitive. You can also specify a variable as the case sensitive parameter.

Return Codes

Value 1 < Value 2 (RC=0)
The first value is less than the second value.

Value 1 = Value 2 (RC=1)
The first value is equal to the second value.

Value 1 > Value 2 (RC=2)
The first value is greater than the second value.

Error (RC=3)
The action could not compare the two values. You may have specified a variable that does not exist.

Comments

- You must enclose literals in quotes.
- It is recommended that the system variable case_sensitive be used for this parameter, so that case sensitivity is consistent throughout your entire application, if required.
- An example of the Compare_Chars action is in the CALC sample application.
Compare_Numbers

Use the Compare_Numbers action to compare two values as numbers and specify the result. You can specify variable or literal values.

Parameters

Value 1 (Input, Required, Default=none)
The first value to compare. The value must be numeric. You can specify a variable or a literal.

Value 2 (Input, Required, Default=none)
The second value to compare. The value must be numeric. You can specify a variable or a literal.

Return Codes

Value 1 < Value 2 (RC=0)
Value 1 is less than value 2.

Value 1 = Value 2 (RC=1)
Value 1 equals value 2.

Value 1 > Value 2 (RC=2)
Value 1 is greater than value 2.

Error (RC=3)
The action encountered a problem while trying to compare the two values. You may have specified:
- A variable that does not exist
- A variable that contains a nonnumeric value

Comments

- The input values must fall between –2 147 483 646 and 2 147 483 646.
- You must enclose literal values in single quotes.
- An example of the Compare_Numbers action is in the CALC sample application.
Concatenate

Use the Concatenate action to concatenate two variables or literals and store the result in a variable.

Parameters

Variable for result (Output, Required, Default=none)
The variable to receive the result.

String 1 (Input, Required, Default=none)
The value of the first string.

String 2 (Input, Required, Default=none)
The value of the second string.

Return Codes

Done (RC=0)
The action concatenated the values and stored the value in the result variable.

Error (RC=1)
An error condition occurred. You may have specified:
- A source variable that does not exist
- Invalid data
- An invalid variable name

Comments

If the variable in which you want to store the result does not exist, DirectTalk/2 creates it as a local variable.
Connect_Screen

Note: You must have the optional Communications Feature to use this action.

Use the Connect_Screen action to connect to the host session. This action must complete successfully before you can use any other host interface actions.

Parameters

Emulator sess group (Required, Input, Default=none)
The name of the emulator session group from which the session should be allocated.

Emulation session groups are defined with Setup when you configure your system. The default emulation session names are contained in the following variables:

- CM32_server for 3270 CM/PC
- 5250_server for 5250CM/PC
- LUA3270_server for 3270 LUA
- 3270_server for 3270 ARTIC
- ASCII_server for Ascii TCP/IP

You can check your session group name by looking at the Communications panels in DirectTalk/2 Setup.

session id var (Required, Output, Default=none)
The name of a variable to receive the identification of the session.

Return Codes

Connected (RC=0)
The connection with the host application has been established.

Error (RC=1)
The action encountered a problem while trying to connect to the host application.

Session unavailable (RC=2)
There were no emulator sessions available.

Already connected (RC=3)
The application is already connected to a host emulation session.

Comments

- You should typically use Connect_Screen to connect to a host before logging on and before using the Wait_for_Call action.
- Up to four host sessions may be connected at any one time.
- You may need to rerun the Connect_Screen action if the host connection is lost.
- If an application is using a specific emulator, Emulator session group should have the group name of the emulator instead of the host server name specified in the 3270_server, CM32_server, LUA32_server, 5250_server, or ASCII_server variables.
- During Setup, subsets of Emulator Sessions can be created by putting them into separate Groups. This may be necessary if, for example, some of the...
sessions are connected to HOST1 and some are connected to HOST2. When using the Connect_Screen action, connection to HOST1 or HOST2 can then be requested by specifying the appropriate Emulation Session Group.

- The Connected return code can be received before display of the host screen is complete. To prevent further actions attempting to operate on incomplete data, follow Connect_Screen with Wait_Scr_Update, or a Search_Screen action set to wait for data at the bottom of the screen.
Create_Variable

Use the Create_Variable action to create and add local and global variables. Local variables are only available to the voice program that is using this action. Global variables are available to all applications running on an individual phone line. The action creates the variable with an initial null value.

Parameters

Variable (Output, Required, Default=none)
The name of the variable to create. You can specify the variable name as a literal.

Type (Input, Optional, Default='local')
Specifies the variable as ’local’ or ’global’.

Return Codes

Variable created (RC=0)
The action created the variable and added it to the appropriate variable pool.

Duplicate variable (RC=1)
The variable name you specified already exists in the variable pool you specified.

Comments

- Use the Create_Variable action to create global variables before using them in an action. In all other cases, using this action is optional, provided the variable is created by some other action (such as Set_Variable) before use.
- Many actions which manipulate strings and make assignments automatically create local variables, for any required output variables.
- An example of the Create_Variable action is in the CALC sample application.
- When the variables are used in an application, the local variable pool is searched first, followed by the global variable pool.
Delete_Record
Use the Delete_Record action to delete the record you specify from the DirectTalk/2 application database.

Parameters

Database server name (Input, Required, Default=none)
The network name of the database server that the action uses to access the application database.

File name (Input, Required, Default=none)
The name of the application database from which to delete the record.

Record key (Input, Required, Default=none)
The key for the record the action is to delete. This value must be the same length as the key length defined when the database was created.

Return Codes

Record deleted (RC=0)
The action deleted the record.

Record not found (RC=1)
The action could not find the record in the database.

Error (RC=2)
An error condition occurred. You specified an incorrect database server name or file name.

Record locked (RC=3)
The record you want to delete is currently locked.

System Variables

close_database (Input)
To close all databases after deleting a record with this action, set the close_database variable to Y or y. Note that this option may cause severe performance degradation. The default setting for close_database is N or n.

db_record_wait (Input)
If the first character of the db_record_wait variable is set to Y or y, and the record to be deleted is currently locked by another session, the Delete_Record action waits until the lock is freed before deleting the record.

If the first character of the db_record_wait variable is not set to Y or y and the record to be deleted is currently locked by another session, the Delete_Record action does not delete the record and returns the Record locked return code.

If the record to be deleted is not currently locked by another session, the Delete_Record action deletes the record regardless of the value of the db_record_wait variable.
Delete_Variable

Use the Delete_Variable action to delete local and global variables.

Parameters

Variable (Output, Required, Default=none)
The name of the variable to delete.

Type (Input, Optional, Default='local')
The variable pool where the variable resides. Use 'local' to specify the variable is in the local variable pool. Use 'global' to specify the variable is in the global variable pool.

Return Codes

Variable deleted (RC=0)
The action deleted the variable from the variable pool.

Variable not found (RC=1)
The variable name does not exist in the type of variable pool you specified.

Comments

- Be careful when deleting global variables because other linked subapplications may use the variable.
- Do not delete system variables.
Delete_Voice

Use the Delete_Voice action to delete a voice record from a database.

Parameters

Database server (Input, Required, Default=none)
The network name of the database server.

File name (Input, Required, Default=none)
The name of the database file which contains the voice record. The file name should be the same OS/2 file name that was used when the database was created. When the voice records database is created, the record length is 4096 and the key length is 19.

Record key (Input, Required, Default=none)
The key that identifies the voice record to delete.

Return Codes

Deleted (RC=0)
The action deleted the voice record. The action returns this return code even if the voice record did not exist in the database.

Error (RC=1)
There was a problem deleting the record from the database. The database server may be inaccessible.

Comment

The Delete_Voice action is not part of the Voice Messaging actions provided by DirectTalk/2.
Disconnect_Scr

Use the Disconnect_Scr action to disconnect from the host application. You must have the optional Communications Feature to use this action.

Parameter

Unbind Flag (Input, Optional, Default='N')

This parameter is no longer supported. If it is specified, it is ignored.

Return Codes

Disconnected (RC=0)

The action disconnected from the host.

Error (RC=1)

The action could not disconnect from the host.

Not connected (RC=2)

The action could not disconnect from the host session because the application is not currently connected to the host.

Comments

- You typically use the Disconnect_Scr action during controlled shutdown of the application.
- You can use a combination of the Disconnect_Scr action and the Connect_Screen action to simulate turning a terminal off and on during error recovery.
- If you connected to multiple host sessions, use the Switch_Screen action to switch to the session you wish to disconnect before issuing this action. You must issue this action once for each host session that you want to disconnect.
Free_Resource

Use the Free_Resource action to free an assigned Voice Recognition, Speech Synthesis, Telecommunication Devices for the Deaf, or ADSI channel from the voice program channel. The freed channel is returned to the appropriate resource pool.

Parameters

Device (Input, Required, default=None)
The name of the device to be freed: ‘VR’ for voice recognition, ‘TTS’ for speech synthesis, ‘TDD’ for Telecommunication Devices for the Deaf, or ‘ADSI’ for ADSI.

Return Codes

Resource Freed (RC=0)
The specified resource channel type was successfully freed.

No attachment (RC=1)
The specified resource channel is not currently assigned to this voice program channel.

Error (RC=2)
The specified parameter was not valid (not VR or TTS in any case mix), or there was an internal processing error.
Get_ADSI

Use the Get_ADSI action to receive ASCII data from an ADSI telephone. The ADSI device transmits DTMF tones which are converted into an ASCII text string which is stored in a variable that is specified by the action.

Parameters

**Target Variable (Output, Required, Default=none)**
The variable used to store the ASCII text string. If the variable does not exist it is created.

**Duration (Input, Optional, Default=none)**
The length of time in seconds to wait for ADSI data. If this parameter is not specified, or is set to 0, the action waits until the caller hangs up, maximum characters are received, or a termination character is received.

**Maximum characters (Input, Required, Default=’30’)**
The action returns when this number of decoded ASCII characters have been received. An ADSI telephone can send up to 40 characters.

**Termination chars (Input, Optional, Default=none)**
The action returns after receiving any of the specified termination characters. Any character or function key can be used as a termination character.

Return Codes

**Stop on duration (RC=0)**
The ADSI data has been received and converted within the duration period. This code is returned at the end of the duration period. The converted data is stored in the specified Target Variable.

**Stop on max char (RC=1)**
The maximum number of ADSI characters have been received, converted, and stored in the specified Target Variable.

**Stop on term char (RC=2)**
The caller has sent a termination character to mark the end of the ADSI data. The data has been converted and stored in the specified Target Variable. The termination character is stored in the specified Target Variable and in the variable ADSI_term_char.

**Error (RC=3)**
A character has been sent that cannot be converted. The data sent may have become corrupted.

**Error (RC=4)**
An error condition has occurred. You may not have specified the Target Variable parameter. An appropriate message is displayed.

**HUP (RC=HUP)**
The caller has hung up.
Comments

Tones are buffered on the ADSI telephone until a send command is issued from the ADSI script. The ADSI telephone can hold a maximum of 40 characters, any data added after this limit is reached is lost.

The encoded DTMF tones can be received using the Get_a_Tone and Get_Tone_String actions.
Get_Length

Use the Get_Length action to determine the length of a string and store the length in a variable.

Parameters

Variable for result (Output, Required, Default=none)
The name of the variable to receive the result.

String (Input, Required, Default=none)
The data whose length you want to determine. You can specify a variable or a literal. If the string is null, the result will be '0'.

Return Codes

Done (RC=0)
The action stored the length in the result variable.

Error (RC=1)
An error condition occurred. For example, you specified:

- A source variable that does not exist
- An invalid variable name

Comments

You must enclose literal values in quotes.
Get_Messages

Use the Get_Messages action to retrieve any new or old messages from a mailbox and play them to the user. You must have the optional Voice Messaging Feature to use this action.

Parameters

Mailbox number (Input, Required, Default=none)
The number of the mailbox from which messages are to be retrieved. You can use the value in the <prefix>_mailbox variable returned by the Search_Directory action.

Type (Input, Required, Default='Locked')
Specifies whether the mailbox is to be locked or shared during retrieval. You can specify:

Locked
Only one user can retrieve messages from the mailbox. The action collects and plays the message in the order that was specified when the mailbox was created.

Shared
Several users can retrieve messages from the mailbox at the same time. Only one user can retrieve each message.

Contents (Input, Required, Default='Voice')
Specifies the type of message to retrieve. You must specify 'Voice'.

Forward digits (Input, Optional, Default='')
The minimum number of trailing digits of the telephone number that are required to forward a message to another mailbox. If you specify zero ('0') or a null value, the user cannot forward messages.

Return Codes

Okay, caller hung up (RC=0)
The user hung up after listening to all the new messages, but before the action ended.

Okay, caller on line (RC=1)
The caller listened to all the messages and is still on the line.

More, caller on line (RC=2)
The caller listened to one or more messages and pressed the key to bypass the rest of the menu. There are still new messages in the mailbox and the caller is still on the line.

Mailbox empty (RC=3)
There are no messages in the mailbox.

Error (RC=4)
The action encountered an error.

Mailbox in use (RC=5)
The action tried to access the mailbox, but it is currently locked by another user.

Caller hung up (RC=14)
The caller hung up before the action completed. There are still new messages in the mailbox.
System Variables

**Note:** All required system variables are defined and set to default values in the Voice Application Developer Setup file during installation and configuration of the optional Voice Messaging Feature.

`mailbox_server` *(Input)*

The network name of the mailbox server. The action uses the mailbox server to access the mailbox.

`gmsg_msg_server` *(Input)*

The network name of the database server for the message database. The action uses the database server to access the message database and retrieve the voice message.

`gmsg_msg_db` *(Input)*

The name of the voice message database.

`directory_server` *(Input)*

The network name of the directory server. The action uses the directory server to find the mailbox the user wants to forward the message to. If you do not specify the directory server, the user cannot forward messages.

`gmsg_notebook` *(Input)*

Qualifies the mailbox to a notebook in that mailbox. This allows a user to retrieve a message in a notebook. If you do not specify a value for this variable or if you specify a null value, the action processes the mailbox in-basket.

`gmsg_phone` *(Input)*

Provides the caller's telephone number in the form of a text record. The structure of the record allows it to be used by a User Action, but it cannot be spoken back to the caller. The action uses this variable to create a description to attach to a message that a user forwards. You can use the value in the `<prefix>_phone` variable from the Search_Directory action.

Comments

- Voice messaging actions require tone telephones. The actions cannot use voice recognition.
- The Get_Messages action uses several prerecorded voice segments. These voice segments are supplied with DirectTalk/2 and are stored in the system voice segment database. The names of these segments start with `gmsg_`. You may rerecord these segments or you can override them by recording them in the voice application or common voice segment databases. Use the voice segment editor for the SYSTM application to find the text of each segment. If you reword the segments, make sure that you do not change the meaning or intent of the segments so that the segments become misleading.
- You can designate mailboxes as shared or locked during message retrieval. If you specify the mailbox as locked, only one user can access the mailbox. If you specify the mailbox as shared, multiple callers can access the mailbox.
The Get_Messages action performs the following steps for a locked scan:

1. Constructs a list of all new and old messages in the in-basket or the notebook you specify.

2. The action plays the number of messages and asks the user if the user wants to review each class of new or old messages. The action plays new messages first, followed by any old messages. The action plays each message, preceded by a header that specifies the date and time the message was recorded.

3. After the action plays each message to the user, it asks the user what to do with the message. The user can:
   - Replay the message. The action replays the header and the message.
   - Delete the message. The action deletes the message.
   - Forward the message. The action asks the user for a telephone number and a message to precede the forwarding message. After the user provides the information, the action sends the message to the mailbox the user requests. If the action cannot find the mailbox, it cancels the forward request. The action continues to process the message.
     
     **Note:** If the trailing digits of the telephone number supplied by the caller is not unique, the message will be forwarded to the first mailbox found which matches the trailing digits. To prevent this, ensure that the required number of forward digits is large enough to make the telephone number entered be unique.
   - Save the message. The action sets the status of the message in the mailbox or notebook to 'old'.
   - Archive the message. The action asks the user for a notebook to copy the message into. If the notebook does not exist, the action creates one, as long as the user has the room to create the notebook. The maximum number of notebooks is defined in the directory entry for the user.
     
     **Note:** The original message is not deleted.
   - Reposition to a specific message. The action asks the user for the number of the message. If the message exists, the action plays the message. If the message does not exist or was deleted, the action plays an error message.
   - Continue to next message. The action plays the next message or informs the user that there are no more messages.
   - Bypass remaining messages. The action skips all the remaining messages in the current class and processing should continue with Return Code 1, More, caller online.

4. If the action detects that the telephone was hung up at any point, processing should continue with Return Code 0, Okay, Caller hung up, if there are no new messages, or with Return Code HUP, Caller hung up, if there are new messages.

The steps for a shared scan are the same except that DirectTalk/2 only plays each message to one user and the 'Reposition to a specific message' option is
not available. If someone else is using the mailbox, the user will not be able to hear all the messages.

- See “GET Application” on page 305 for an example of the Get_Messages action.
Get_Next_Record

Use the Get_Next_Record action to perform a sequential access through a DirectTalk/2 application database.

Parameters

**Database server name (Input, Required, Default=none)**
- The network name of the database server the action uses to access the application database.

**File name (Input, Required, Default=none)**
- The name of the application database from which the action retrieves the record.

**Record key variable (Input/Output, Required, Default=none)**
- A variable containing the current key. The action retrieves the next record with a key greater than the value you specify for this parameter. You can use an exclamation point (!) to specify that you want to retrieve the first record in the database.
- The action replaces the value in this variable with the key of the record it retrieves. When the action reaches the end of the database file, it returns a null value.

**Record data variable (Output, Required, Default=none)**
- The variable that receives the data from the record associated with the key. When the action reaches the end of the file, it returns a null value.

Return Codes

**Record found (RC=0)**
- The action found a record and put the data associated with the key in the Variable for data parameter.

**End of file (RC=1)**
- The action reached the end of the file. The action processed all the records or there were no records in the file for the action to process.
- The action sets the data and key variables to a null value.

**Error (RC=2)**
- An error condition occurred. You specified an incorrect database server name, file name, or key.

System Variables

**db_record_lock (Input)**
- Indicates whether or not the action will lock a record retrieved for update.
- If the first character of the db_record_lock variable is set to Y or y, the Get_Next_Record action retrieves and locks the record whose key follows sequentially after the specified Record key. The record is unlocked when it is deleted, updated, when a Get_Next_Record is performed by the same session, or when a Get_Record is performed by the same session for a different record. If the record returned by the action was previously locked by the current session, the lock is not maintained, but will be re-established as specified by db_record_lock.
If the first character of the `db_record_lock` variable does is not set to Y or y, the `Get_Next_Record` action retrieves the record whose key follows sequentially after the specified record key as read-only and does not lock the record.

### db_record_wait (Input)
Indicates whether or not the `Get_Next_Record` action is to wait for the next record to become available, or to continue with another record in the database sequence.

## Comments

To perform a sequential access:

1. Set the key variable to '! ' to indicate the start of the database.
2. Use this action and specify the key variable to retrieve the first record.
3. Process this record.
4. Run this action in a loop until you receive the End of File return code. The action automatically updates the Record key variable parameter and uses it to retrieve the next record.

The records that the action accesses are not locked.

If the record following that specified by the Record key parameter is already locked, the `Get_Next_Record` action ignores that record and attempts to retrieve the next record.

If the record specified by the Record key parameter is already locked and the first character of the `db_record_wait` variable is set to Y or y, the `Get_Next_Record` action waits until the record is freed before returning the data and the Record found return code.

**Note:** `db_record_lock` must be set to Y or y for `db_record_wait` to have any effect. Otherwise, `db_record_wait` will be ignored, that is, it will act as `db_record_wait=N`.

If the record specified by the Record key parameter is already locked and the first character of the `db_record_wait` variable does not exist or is not set to Y or y, the `Get_Next_Record` action ignores that record and attempts to retrieve the next record in sequence.
Get_Record

Use the Get_Record action to search an application database and return a specific record in the variable you specify.

Parameters

- **Database server name (Input, Required, Default=none)**
  The network name of the database server used to access the database.
- **File name (Input, Required, Default=none)**
  The name of the database file you want to search.
- **Record key (Input, Required, Default=none)**
  The key for the record you want to find in the file.
- **Target Variable (Output, Required, Default=none)**
  The variable to receive the record data.

Return Codes

- **Record found (RC=0)**
  The action found the key and placed the data associated with the key in the target variable.
- **Record not found (RC=1)**
  The action did not find the key in the database. The action did not place any value in the result variable.
- **Error (RC=2)**
  An error condition occurred. You may have specified an incorrect database server, file name, or record key.
- **Record locked (RC=3)**
  The record you want to get is currently locked.

System Variables

- **db_record lock (Input)**
  Indicates whether or not the action will lock a record retrieved for update. If the first character of the db_record_lock variable is set to Y or y, the Get_Record action retrieves and locks the record. The record is unlocked when it is deleted, updated, when a Get_Record is performed by the same session, or when another Get_Record is performed by the same session for a different record. If the record specified by the Record key parameter is already locked by the current session, the lock is maintained.

  If the first character of the db_record_lock variable is not set to Y or y, the Get_Record action retrieves the record as read-only and does not lock the record.
**db_record_wait (Input)**

Indicates whether or not the Get_Record action is to wait for the next record to become available, or to continue with another record in the database sequence.

If the first character of the db_record_wait variable is set to Y or y, and the specified record is locked by another session, the Get_Record action waits until the record is freed before returning the data and the Record found return code.

If the first character db_record_wait variable is not set to Y or y, and the specified record is locked by another session, the Get_Record action returns the Record locked return code and no record data.

**Comments**

The length of the record key does not need to be the same length as the database key length, but position is important because the action will assume trailing blanks to the length of the database key. For example, a record key of '1' will find a record with an actual key of:

'1'

(that is, one padded with blanks to the database key length), but it will not find a record with an actual key of:

'1'.

**Note:** db_record_lock must be set to Y or y for db_record_wait to have any effect. Otherwise, db_record_wait will be ignored, that is, it will act as db_record_wait=N.
Get_Screen_Data

Note: You must have the optional Communications Feature to use this action.

Use the Get_Screen_Data action to get information from a specific location on the virtual screen and store the information in a DirectTalk/2 variable.

Parameters

Starting row (Input, Required, Default=none)
Specifies the row on the screen from which to get the information.

Starting column (Input, Required, Default=none)
Specifies the starting column on the screen from which to get the information.

Variable for data (Output, Required, Default=none)
The variable in which to place the information.

Length (Input, Optional, Default=none)
Specifies the length of data to retrieve. You can specify a data length to get information from only part of the simulated screen or the entire screen. If you do not specify a data length, the action retrieves information from the specified starting row and column to the end of the simulated screen.

Return Codes

Data obtained (RC=0)
The action retrieved the information.

No data available (RC=1)
The action found that the host had not yet updated the screen image.

Not connected (RC=2)
The action could not access the screen because there is no session currently active with the host application. This return code can indicate that the host is not currently running.

Session error (RC=3)
The action encountered a problem with the session with the host application. This return code can indicate that the host is not currently running.

Session disconnected (RC=4)
The action could not retrieve the information because the session with the host has been disconnected. This return code can indicate that the host is not currently running.

Comments

You must establish a connection to the host application with the Connect_Screen action prior to using this action.
Get Substring

Use the Get_Substring action to extract a portion of a string and copy the portion into a result variable. You must specify the starting position and length of the substring to extract.

Parameters

**Variable for result (Output, Required, Default=none)**
The variable to receive the result.

**Source string (Input, Required, Default=none)**
The value of the source string. You can specify a variable or a literal. The action only copies the substring you specify. It does not remove the substring from the source string.

**Starting position (Input, Required, Default=’1’)**
The starting position in the source string to begin extracting. The positions are numbered starting with one. You can specify a variable or a literal.

**Length (Input, Required, Default=none)**
The number of characters to extract from the source data. You can specify a variable or a literal. If you specify a length greater than the number of characters available from the starting position, the action returns only those available characters. If you specify a length of ‘0’, the action returns an empty string.

Return Codes

**Done (RC=0)**
The substring was copied into the result variable.

**Error (RC=1)**
An error condition occurred. You may have specified:
- A source variable that did not exist
- Invalid data
- An invalid variable name
- An invalid length

Comments

- You must enclose literals in quotes.
- If the result variable you specify does not exist, the action creates it as a local variable.
- If the length you specify is greater than the number of characters that are available from the starting position, the action returns only the characters that are available.
- The action does not pad or add blanks to the end of the string.
Get_a_Tone

Use the Get_a_Tone action to receive a single key the caller pushed on the telephone key pad. If there is no key tone in the buffer when the action begins, it waits a specified number of seconds for a tone. The voice application receives the digits on a first in, first out (FIFO) basis. The action places the digit into a variable and returns the return code associated with that key.

Return Codes

Key 0 pressed (RC=0)
The caller pressed key 0.

Key 1 pressed (RC=1)
The caller pressed key 1.

Key 2 pressed (RC=2)
The caller pressed key 2.

Key 3 pressed (RC=3)
The caller pressed key 3.

Key 4 pressed (RC=4)
The caller pressed key 4.

Key 5 pressed (RC=5)
The caller pressed key 5.

Key 6 pressed (RC=6)
The caller pressed key 6.

Key 7 pressed (RC=7)
The caller pressed key 7.

Key 8 pressed (RC=8)
The caller pressed key 8.

Key 9 pressed (RC=9)
The caller pressed key 9.

Key * pressed (RC=10)
The caller pressed key *, A, B, C, or D.

Key # pressed (RC=11)
The caller pressed key #.

Time_out (RC=T1)
The caller did not press a telephone key within the time limit. The voice logic module that ran in the last Play_Module action defines the time limit. When the time limit expires, the action returns this return code and sets the timeout_flag system variable to '1'. You can use step -1 to repeat the previous voice logic module that was played.

Last repeat (RC=T2)
The caller did not respond after a number of repeated message prompts. The number of repeats permitted before receiving this return code is defined when you create the voice logic module for the prompt. The caller may have hung up the telephone.

Caller hung up (RC=HUP)
The caller hung up the telephone.
System Variables

last_dtmf_tone (Output)
The action sets this variable when the caller presses a key on the telephone key pad. The value of this variable is the single character of the key that the caller pressed, such as 0 through 9, *, #, A, B, C, or D. The action resets this variable to a null value each time the action runs.

timeout_flag (Output)
The action sets the value of this variable to '1' when the action returns the time out return code. You may want to test this variable in your voice logic modules to offer additional help to callers who do not respond.

Comments

• DirectTalk/2’s automatic buffering of the digits allows you to write voice applications so that callers can fast-path through the input prompts without listening to the menu instructions. Each time the Get_a_Tone action runs, it removes the next character from the buffer. If the buffer is empty, this action waits for the caller to press a key.

• You can use the special step number (-1) with the return code T1, Time_out, to rerun the last Play_Module action. If a caller does not respond, the application can automatically replay the menu of choices to the caller. This will also increment the internal count of the number of times the application has played the menu.

• When a DTMF A, B, C, or D tone is received, it is placed in the last_dtmf_tone variable and the Key '*' pressed return code is returned.

• An example of the Get_a_Tone action is in the sample MENU application.
Get_Tone_String

Use the Get_Tone_String action to receive one or more keys the caller pressed on the telephone key pad. You can specify the minimum number and maximum number of keys, and the termination and escape keys the application will accept from a caller.

Parameters

Minimum tones (Input, Required, Default=’1’)
The minimum number of keys the caller must press, not including the termination character.

Maximum tones (Input, Required, Default=’8’)
The maximum number of keys the caller can press, not including the termination character. The maximum must be equal to or greater than the minimum tones parameter. If the maximum equals the minimum and you did not specify a termination character, the action returns after the caller presses this exact number of keys.

Termination key (Input, Optional, Default=’#’)
The key the caller must press to signify the end of the string of keys. The action places the termination key in the dtmf_term_char system variable. If you specify a termination character, the action does not return a return code until the caller presses the termination key or a time out occurs, regardless of the maximum and minimum tones you specify. If you do not specify a termination character, the action uses an implicit termination when the caller presses the maximum number of keys. You can specify more than one termination character. When the caller presses any one of the termination keys, the action returns the appropriate return code. If the caller presses one of the escape keys, the action places all the characters it received prior to receiving the escape key into the last_dtmf_data system variable.

Escape key(s) (Input, Optional, Default=’’)
The key the caller can press to bypass pressing a string of keys after the action prompts for the keys, for example, if the caller makes a mistake and wants to retry. You can define more than one escape key. If the caller presses one of the escape keys, the action:

- Places all the characters it received prior to receiving the escape key into the last_dtmf_data system variable
- Assigns the escape key the caller pressed to the dtmf_escape_char system variable
- Returns the Escape entered return code

Return Codes

Tones okay (RC=0)
The caller selected a string of keys which satisfy the parameters you specified. The action assigns the keys the caller pressed to the last_dtmf_data system variable. If you specified a termination key and if the caller pressed a termination key, the action assigns the key to the dtmf_term_char system variable.
Too few tones (RC=1)
The caller pressed less than the minimum number of keys and then pressed the termination key. The action returns this return code only if the caller completes a selection with the termination key. The action assigns the keys the caller pressed (up to the point of pressing the termination key) to the last_dtmf_data system variable. The action assigns the termination to the dtmf_term_char system variable. If you did not specify a termination key or if the caller did not press the termination key, the action returns the Time_out return code.

Too many tones (RC=2)
The caller pressed more than the maximum number of keys, followed by the termination key. The action assigns the values to the system variables in the same way as return code 1, Too few tones.

Escape entered (RC=3)
The caller pressed the escape key prior to satisfying the other parameters for this action. The action places all the keys the caller pressed prior to pressing the escape key in the last_dtmf_data global system variable. The action places the escape key the caller pressed in the dtmf_escape_char global system variable.

Time out (RC=T1)
The caller did not satisfy the parameters for the action within the time limit. For example:

- You specified a termination key, but the caller did not press the termination key.
- You did not specify a termination key and the caller did not press enough keys to satisfy the Maximum tones parameter.

The last voice logic module (Play_Module action) that ran defines the time limit. When the time limit expires, the action returns this return code and sets the timeout_flag system variable to 1. The action places all the keys the caller presses prior to the time out in the last_dtmf_data global system variable. You can use step -1 to repeat the previous voice logic module that was played.

Last repeat (RC=T2)
The action prompted (Play_Module action) the caller several times, but the caller did not respond completely. The voice logic module defines the number of times the caller can be prompted before the action returns this return code. The action places all the keys the caller pressed prior to the point this action returned this return code in the last_dtmf_data global system variable. You may want to test for the last_repeat system variable in your voice logic modules to offer additional help to callers who do not respond.

Caller hung up (RC=HUP)
The caller hung up the telephone. The action updates the last_dtmf_data system variable.
System Variables

last_dtmf_data (Output)
The values of all the keys, except the termination or escape keys, the caller pressed up to the point the action returned a return code. Values can be 0 through 9, #, *, and DTMF tones A, B, C, and D. The action resets this variable to a null value each time the action runs.

dtmf_term_char (Output)
The value of the termination key the caller pressed. The action resets this variable to a null value each time the action runs.

dtmf_escape_char (Output)
The value of the escape key the caller pressed. The action resets this variable to a null value each time the action runs.

timeout_flag (Output)
The action sets this variable to '1' when the action returns the Time-out return code.

Comments

• DirectTalk/2 automatically places each of the keys the caller presses into a buffer. The Get_Tone_String action retrieves the keys from the buffer and places the keys in the last_dtmf_data variable. If the buffer is empty or does not contain enough keys to terminate the action, the action will wait for more keys. The application receives the keys on a FIFO basis.

• There may be an instance in the application where, prior to this action, you want to clear the digits in the buffer using the Clear_Tones action. This may be useful when the caller makes an invalid selection and the action will prompt the caller again to press the selection. In addition, some DirectTalk/2 actions, such as Play_Module, detect tones but do not retrieve the keys. You may want to follow those actions with this action once those actions detect keys the caller pressed.

• You can use the special state number (-1) with the return code T1, Time_out, to rerun the last Play_Module action. If a caller does not respond, the application can replay the menu of choices to the caller. This will also increment the internal counter which keeps track of the number of times the application played the menu.

• The DTMF A, B, C, and D tones are treated as any other and may be specified for use by this action.

• An example of the Get_Tone_String action is in the CALC sample application.
Get_Voice_Cont

Note: You must have the optional Voice Recognition Feature and the appropriate voice recognition cards to use this action. See the IBM CallPath DirectTalk/2 Installation Guide.

Use the Get_Voice_Cont action to recognize one or more words spoken by a caller with a given vocabulary set and return the words to the application. You determine which caller responses the action will accept or recognize by choosing one of the available vocabulary sets. This action allows a caller to speak more naturally and does not prompt between words in a string.

Parameters

Minimum words (Input, Required, Default=none)
The minimum number of expected words.

Maximum words (Input, Required, Default=none)
The maximum number of expected words. The maximum number of words must be equal to or greater than the minimum number of words. A lower maximum words value gives a higher accuracy of recognition. For greatest accuracy, keep the difference between the minimum number and maximum number of words as small as possible.

VR subvocabulary (Input, Required, Default=none)
The vocabulary set of valid recognizable word responses. You can specify any vocabulary set defined for a particular language. See the DirectTalk/2 language information manual for the language you are using for the supported voice recognition vocabulary sets.

Starting prompt tone (Input, Optional, Default='y')
Use 'y' to specify a starting tone is played to prompt the user for input. Specify 'n' to omit playing a starting prompt tone.

Return Codes

String okay (RC=0)
The input string of words was recognized. The top two response choices are stored in last_cont_resp1 and last_cont_resp2.

Unrecognized (RC=1)
The continuous voice recognition function was unable to return a recognizable string.

No VR line (RC=2)
There are no voice recognition channels which support continuous voice recognition available.

Parameter error (RC=3)
At least one of the input parameters was not valid.

Spoken too long (RC=4)
The continuous voice response software calculates the maximum time a response can take based on the Maximum words parameter of the action. A response is deemed to have terminated if a period of silence greater than the Final Silence Duration VR configuration parameter is detected. If the response time exceeds the calculated maximum time, this return code is generated.
Time out (RC=T1)
  The first utterance was not received within the time specified by the
  VR_INITSDUR_SILENCE parameter.

Last repeat (RC=T2)
  The action repeated the prompt a specified number of time
  (Play_Module action), but the caller did not respond completely. The
  voice logic module defines the number of repeats that are permitted
  before the action returns this return code. The action places the caller's
  response prior to this return code in the last_cont_resp1 variable.

Caller hung up (RC=HUP)
  The caller hung up the telephone.

System Variables

last_cont_resp1 (Output)
  Contains the string of character values for each word the caller spoke.
  This is the top choice hypothesis as determined by the continuous voice
  recognition function. If you specify a variable-length range voice
  response, the action treats “Stop” as a termination word and does not
  place it in this system variable. The action resets this variable each
time the action is run.

last_cont_resp2 (Output)
  Contains the alternate string of character values for each word the caller
  spoke. This is the second choice hypothesis as determined by the
  continuous voice recognition function. The length of this hypothesis
does not affect the determination of the Too few or Too many return
  codes. If you specify a variable-length range voice response, the action
  treats “Stop” as a termination word and does not place it in this system
  variable. The action resets this variable each time the action is run.

Comments

- The top two returned hypotheses of the continuous voice recognition function
  are stored in the last_cont_resp1 and last_cont_resp2 variables.

- The length of the second returned hypothesis does not affect the Too few or
  Too many return codes.

- The timeout set by a voice logic module in a preceding Play_Module action will
  not affect the timeout processing of the Get_Voice_Cont action. This is
  different from the other voice recognition actions.
Get_Voice_Resp

Note: You must have the optional Voice Recognition Feature to use this action.

Use the Get_Voice_Resp action to recognize one or more words spoken by a caller with a given vocabulary set and return the words to the application. You determine which caller responses the action will accept or recognize by choosing one of the available vocabulary sets. You can also specify a set of escape responses that a caller can use to stop the processing in the Get_Voice_Resp action.

Parameters

Minimum words (Input, Required, Default='1')
The minimum number of words the caller must say.

Maximum words (Input, Required, Default='8')
The maximum number of words the caller can say. The maximum number of words must be equal to or greater than the minimum number of words. If you do not specify that the minimum and maximum number of words are equal, the caller must say "Stop" to indicate the end of each response.

Vocabulary (Input, Required, Default='3')
The vocabulary set of valid recognizable word responses. You can specify any vocabulary set defined for a particular language. See the DirectTalk/2 language information manual for the language you are using for the supported voice recognition vocabulary sets.

Escape string (Input, Optional, Default=' ') 
The string of one or more responses you specify to cause processing to continue with the Escape return code. Use one character for each response. If the caller speaks any of these responses, the action returns immediately and places the corresponding character in the resp_escape_char variable. The action places all the responses the caller spoke before the action processed the Escape return code in the last_voice_resp variable.

Return Codes

Response okay (RC=0)
The caller spoke the words that satisfy the parameters you specified for the action. The action assigns the letters that correspond to each word the caller spoke to the last_voice_resp system variable.

Too few words (RC=1)
The caller spoke less than the minimum number of words, followed by "Stop." The action assigns the characters for the words that the caller spoke up to this point to the last_voice_resp system variable. The action only returns this return code if the caller said "Stop" after the action asked for a variable-length response. The action returns the return code T1, Time out, when the action asks for a variable-length response and the caller does not say "Stop" at the end.
Too many words (RC=2)
The caller spoke more words, followed by "Stop" than the maximum number of words allowed. The action assigns the characters for the words that the caller spoke up to this point to the last_voice_resp system variable.

Unknown Response (RC=3)
The word the caller spoke was not part of the vocabulary list or the action could not recognize the word. The action assigns a character to the resp_term_char system variable to specify the reason why it terminated the caller’s response. See the resp_term_char system variable for the list of characters the action assigns.

No VR Line (RC=4)
This could occur if the application is running on a voice line not configured for use with voice recognition, or if there are fewer voice recognition lines than telephone lines and all the voice recognition lines are in use.

Escape (RC=5)
The caller’s response matches one of the responses you specified in the Escape string parameter. The action stores the corresponding character in the resp_escape_char variable. The action stores all the other characters in the last_voice_resp variable.

Time out (RC=T1)
The caller did not satisfy the parameters for this action within the time limit. For example, if the action asks for a variable-length response and the caller does not say "Stop", the action returns this return code. The last voice logic module (Play_Module) that ran defines the time limit. When the time limit expires, the action returns this return code and sets the timeout_flag system variable to 1. The action assigns the characters for the words that the caller spoke up to this point to the last_voice_resp system variable.

Last repeat (RC=T2)
The action repeated the prompt a specified number of times (Play_Module action), but the caller did not respond completely. The voice logic module defines the number of repeats that are permitted before the action returns this return code. The action places the caller’s response prior to this return code in the last_voice_resp global system variable.

Caller Hung up (RC=HUP)
The caller hung up the phone.

System Variables

last_voice_resp (Output)
Contains the character value for each word the caller spoke. If you specify a variable length range voice response, the action treats "Stop" as a termination word and does not place it in this system variable. The action resets this variable each time the action is run.

resp_term_char (Output)
Contains the character value indicating the reason why the input was terminated. The characters are:
The caller said "Stop."

'T' The caller said "Stop."

'G' The caller spoke the response too softly and the word is detected as a gap.

'S' The caller spoke the words too soon or before the prompt.

'U' The action did not recognize the word the caller spoke.

resp_escape_char (Output)
The response that caused the action to return the Escape return code.

timeout_flag (Output)
The action sets this variable to 1 when it returns the T1 return code, Time out.

Comments

• See the DirectTalk/2 language information manual for the language you are using for supported voice recognition vocabulary sets.

• Table 18 contains the characters the action returns when it detects an error. If the action detects any of these conditions, it returns immediately with the Unknown Response return code. The resp_term_char variable contains the character indicating the type of error.

<table>
<thead>
<tr>
<th>Character</th>
<th>Error Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>The caller spoke the response too softly.</td>
</tr>
<tr>
<td>S</td>
<td>The caller spoke the words too soon or before the prompt.</td>
</tr>
<tr>
<td>U</td>
<td>The action did not recognize the word the caller spoke.</td>
</tr>
</tbody>
</table>

• You can use the Escape string parameter to allow the caller to get help and terminate a response using "Cancel" in addition to using "Stop":
  – If you want to offer the caller the option of getting help at any time by saying "Help," specify H in the Escape string parameter and use a vocabulary that contains the word "help".
  – If you want to offer the option of ending before the required number of responses by saying "Cancel", specify C in the Escape string parameter and use a vocabulary that contains the word "cancel".
  – If you want to offer both options to the caller, specify HC in the Escape string parameter and use a vocabulary that contains the words "help" and "cancel". Check the resp_escape_char variable to provide the appropriate response.

• The action plays a beep to prompt the caller for a response. The caller can only speak one word after the beep. If you specify that the action is to request more than one response from a caller, the caller will hear a beep for each response. The action expects the caller to say "Stop" to indicate the end of words if the value of the Minimum words parameter is less than the value of the Maximum words parameter. Otherwise, the action will return after the caller has spoken the number of responses you specified (the value of the Minimum words parameter equals the value of the Maximum words parameter).
- You can use the special step number (-1) with the Time out return code. This step takes the application to the last Play_Module action run prior to this action. The action can then replay a prompt to the caller when the caller does not respond completely.
- An example of the Get_Voice_Resp action is in the sample MENU application.
Hang_up_Phone

Use the Hang_up_Phone action to change the hook state of the line. This can hang up the telephone and end the incoming or outgoing call. This can also keep a line busy (offhook) while the application is unavailable owing to host processing or other reasons.

Parameters

Offhook (Input, Optional, Default='no')

Specify whether to leave the telephone offhook after disconnecting the call. Specify 'no' to indicate the telephone should be placed onhook (ready to answer the next call). Specify 'yes' to indicate that telephone should remain offhook (the telephone line remains busy).

Return Code

Phone hung up (RC=0)

The action ended the telephone line connection.

Comments

- You should use this action prior to looping back to the Wait_for_Call or Place_A_Call actions that establishes a telephone connection.

- If you specify Offhook='yes', use another Hang_up_Phone action with Offhook='No' to put the line onhook (ready to answer another call) before looping back to the Wait_for_Call action. If Offhook='yes', the line will be placed on hook for up to two seconds before it goes back off hook. This option is therefore not recommended if you are using an ACD or hunt group.

- An example of the Hang_up_Phone action is in the sample MENU application.

- Very few E1 CAS protocols allow a channel to remain offhook when there is no call established. If Offhook is set to 'yes', the action will attempt to put an E1 CAS channel into that state but may not be able to do so, and no indication of the failure is reported. You should check the operation of an E1 CAS trunk before relying on the operation of this parameter.
Link_to_Appl

Use the Link_to_Appl action to link to and run a subapplication. The subapplication uses its own local variable pool and shares variables in the global variable pool. The subapplication also uses its own application startup parameters. Use global variables to share information between a voice program and a subprogram. The subprogram begins running at the first statement of the application unless you specify otherwise using the Initial step parameter.

If a setup file exists (<subapplication_name>.SU), DirectTalk/2 will automatically create and set variables in the file as local variables when the subapplication is loaded and started.

Parameters

Application (Output, Required, Default=none)
The name of the program to link to.

Initial step (Input, Optional, Default=none)
The step in the subapplication at which to perform the link.

Return Codes

Return code 0 (RC=0)
The subapplication returned a '0'.

Return code 1 (RC=1)
The subapplication returned a '1'.

Return code 2 (RC=2)
The subapplication returned a '2'.

Return code 3 (RC=3)
The subapplication returned a '3'.

Return code 4 (RC=4)
The subapplication returned a '4'.

Return code 5 (RC=5)
The subapplication returned a '5'.

Return code 6 (RC=6)
The subapplication returned a '6'.

Return code 7 (RC=7)
The subapplication returned a '7'.

Return code 8 (RC=8)
The subapplication returned an '8'.

Return code 9 (RC=9)
The subapplication returned a '9'.

Return code 10 (RC=10)
The subapplication returned a '10'.

Return code 11 (RC=11)
The subapplication returned an '11'.

Return code 12 (RC=T1)
The subapplication returned a '12'.
Return code 13 (RC=T2)
The subapplication returned a '13'.

Return code 14 (RC=HUP)
The subapplication returned a '14'.

System Variables

application_name (Output)
Set to the name of the current voice application or the name of the subapplication.

linked_from_appl (Output)
Keeps the name of the voice application prior to the link.

Comments

• You can develop subapplications in the same manner as other applications. Subapplications, however, do not normally use the Wait_for_Call action.

• You only have to specify values for the return codes that can be returned by the subapplication. You must specify a value for at least one return code. If a subapplication returns an undefined return code, the application will halt.

• Use global variables to pass parameters to subapplications.

• Subprograms are similar to subroutines in a computer program. For example, you can define common processing once and you can invoke a subapplication at any point in one or more applications.

• Subapplications are not user-created actions. See the IBM CallPath DirectTalk/2 Application Programmer’s Guide for more information on creating your own actions.

• An example of the Link_to_Appl action is in the sample MENU application.
Log_a_Message

Use the Log_a_Message action to write a user-defined message to the Application Session log file. Each message can contain a single substitution value that you can use to include specific information for the application, such as noting the step number of an action with an unexpected error.

Parameters

Message ID (Input, Required, Default=none)
The message identifier defined in the user message file. The message identifier can be up to 8 characters in length.

Replacement text (Input, Optional, Default=' ')
The text you want to replace the substitution character with. The message text can be up to 128 bytes.

Return Codes

Message logged (RC=0)
The action logged the message.

Error (RC=1)
An error condition occurred. The action could not write the message.

Comments

The Log_a_Message action:

- Searches the user message file (USRLOGMS.ERE) for the message ID you specify
- Retrieves the message
- Substitutes the text you specify (indicated by %s)
- Writes the result to the message log file (<APPLICATION SESSION NAME>.LOG)

The format of the source message in the user message file is:

message_id <------ message_text ------>

where:

- Message_id is the message identifier you specify to search on, up to 8 characters long.
- Message_text is the text of the message, up to 128 characters long. The text can contain only one % to indicate where to place the substitution text. Do not use a % in the message text other than as a single substitution indication.
- Note that the file USRLOGMS.ERE is an ASCII flat file, not a DirectTalk/2 database. For example, a small USRLOGMS.ERE file might look like this:

  KFS0001 System started at %s
  KFS0002 Error detected at step %s
  ERE0001 Customer number %s called
Log_Statistics

Use the Log_Statistics action to write the statistics record you define to the DirectTalk/2 statistics file. You can log a maximum of 32 variables (statistics items). You can optionally include a set of call detail statistics that DirectTalk/2 maintains in the statistics.

Return Codes

Logged (RC=0)
The action logged the statistics record using the current values for each of the statistics items.

Error (RC=1)
This code is returned if the action encounters any error that prevents the statistics record from being successfully logged.

System Variables

The action automatically logs the following call detail statistics in addition to any statistics you have defined if you specify a value of “Y” for the cs_record variable.

These values are global variables. DirectTalk/2 automatically creates and maintains these global variables except where noted below. Your application can update these values if necessary. DirectTalk/2 does not create these variables until it runs the first Wait_for_Call or Place_a_Call action.

cs_calltype
The type of call the action processed. The type can be:

I  Inbound
O  Outbound

cs_calltime
The time the application answered the inbound call or the time the application dialed the outbound call.

cs_calldate
The date the application answered the inbound call or the date the application dialed the outbound call.

cs_duration
The length of the call in seconds.

cs_number
The string the action dialed for an outbound call. This variable is set to null for an inbound call.

cs_applname
The name of the application. The application name is set when the first Wait_for_Call or Place_a_Call action runs. This value does not change if you link to a subapplication.

cs_hangstep
The step where the application detected that the caller hung up. If the application detected the hang up in a linked application, this step is in the linked application.
cs_lineno

The telephone physical port number on which the application placed or received the call.

cs_termtype

The type of termination, namely, how the call ended. DirectTalk/2 detects when the caller or the application hangs up and updates this field. If there are other ways that the application can end, such as transferring to an operator, the application must set the value for this variable.

A  The application hung up. DirectTalk/2 sets this value.
C  The caller hung up. DirectTalk/2 sets this value.
T  The application transferred the call. Your DirectTalk/2 application must set this value. You must use this value if you want to include information about transfers on the standard reports.

cs_procedure

A string the application uses to identify the processing being performed when the caller hung up. Your application can set this variable at the beginning of a group of steps (procedure) that performs a particular function. If a caller hangs up while in that group of steps, the action logs the name you specified. You can summarize this information using the sample termination report. You must make sure that the application sets this variable if you intend to create the termination report. This value can contain any characters including blanks, but must include at least one non-blank character.

Comments

- You can use these call detail statistics with the report program to create a set of standard reports about your DirectTalk/2 applications. See the DirectTalk/2 Administrator’s Guide for more information. You can use the Add_Stat_Item action to define your own statistics to save with the call detail statistics. See “Add_Stat_Item” on page 161 for more information.
- If you are using the call detail statistics that DirectTalk/2 will create and maintain, but sure to do the following:
  - Issue the Log_Statistics command only once per call.
  - Issue the Log_Statistics action after performing the Hang_up_Phone action.
  This will ensure that the statistics records are complete and that reports created using STATRPT.CMD will be accurate. Otherwise, results can be unpredictable.
- The action determines the length of the record in the statistics file by the number of statistics items. An example of such a statistics record is:
  APPLSTAT VMSDBU388 amach19900307103706 cs_calltype(I) cs_calltime(47)
- Records consist of a header and the data. Table 19 on page 227 shows the format of each record.
For call detail statistics purposes, a call starts either when a \texttt{Wait\_for\_Call} action answers the phone or when a \texttt{Place\_a\_Call} action detects an Answer return code. The call ends with a \texttt{Hang\_up\_Phone} action. Run the \texttt{Hang\_up\_Phone} action immediately after DirectTalk/2 detects a caller Hang-up return code to ensure that the call detail statistics are accurate. The \texttt{Wait\_for\_Call} action will reset all call detail statistics variables. The \texttt{Place\_a\_Call} action will reset all call detail statistics variables only if it has not been preceded by a \texttt{Wait\_for\_Call} action, or if the \texttt{cs\_calltype} variable is not set to I. To collect information about the calls for standard reports, do the following:

1. Use the \texttt{Set\_Variable} action to assign a value of “Y” to the \texttt{cs\_record} variable (you do not need to create this variable). You should assign this value once at the beginning of your application.

2. If you want to break down the Termination Report by procedure, use the \texttt{Set\_Variable} action to assign a descriptive string to the \texttt{cs\_procedure} variable at the beginning of each group of steps that perform a major function in your application.

For example, in a banking application that allows account queries and transfers, you can set the \texttt{cs\_procedure} to Verify Acct before the caller specifies his or her account number and personal identification number (PIN). Then you can set the \texttt{cs\_procedure} variable to Acct Query or Acct Transfer at the beginning of the processing for the query and transfer functions. If the caller hangs up during an account transfer, the action will log the value of Acct Transfer for the \texttt{cs\_procedure} variable in the statistics file. For example:

\begin{verbatim}
cs\_procedure(Acct Transfer)
\end{verbatim}
3. Since the Place_a_Call action is typically used to produce a transfer as part of an inbound call, the Place_a_Call action will not reset the call detail statistics or be considered a separate call in all cases. To ensure that the DirectTalk/2 records the Place_a_Call as a separate call, set the cs_calltype variable to null (" ") with a Set_Variable action just before the Place_a_Call.

4. The Place_a_Call action updates all statistics variables only if the 'answered' return code (RC=0) is returned.
Pause

Use the Pause action to specify the number of seconds to suspend the running of the application.

Parameter

Number of seconds (Input, Required, Default=none)
The number of seconds to suspend the application.

Return Code

Pause complete (RC=0)
The application was suspended for the number of seconds you specified.
Place_a_Call

Use the Place_a_Call action to dial an outgoing call with call standard or enhanced call progress analysis. DirectTalk/2 uses call progress analysis to report the status of the call to the voice program.

You specify the type of call progress analysis using the Type of Call Progress Network Interface (NIF) channel parameter during installation configuration (see the IBM CallPath DirectTalk/2 Installation Guide).

Parameters

Phone Number (Input, Required, Default=none)

The telephone number and the switch feature characters the action is to dial.

The following characters are supported in the Phone Number parameter:

I causes the dialing to wait for the international dial tone as defined in the Telephony Server configuration. Use this only when Enhanced call progress analysis is enabled. You can place the character within the number to cause a wait for detection.

L causes the dialing to wait for the local dial tone as defined in the Telephony Server configuration. Use this only when Enhanced call progress analysis is enabled. You can place the character within the number to cause a wait for detection.

M causes the dialing to proceed using MF tone dialing.

N causes the dialing to test for dial tone by waiting for a period of nonsilence. The nonsilence time is defined in the Telephony Server configuration. This is supported only as the first character in a number.

P causes the dialing to proceed using pulse dialing.

T causes the dialing to proceed using DTMF tone dialing.

W causes the dialing to test for dial tone by waiting for a period of time. The wait time is wait for a period of time as defined in the Telephony Server configuration (this is a nonpositive detection method). This is supported only as the first character in a number.

X causes the dialing to wait for the extra dial tone as defined in the Telephony Server configuration. Use this only when Enhanced call progress analysis is enabled. You can place the character within the number to cause a wait for detection.

& Causes a hook flash (temporary onhook).

, Causes a 2-second pause in the dialing sequence.

Note: The default method of dial tone detection is L and you specify it using the Dial Tone Detection Type parameter during installation and configuration. If you enable only Standard call progress analysis, this equates to N for analog lines and W for the following types of T1 lines: FXS Loop, SAS Loop, and E+M Immediate. You cannot configure dial tone detection for E1, and the following T1 lines: FXS Ground and E+M Wink.
Return Codes

Answered (RC=0)
The party the action called answered the telephone and is on the line.

Busy (RC=1)
The action received a busy signal after dialing the telephone number.

No Answer (RC=2)
No one answered the telephone after a specified number of rings (Standard) or time (Enhanced). The number of rings and time are set in the installation configuration.

No ring (RC=3)
The telephone did not ring. This may indicate a communication problem with the telephone switch, a cabling problem, or the telephone number is invalid.

Operator intercept (RC=4)
The action received three tones indicating that the number is no longer in service or the number has changed. You may have specified an invalid telephone number exchange.

Answering machine (RC=5)
The action used the Positive Answer Machine Detection (PAMD) method (supplied by Dialogic) or the Timed Answer Machine Detection (TAMD) method (based on the time of the initial greeting after the telephone was answered) to detect an answering machine. You set the period of time in the Telephony Server configuration. The action does not always return this return code in every case where an answering machine answers the telephone.

No dial tone (RC=6)
The action did not detect a dial tone, and therefore did not attempt to place a call.

Application quiesce (RC=7)
A request to stop the application has been received.

FAX/Modem (RC=8)
A FAX or modem has been detected. You need to use Enhanced call progress analysis to detect a FAX/Modem.

Caller hung up (RC=HUP)
If the action returns this return code as part of transferring a call, then this return code indicates the caller hung up.

Comments

- You can use the special characters ' & ' and ',' to perform switch-dependent features, such as transferring a call. For example, you can transfer the caller by placing a call to '&,*7xxxx' where 'xxxx' is the extension of the transfer and '&7' is the transfer feature code for a ROLM CBX. The feature codes differ depending on what type of switch is being used. In this case, the voice system notifies the application if the person the party is transferred to is not there so that the application can give the called party other options.

- The Place_a_Call action tests the stop_application flag. If a quiesce is indicated by the flag, the Place_a_Call action immediately exits with the Application Stop return code. If a quiesce is not indicated by the flag, the
Place a Call action continues processing. In addition, if you execute a Place a Call action within another call, the stop_application flag is ignored. A Place a Call action is considered to be within another call if a Wait_for_Call action or a Place a Call action has already been successfully executed (that is, a caller is already on the line) and a Hang-up action has not been executed.

- If you want your voice application to dial numbers without any indication of whether the telephone is answered, use the Put_Tone_String action.

- This action takes the line offhook, if it isn’t already, before dialing. Also, it does not put the line onhook (hang up) after it has completed regardless of the return code taken; the application maintains control of the hook state.
Play_Module

Use the Play_Module action to run a voice logic module to conditionally play voice segments to callers. The action can detect when the caller has pressed a key on the telephone key pad. This gives the caller the ability to fast-path through the voice menu to a known point in the voice application.

Parameters

Voice module name (Input, Required, Default=none)
The name of the voice logic module to run. The name can be up to 15 characters in length. You must create the voice logic module prior to running this action. You can specify the voice logic module name as a literal or variable. If you specify the name as a literal, you must enclose the name in single quotes.

Force play (Input, Optional, Default='no')
Specifies whether the caller must listen to the entire voice segment or can interrupt the voice segment by pressing a key on the telephone. Use 'no' to specify that the caller can interrupt the voice segment by pressing a key on the telephone key pad. Use 'yes' to specify that the caller must listen to the entire voice segment before the application will process any keys the caller presses.

VR stop vocabulary (Input, optional, default='0')
Specifies whether word detection or voice detection are enabled. Use '-1' to enable voice stop, that is, to interrupt the play function following the utterance of a sound by the caller. Use a subvocabulary number to enable word detection, that is, to interrupt the play function and attempt to resolve the utterance into a word.

Note: The VR stop vocabulary cannot be used with continuous voice recognition.

Return Codes

Play complete (RC=0)
The voice logic module ran to completion.

Key detected (RC=1)
If you set the Force play parameter to 'yes', this return code indicates:
- The action runs the voice logic module to completion.
- There is also a key the voice program must read from the buffer.

If you set the Force play parameter to 'no', the action interrupts the voice logic module and returns this return code to indicate that there is a key to be read from the key buffer. This allows the caller to fast-path through the voice application. When this action starts, it:
- Returns this return code, if there are any existing keys currently in the key buffer
- Does not play the voice logic module

For example, the caller pressed a key before the Play_Module action began and the action immediately returns the Key detected return code without playing the voice logic module.
Word detected (RC=2)
During a Play_Module action with word detection enabled, the caller has used voice cutthrough.

Voice detected (RC=3)
During a Play_Module action with voice detection enabled, the caller has used voice stop.

Unknown response (RC=4)
Word detection was not able to recognize the caller's utterance.

No VR line (RC=5)
This could occur if the application is running on a voice line not configured for use with voice recognition, or if there are fewer voice recognition lines than telephone lines and all the voice recognition lines are in use.

No TTS line (RC=6)
This could occur if the application is running on a voice line not configured for text-to-speech or if there are fewer text-to-speech lines than telephone lines and all the text-to-speech lines are in use.

Caller hung up (RC=HUP)
The caller hung up the phone.

System Variables

timeout_value (Output)
The action sets the value of this variable to the time-out value specified in the voice logic module that ran.

last_repeat (Output)
The action sets the last repeat to '1' to indicate that a voice logic module is being played for the last time and this is the last chance for the caller to press a key. Otherwise, the action sets the last repeat to '0'.

resp_term_char (Output)
This variable is set if word detection is enabled and if the caller's response was 'stop' or is unrecognized.

If RC=2 (word detected), resp_term_char is set to 'T' if the word 'stop' is detected.

If RC=4 (unknown response), resp_term_char contains one of the following characters to indicate the reason for the unknown response:

- G=Spoke too soon
- S=Too soft
- U=Unknown

last_voice_resp (Output)
Contains the caller's response if word detection is enabled.

If RC=2 (word detected), last_voice_resp contains the character corresponding to the recognized word:

If RC=4, the last_voice_resp contains one of the following characters to indicate the reason for the unknown response:

- G=Spoke too soon
- S=Too soft
timeout_flag (Output)
The action resets this variable when a different voice logic module is played.

Comments

- Any tones that exist in the buffer prior to the start of the Play_Module action cause the action to return the return code 1, key detected. If the Force play parameter is set to 'no', the action will return this code immediately—no voice module will be played.
- You should decide whether to use the Clear_Tones action to clear any tones from the buffer prior to running this action. For example, if you want the caller to be able to fast path through the application, then you should not perform the Clear_Tones action.
- You can use the last_repeat and timeout_flag variables with the IF function when creating a voice logic module to allow conditional playing of a voice segment.
- An example of a Play_Module action is in the CALC sample application.
Play_Text

Note: You must have the optional Speech Synthesis Feature to use this action.

Use the Play_Text action to synthesize voice from records that are in a DirectTalk/2 text database. The action plays the message to completion unless the caller presses any telephone key or hangs up the telephone.

Parameters

Database server (Input, Required, Default=none)
The network name of the Database Server.

File name (Input, Required, Default=none)
The name of the database file which contains the text records. Use the same OS/2 file name that was used to create the database using the database create utility. When the text records database is created, the record length is 4096 and the key length is 19.

Record key (Input, Required, Default=none)
The key to use to retrieve the text record. The length of the key can be up to 15 characters.

Return Codes

Play done (Return Code=0)
The action played the complete text record.

Key pressed (Return Code=1)
The caller pressed a key on the telephone key pad and interrupted the text record. The key the caller pressed remains in the key buffer. You should:

- Read the key using the Get_a_Tone or Get_Tone_String actions
- or
- Discard the key using the Clear_Tones action.

No text line (Return Code=2)
There is currently no text-to-speech facility available.

Error (Return Code=3)
There was a problem retrieving the record from the database. For example, the record does not exist, or the Database Server is inaccessible.

Caller hung up (Return Code=HUP)
The caller hung up the telephone. The action stopped playing the text record.

Comments

- Do not use this action to modify the exception dictionary; that is, do not use the ~x] reset.
Play_Text_String

**Note:** You must have the optional Speech Synthesis Feature to use this action.

Use the Play_Text_String action to synthesize voice from data contained in a DirectTalk/2 variable. The action plays the message to completion unless the caller presses any telephone key or hangs up the telephone.

**Parameter**

- **Variable name or literal (Input, Required, Default=none)**
  
  The string to be synthesized.

**Return codes**

- **Play done (RC=0)**
  
  The action played the complete text.

- **Key pressed (RC=1)**
  
  The caller pressed a key on the telephone key pad and interrupted the text. The key the caller pressed remains in the key buffer. You should:
  
  - Read the key using the Get_a_Tone or Get_Tone_String actions
  
  or
  
  - Discard the key using the Clear_Tones action.

- **No text line (RC=2)**
  
  There is currently no text-to-speech facility available.

- **Caller hung up (RC=HUP)**
  
  The caller hung up the telephone. The action stopped playing the text.

**Comments**

- Do not use this action to modify the exception dictionary; that is, do not use the `\^[x]` reset.
Play_Voice

Use the Play_Voice action to play voice records that were recorded from callers and saved in a database using the Record_Voice action. The action plays the voice record to completion unless the caller presses any telephone key or hangs up the telephone.

If you have the Voice Cut Through feature installed (see “Voice Cutthrough” on page 111) the the voice record can also be interrupted by the caller’s voice.

Parameters

**Database server (Input, Required, Default=none)**
The network name of the database server.

**File name (Input, Required, Default=none)**
The name of the database file which contains the voice records. Use the same OS/2 file name that was used when the database was created. When the voice records database is created, the record length is 4096 and the key length is 19.

**Record key (Input, Required, Default=none)**
The key used to retrieve the voice record. The length of the key can be up to 15 characters.

**VR stop vocabulary (Input, optional, default='0')**
Specifies whether word detection or voice detection are enabled. Use '-1' to enable voice detection, that is, to interrupt the voice input/output function following the utterance of a sound by the caller. Use a subvocabulary number to enable word detection, that is, to interrupt the voice input/output function and attempt to resolve the utterance into a word.

Return Codes

**Play done (RC=0)**
The action played the complete voice record.

**Key pressed (RC=1)**
The caller pressed a key on the telephone key pad and interrupted the voice segment. The key the caller pressed remains in the key buffer. You should:
- Read the key using the Get_a_Tone or Get_Tone_String actions.
- Discard the key using the Clear_Tones action.

**Error (RC=2)**
There was a problem retrieving the record from the database. For example, the record does not exist or the database server is inaccessible.

**Word detected (RC=3)**
During a Play_Module action with word detection enabled, the caller has used voice stop.

**Voice detected (RC=4)**
During a Play_Module action with voice detection enabled, the caller has used voice cutthrough.
No VR line (RC=5)
This could occur if the application is running on a voice line not configured for use with voice recognition, or if there are fewer voice recognition lines than telephone lines and all the voice recognition lines are in use.

Unknown response (RC=6)
Word detection was not able to recognize the caller’s utterance.

Caller hung up (RC=HUP)
The caller hung up the telephone. The action stopped playing the voice record.

System Variables

resp_term_char (Output)
This variable is set if word detection is enabled and if the caller says the word ‘stop’ or gives a response that is not recognized.

If RC=3 (word detected), resp_term_char is set to ‘T’ if the word ‘stop’ is detected.

If RC=6 (Unknown response), resp_term_char indicates the reason for the unknown response:

- G=Spoke too soon
- S=Spoke too softly
- U=Reason unknown

last_voice_resp (Output)
This variable contains the caller’s response if word detection is enabled.

If RC=3 (word detected), resp_term_resp contains the recognized character or digit, or one of the following characters which correspond to the word spoken:

- T=Stop
- C=Cancel
- H=Help

If RC=6 (Unknown response), resp_term_char indicates the reason for the unknown response:

- G=Spoke too soon
- S=Spoke too softly
- U=Reason unknown

Comments

- Use this action with the Record_Voice action to play a recorded voice.
- Use the Delete_Voice action to remove any voice records from the database.
- This action is not normally used to play voice segments you record using the voice segment editor, such as prompts, menus, or greetings.
- The Play_Voice action is not part of the Voice Messaging actions provided by DirectTalk/2.
Put_Tone_String

Use the Put_Tone_String action to execute transfer without call progress monitoring (blind transfer) or to issue a command to a switch.

Parameter

Tone string (Input, Required, Default=none)
The telephone number and the switch feature characters the action is to dial.

Return Codes

String dialed (RC=0)
The action dialed the tone string you specified.

Caller hung up (RC=HUP)
The action detected the telephone was hung up.

Comments

- Other than the digits 0–9, #, and *, you can use two special characters with this action: ‘&’ (Hook Flash) and ‘,’ (pause for switch-processing). You can also use DTMF tones A, B, C, and D.
- You can use these special characters to perform switch-dependent features, such as transferring a call to an extension that is always answered. For example, you can transfer the caller by placing a call to ‘&,*7xxxx’, where ‘xxxx’ is the extension of the transfer and ‘*7’ is the transfer feature code for a ROLM CBX. The feature codes would differ depending on the type of switch being used.
- If you want your voice application to dial numbers with an indication of whether the telephone is answered, use the Place_a_Call action.
- This action does not change hook state before or after dialing the number. If you wish to place an outgoing call without Call Progress Monitoring, use the Hang_up_Phone action to take the line offhook prior to using Put_Tone_String.
Put_User_Info

Use the Put_User_Info action to change information in the directory such as the password and paging information. You specify the information you want to change in variables. These variables consist of a prefix you must specify and a predetermined suffix. Use an action parameter to set the prefix. The Put_User_Info action uses the directory identification value and name prefix you specify to update the information in the directory entry.

You must have the optional Voice Messaging Feature to use this action.

Parameters

Directory ID (Input, Required, Default=none)
The unique identifier for the directory entry to be updated. Use the value in the <prefix>_idno variable from the Search_Directory action.

Variable prefix (Input, Optional, Default='direct')
The variable prefix for the variables you use to update the directory information. You can use the same prefix as the one you use in the Search_Directory action.

Return Codes

Updated (RC=0)
The action updated the information in the directory.

Data error (RC=1)
You did not specify a value for one or more variables.

Error (RC=2)
The action could not access the directory.

User Variables

<prefix>_pswd
The password for the directory entry.

<prefix>_pagetype Required, (Default=none)
The paging type. You can specify 'V', 'D', or ' ' (no paging).

<prefix>_pagephon
Paging phone number (must be all numeric and is required if the <prefix>_pagetype variable is 'V' or 'D').

<prefix>_pagedata
Digital paging data (must be all numeric and is required if the <prefix>_pagetype variable is 'D').

<prefix>_rtime
The maximum recording time for a message.

System Variable

directory_server (input)
The name of the remote server on which the directory is held.
Comments

- The voice application passes the values of the variables in the Search_Directory action to the variables in the Put_User_Info action.

- You can use the variables returned by the Search_Directory action as the variables in the Put_User_Info action.

- The only directory entries in the database which you can change using the Put_User_Info action are the password, paging type, pagephon, pagedata, and recording time. These user variables must exist, even if only set to null values.
Receive_TDD

Use the Receive_TDD action to receive TDD data from a caller. The action will convert the received TDD data into an ASCII text string and store it in the variable specified by the action. The action always stops on receipt of a DTMF digit.

For any application using Receive_TDD, permanently assign a TDD resource to the application for the duration of each call by using the Assign_Resource and Free_Resource actions. Assign this resource on a "per call" basis to ensure that the TDD receive shift state is correctly initialized and maintained throughout the call.

Note: Data can be received only while this action is active. Any data transmitted while the action is not active is lost.

Parameters

TDD text variable (Output, Required, Default=none)
The variable to store the ASCII text. If the variable does not exist, it is created.

Duration (Input, Optional, Default='10')
Amount of time, in seconds, to wait for incoming data.

Maximum characters (Input, Required, Default='40')
The maximum amount of characters a caller can send.

Termination characters (Input, Optional, Default=none)
Characters a caller can send to signify the end of the data they are sending.

Return Codes

Stop on duration (RC=0)
TDD data was received and converted in the duration period. The converted data was stored in the specified variable.

Stop on max char (RC=1)
The maximum number of TDD characters was received and converted. The converted data was stored in the specified variable.

Stop on term char (RC=2)
The caller has sent a termination character to signify the end of the TDD data. The data was converted and stored in the specified variable. The termination character was stored in the specified variable and in the variable tdd_term_char.

Key pressed (RC=3)
This indicates the receipt of a DTMF digit. This means that the incoming TDD data was interrupted and may not be complete. The caller has probably interrupted the incoming TDD data by pressing a key on the phone pad. The data received was converted and stored in the specified variable.

Error (RC=4)
An error condition occurred. You may not have specified the TDD text variable parameter.

Caller hung up (RC=HUP)
The caller hung up the telephone.
The DirectTalk/2 TDD implementation is through software—there is no hardware assist to generate or decode the “modem” character signals.

Basic TDD transmission signaling is used. This is half duplex at approximately 7 characters per second. There is no handshaking to determine that the remote device is a TDD terminal.

There is considerable overhead when receiving TDD data. It is suggested that you use high performance machines and that no more than two lines are receiving TDD data simultaneously. The overhead incurred when transmitting TDD data is not so significant.

In addition to the actions described in this chapter, see also the Assign_Resource and Free_Resource actions described in “Telephony Actions” on page 153.

The TDD data received is assumed to be BAUDOT.

Table 20 is the ASCII/BAUDOT conversion table used by DirectTalk/2.

Any characters not found in Table 20 will be converted as blanks.

The LF (linefeed) character will be ignored and the CR (carriage return) character will be returned as a <.

The variable specified to store the ASCII text will store as much data as was received before termination. This means that although the receipt of data may have been terminated due to an error, the variable may contain partial data.

If the Key pressed return code is returned, the DTMF digit can be retrieved using Get_a_Tone or Get_Tone_String.

<table>
<thead>
<tr>
<th>Table 20 (Page 1 of 4). TDD Code Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Letters</td>
</tr>
<tr>
<td>BKSP</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>LF</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>SPACE</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>s</td>
</tr>
<tr>
<td>I</td>
</tr>
<tr>
<td>i</td>
</tr>
<tr>
<td>U</td>
</tr>
<tr>
<td>u</td>
</tr>
<tr>
<td>CR</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>r</td>
</tr>
<tr>
<td>Character</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>J</td>
</tr>
<tr>
<td>j</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>n</td>
</tr>
<tr>
<td>F</td>
</tr>
<tr>
<td>f</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>K</td>
</tr>
<tr>
<td>k</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>t</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td>z</td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td>l</td>
</tr>
<tr>
<td>W</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>h</td>
</tr>
<tr>
<td>Y</td>
</tr>
<tr>
<td>y</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>p</td>
</tr>
<tr>
<td>Q</td>
</tr>
<tr>
<td>q</td>
</tr>
<tr>
<td>O</td>
</tr>
<tr>
<td>o</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>G</td>
</tr>
<tr>
<td>g</td>
</tr>
<tr>
<td>FIGURES</td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td>m</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>v</td>
</tr>
<tr>
<td>LETTERS</td>
</tr>
</tbody>
</table>

**Figures**

<p>| BKSP   | 00000 |
| 3      | 00001 |</p>
<table>
<thead>
<tr>
<th>Character</th>
<th>BAUDOT¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF</td>
<td>00010²</td>
</tr>
<tr>
<td>-</td>
<td>00011</td>
</tr>
<tr>
<td>SPACE</td>
<td>00100</td>
</tr>
<tr>
<td>8</td>
<td>00110</td>
</tr>
<tr>
<td>7</td>
<td>00111</td>
</tr>
<tr>
<td>CR</td>
<td>01000³</td>
</tr>
<tr>
<td>$</td>
<td>01001</td>
</tr>
<tr>
<td>4</td>
<td>01010</td>
</tr>
<tr>
<td>'</td>
<td>01011</td>
</tr>
<tr>
<td>.</td>
<td>01100</td>
</tr>
<tr>
<td>!</td>
<td>01101</td>
</tr>
<tr>
<td>:</td>
<td>01110</td>
</tr>
<tr>
<td>(</td>
<td>01111</td>
</tr>
<tr>
<td>5</td>
<td>10000</td>
</tr>
<tr>
<td>&quot;</td>
<td>10001</td>
</tr>
<tr>
<td>)</td>
<td>10010</td>
</tr>
<tr>
<td>2</td>
<td>10011</td>
</tr>
<tr>
<td>=</td>
<td>10100</td>
</tr>
<tr>
<td>6</td>
<td>10101</td>
</tr>
<tr>
<td>0</td>
<td>10110</td>
</tr>
<tr>
<td>1</td>
<td>10111</td>
</tr>
<tr>
<td>9</td>
<td>11000</td>
</tr>
<tr>
<td>?</td>
<td>11001</td>
</tr>
<tr>
<td>+</td>
<td>11010</td>
</tr>
<tr>
<td>FIGURES</td>
<td>11011</td>
</tr>
<tr>
<td>.</td>
<td>11100</td>
</tr>
<tr>
<td>/</td>
<td>11101</td>
</tr>
<tr>
<td>:</td>
<td>11110</td>
</tr>
<tr>
<td>LETTERS</td>
<td>11111</td>
</tr>
<tr>
<td>Other</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>SOH</td>
<td>NULL</td>
</tr>
<tr>
<td>STX</td>
<td>NULL</td>
</tr>
<tr>
<td>ETX</td>
<td>NULL</td>
</tr>
<tr>
<td>EOT</td>
<td>NULL</td>
</tr>
<tr>
<td>ENQ</td>
<td>NULL</td>
</tr>
<tr>
<td>ACK</td>
<td>NULL</td>
</tr>
<tr>
<td>BEL</td>
<td>NULL</td>
</tr>
<tr>
<td>BKSPC</td>
<td>00000</td>
</tr>
<tr>
<td>HT SPC</td>
<td>00100</td>
</tr>
<tr>
<td>LF</td>
<td>00010²</td>
</tr>
<tr>
<td>VT LF</td>
<td>00010²</td>
</tr>
<tr>
<td>Character</td>
<td>BAUDOT1</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td>FF LF</td>
<td>000102</td>
</tr>
<tr>
<td>CR</td>
<td>010003</td>
</tr>
<tr>
<td>SO</td>
<td>NULL</td>
</tr>
<tr>
<td>SI</td>
<td>NULL</td>
</tr>
<tr>
<td>DLE</td>
<td>NULL</td>
</tr>
<tr>
<td>DC1</td>
<td>NULL</td>
</tr>
<tr>
<td>DC2</td>
<td>NULL</td>
</tr>
<tr>
<td>DC3</td>
<td>NULL</td>
</tr>
<tr>
<td>DC4</td>
<td>NULL</td>
</tr>
<tr>
<td>NAK</td>
<td>NULL</td>
</tr>
<tr>
<td>SYN</td>
<td>NULL</td>
</tr>
<tr>
<td>ETB</td>
<td>NULL</td>
</tr>
<tr>
<td>CAN</td>
<td>NULL</td>
</tr>
<tr>
<td>EM</td>
<td>NULL</td>
</tr>
<tr>
<td>SUB ?</td>
<td>11001</td>
</tr>
<tr>
<td>ESC</td>
<td>NULL</td>
</tr>
<tr>
<td>IS4 LF</td>
<td>000102</td>
</tr>
<tr>
<td>IS3 LF</td>
<td>000102</td>
</tr>
<tr>
<td>IS2 LF</td>
<td>000102</td>
</tr>
<tr>
<td>IS1 SPC</td>
<td>00100</td>
</tr>
<tr>
<td>SPACE</td>
<td>00100</td>
</tr>
<tr>
<td>_ SPACE</td>
<td>00100</td>
</tr>
<tr>
<td>` SPACE</td>
<td>00100</td>
</tr>
<tr>
<td>DEL</td>
<td>NULL</td>
</tr>
</tbody>
</table>

**Note:**

1. The letters and figures codes for BAUDOT are the same. Their meaning is determined by the "shift" state (Letters or Figures) as set by the FIGURES or LETTERS code. The default is Letters.
Record_Voice

Use the Record_Voice action to record a caller's voice response and save it in a database. The action records the caller's voice until the caller presses any telephone key or hangs up the telephone, or until the action reaches the time limit you specify, or until there is a period of silence that equals the length you set in the configuration parameter Amount of Silence to End Recording.

Parameters

Database server (Input, Required, Default=none)
The network name of the database server.

File name (Input, Required, Default=none)
The name of the database file to contain the voice records. You should use the same file name as the OS/2 file name used when the database was created. When the voice record database is created, the record length is 4096 and the key length is 19.

Record key (Input, Required, Default=none)
The key used to save the record. The length of the key must be less than or equal to the defined length of the key of the database file. The maximum length of the key is 15 characters.

Duration (Input, Required, Default=none)
The maximum time limit for the recorded voice. The action will stop recording the caller's voice after this amount of time and save the record in the database, unless the caller presses a key or hangs up the telephone.

Return Codes

Key pressed (RC=0)
The caller pressed a key to indicate the end of the voice record. The action saved the voice record in the database. The key the caller pressed remains in the key buffer. You should:

- Read the key using the Get_a_Tone or Get_Tone_String actions.
- Discard the key using the Clear_Tones action.

The action updates the recording_time system variable to the length, in seconds, of the recorded voice.

Maximum time (RC=1)
The caller spoke up to the maximum time limit as defined by the Duration parameter. The action saved the record in the database. The action updated the recording_time system variable to the length, in seconds, of the recorded voice.

Duplicate record (RC=2)
The key you specified in the Record Key parameter already exists in the database. The action did not record the current voice record. You should delete the existing record using the Delete_Voice action or select a different key and rerun this action.

Record failure (RC=3)
There was a problem placing the voice record into the database. For example, the record could not be created or the database server is inaccessible.
Maximum silence (RC=4)
The caller has been silent for as long as specified in the Record Time Silence configuration parameter.

Caller hung up (RC=HUP)
The caller hung up the phone. The action saved any voice spoken up to this point in the database. The action updated the `recording_time` variable to the length of time, in seconds, of the recorded voice record.

System Variable

`recording_time` (Output)
The amount of time, in seconds, of the recorded voice. The action resets this variable each time it runs.

Comments

- This action is often used with the Play_Voice action to record and play voice records.
- You must create the database prior to running this action.
- You can use the Delete_Voice action to delete any recorded voice records from the database.
- The Record_Voice action is not part of the Voice Messaging actions provided by DirectTalk/2.
**Refresh_OIA**

**Note:** You must have the optional Communications feature installed to use this action.

Use the Refresh_OIA action to update the DirectTalk/2 system variables OIA_data_ascii and OIA_data with the latest copy of the Operator Information Area (OIA) from 3270 and 5250 terminal emulation sessions. The Operator Information Area for the 3270 and 5250 terminals does not form part of the screen data and is obtained, from the emulated terminal, via these system variables. The variables are not automatically updated by DirectTalk/2 and you should use this action immediately prior to using these system variables to ensure the data reflects the latest OIA status.

For an EHLLAPI host session, both variables are updated.

For an ARTIC or LUA host session, OIA_data_ascii is updated and the OIA_data is set to binary zero.

**Parameters**

None.

**Return Codes**

**Variables refreshed (RC=0)**
The OIA variables have been refreshed.

**Error (RC=1)**
The action could not refresh the OIA variables.

**Not connected (RC=2)**
The action could not refresh the OIA variables because the application is not currently connected to a host.

**System Variables**

**OIA_data_ascii (Output)**
The action stores the OIA ASCII data in this variable.

**OIA_data (Output)**
The action stores the OIA binary data in this variable.

**Comments**

- You must establish a connection to the host application with the Connect_Screen action before using this action.

  Both the 3270 and 5250 terminals provide an Operator Information Area (OIA) at the bottom of the screen, which shows the current status of the terminal. You may need to use this information to determine if the keyboard is locked or to diagnose error situations reported by other communications actions.

  The different terminal emulators, supported by DirectTalk/2 provide the OIA data in different formats each of which is described below.

  **Note:** OIA data is not available for ASCII terminals.
**EHLLAPI emulators**

The OIA data is returned in a 103 byte block consisting of:

- Identifier byte
- 80-character image of the OIA as seen at the bottom of the window when the terminal session is viewed on the OS/2 desktop
- 22 bytes of OIA Group Indicator bits

This data is copied to the OIA system variables as follows:

**OIA_ascii_data:**

OIA_ascii_data contains an 80 character ASCII image of the OIA.

For CM/2 sessions this is the same as seen in the OIA line of the terminal window.

For PCOMM sessions DirectTalk/2 translates the special characters, that are displayed on the PCOMM OIA line, into an equivalent ASCII image in a similar format to that for CM/2. The layout and meaning of the data is:

**Column 1: Control Unit Status**

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>DFT connection through a 3274 Control Unit</td>
</tr>
<tr>
<td>S</td>
<td>DFT connection through a 3174 Control Unit</td>
</tr>
<tr>
<td>I</td>
<td>DFT connection to a 4361 Host (CM/2 only)</td>
</tr>
<tr>
<td>N</td>
<td>DFT connection to a 9370 Host</td>
</tr>
<tr>
<td>M</td>
<td>Attachment to a host other than above (PCOMM only)</td>
</tr>
<tr>
<td>2</td>
<td>Attachment to a host other than above (CM/2 only)</td>
</tr>
<tr>
<td>V</td>
<td>Attachment to a host system (PCOMM 5250 only)</td>
</tr>
</tbody>
</table>

**Column 2: Attachment Type**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Attachment method is non-SNA</td>
</tr>
<tr>
<td>B</td>
<td>Attachment method is SNA</td>
</tr>
</tbody>
</table>

**Column 3: System Available**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>Terminal is connected to a host application</td>
</tr>
<tr>
<td>?</td>
<td>Terminal is not connected to any host program (SNA only)</td>
</tr>
</tbody>
</table>

**Columns 4-7: Session ID**

aaaa The first 4 characters of the long name of the session (CM/2 only);
the short name of the session (PCOMM only)

**Columns 9-17: Input Inhibited**

Refer to the CM/2 or PCOMM User’s Guide for a description of the first
three errors.

X COMMnnn Communications error

X MACHnnn Machine check error

X PROGnnn Program check error

X SYSTEM The host has locked the keyboard
The host has not yet acknowledged the last input
Unsupported function selected
Too much data entered in a field
Attempt to send data to a protected field
An error occurred during a presentation space print operation
Printer not available for a presentation space print operation
Input is inhibited (CM/2 5250 only)

Columns 19-25: Communication Reminder

COMMnnn Refer to CM/2 or PCOMM User’s Guide for a description of this message.

Columns 26-27: Message Waiting Indicator (5250 only)

MW The host system has one or more messages for the session

Columns 31-38: Host Name (CM/2 5250 only)

ab..cd APPC Partner LU Alias of the host computer when the session is active. Blank when the session is not active.

Columns 42-44: Numeric Lock (PCOMM 3270 only)

NUM The cursor is in a numeric field and the 3270 numeric lock feature is enabled

Columns 61-65: Printer Status (CM/2 only)

o-o The emulation session is configured for print support
o-* A presentation space print request is pending or in progress
o_ An error occurred during a presentation space print operation

Columns 73-80: Emulator Name

ab..cd The DirectTalk/2 emulation session name as displayed in the Node Manager.

OIA_data:

OIA_data contains the Group Indicator bits translated from the EHLLAPI binary form into an ASCII string. Each bit of the binary data is converted into a '0' or '1' character producing a string of 176 characters as in the following example:

<byte 1><byte 2> ... <byte21><byte22>
bit posn: 0123456701234567 ... 0123456701234567
bits: 0010010000010000 ... 0100010000000000
chars: '0010010000010000 ... 0100010000000000'

The Group Indicator meanings are described in the relevant publications for the communication product you are using:

CM/2: Communications Manager/2 EHLLAPI Programming Reference (SC31-6163).

PCOMM: Personal Communications Version 4.0 Programmer’s Reference for OS/2 (S85G-8681).
**ARTIC and LUA emulators**

The emulators for ARTIC and LUA connections do not generate the Group Indicator bits. For these emulators only the OIA_data_ascii variable is refreshed by this action.

**OIA_ascii_data:**

The layout and meaning of the data is:

<table>
<thead>
<tr>
<th>Column 1: Control Unit Status</th>
<th>4</th>
<th>Always set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column 2: Attachment Type</td>
<td>B</td>
<td>Attachment method is SNA</td>
</tr>
<tr>
<td>Columns 3-6: System Available</td>
<td></td>
<td>Terminal is connected to host system services</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Terminal is connected to a host application</td>
</tr>
<tr>
<td>Columns 11-18: Input Inhibited</td>
<td></td>
<td>The host has not yet acknowledged the last input</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>The keyboard is locked</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Unsupported function selected</td>
</tr>
<tr>
<td>Columns 73-80: Emulator Name</td>
<td>ab..cd</td>
<td>The DirectTalk/2 emulation session name as displayed in the Node Manager</td>
</tr>
</tbody>
</table>

**OIA_data:**

(NULL) the action sets this variable to zero length
**RemoveStatItem**

Use the RemoveStatItem action to remove the variable you specify from the statistics record.

**Parameter**

**Variable (Input, Required, Default=none)**

The name of the variable you want to remove from the statistics record.

**Return Code**

**Removed (RC=0)**

The action removed the variable you specified from the statistics record.
Return_from_Appl

Use the Return_from_Appl action to return processing from a subapplication to the linking application. This action restores the linking application's local variable pool. If this action is used in the main application, the application will stop.

Parameter

Return code (Input, Optional, Default='0')

The valid return codes (0 through 14) for the linking application's link action. You can specify a variable or a literal.

System Variables

application_name (Output)

The name of the current application. The application name returns to the name of the linking application.

linked_from_appl (Output)

The name of the linking or calling application. The name returns to the value it had prior to linking to this application.

Comments

- You can develop subapplications in the same manner as other applications, such as using DirectTalk/2 actions.
- Subapplications are not user-created actions. See the IBM CallPath DirectTalk/2 Application Programmer’s Guide for more information on creating your own actions.
- To share information between the application and the subapplication, assign values to global variables in the subapplication prior to returning to the application. You can use the Return_from_Appl action to specify the return code value for the link statement in the Link_to_Appl action. This action returns a default value of zero.
- Use this action at the end of the main application shutdown or quiesce routine to stop the main application.
- An example of the Return_from_Appl action is in the CALC sample application.
Search_Directory

Use the Search_Directory action to retrieve a directory entry, based on a telephone number.

You must have the optional Voice Messaging Feature to use this action.

Parameters

Phone number (Input, Required, Default=none)
The telephone number the action uses to search the directory.

Search digits (Input, Required, Default='5')
The number of trailing digits the action uses to search the directory. You can use an extension.

Variable prefix (Input, Optional, Default='direct')
The prefix the action uses when creating the variable in which to save the data.

Return Codes

Entry found (RC=0)
The action found the entry.

Entry not found (RC=1)
The action could not find the entry you specified.

More than 1 entry (RC=2)
The action found more than one entry for the telephone number you specified and retry. You must specify more digits for the telephone number.

Error (RC=3)
The action could not access the database.

User Variables

These variables are created and returned by the action. They are output variables.

<prefix>_dept
The department of the mailbox owner.

<prefix>_fnm
The first name of the mailbox owner.

<prefix>_grtgkey
Generated key for a personal greeting voice segment in the recorded names and greetings database.

<prefix>_idno
The unique identification value for the entry.

<prefix>_lnm
The last name of the mailbox owner.

<prefix>_loc
The location of the mailbox owner.

<prefix>_mailbox
The assigned mailbox number of the mailbox owner.
**System Variable**

directory_server (Input)

The name of the remote server where the directory is held.

**Comments**

- The voice application can pass the values of the variables in the Search_Directory action to the Put_User_Info action by specifying the same prefix in both actions.

- You can pass the value in the `<prefix>_namekey` variable to the Take_a_Message action to create a system greeting with the mailbox owner's name.

- Use the name in the `<prefix>_grtgkey` variable to record a personal greeting in the names database. You can pass the value in the `<prefix>_grtgkey` variable to the Take_a_Message action to generate a personal greeting from the mailbox owner.

- You can use all or part of the telephone number (such as an extension) in the search. Use enough digits of the telephone number to make it unique. Under normal circumstances, the caller has specified the telephone number in response to a prompt from the voice application.

- The action places the results in local variables. These variables consist of a prefix you specify and a predetermined suffix. Use an action parameter to set the prefix. The action creates local variables if they do not already exist.
**Search_Screen**

**Note:** You must have the optional Communications Feature to use this action.

Use the Search_Screen action to perform a single search on the virtual screen for the information you specify.

You must specify the string of characters you want the action to search for. You must also specify the starting row and column to begin the search.

The action searches from the starting row and column to the end of the screen. You can specify row 1, column 1 to see if a string appears anywhere on the screen.

**Parameters**

- **Text to search for (Input, Required, Default=none)**
  
  The text you want the action to search for on the simulated screen.

- **Starting row (Input, Required, Default=1)**
  
  Specifies the row on the screen in which to begin searching for the text you specify.

- **Starting column (Input, Required, Default=1)**
  
  Specifies the column in the row on the screen in which to begin searching for the text you specified.

**Return Codes**

- **Text found (RC=0)**

  The action found the text.

- **Text not found (RC=1)**

  The action did not find the text.

- **Not connected (RC=2)**

  The action could not access the screen because there is no session currently active with the host application. This return code can indicate that the host is not currently running.

- **Session error (RC=3)**

  The action encountered a problem with the session with the host application. This return code can indicate that the host is not currently running.

- **Session disconnected (RC=4)**

  The action could not retrieve the information because the session with the host has been disconnected. This return code can indicate that the host is not currently running.

**System Variables**

- **last_found_row**

  The starting row on the screen of the found text. This variable is only updated when RC=0.

- **last_found_col**

  The starting column on the screen of the found text. This variable is only updated when RC=0.
Comments

- You must establish a session with the host application using the Connect_Screen action prior to running this action.

- If this action immediately follows the Connect_Screen action, your search may start before the emulator screen is completely displayed. To prevent your search attempting to operate on incomplete data, follow Connect_Screen with Wait_Scr_Update, or use this action to wait for data at the bottom of the screen before searching for the data you really require.

- This action distinguishes between upper and lower case characters.

- You can use a series of Search_Screen actions to determine which screen is presented by the host application. For example, during the logon process, the order of screens may vary.

- If the target text is found, the starting position of the string is returned in the last_found_row and last_found_col variables. These variables are only set if the target is found. They are not changed or reset for any other return code.
Search_String

Use the Search_String action to determine the position of the first occurrence of a literal or variable string from a starting point within the source string. You must specify the string to search for and the position in the source string to start the search.

Parameters

Variable for result (Output, Required, Default=none)
The variable in which the action stores the position of the string.

Source string (Input, Required, Default=none)
The source string to search. You can specify a variable or a literal.

String to search for (Input, Required, Default=none)
The string to search for. You can specify a variable or a literal.

Starting position (Input, Required, Default='1')
The position, relative to one, in the source string at which to begin searching. You can specify a variable or a literal.

Return Codes

String found (RC=0)
The action found the string in the source string and stored the position of the string in the result variable.

String not found (RC=1)
The action did not find the string in the source string. The action did not place any value in the result variable.

Error (RC=2)
An error condition occurred. You may have specified:
- A source variable that did not exist
- An invalid variable name
- An invalid starting position

System Variable

case_sensitive (Input)
If the case_sensitive variable is set to “Y” or “y” in the first byte then the Search_String search is case sensitive; otherwise, the search is not case sensitive.

It is recommended that the system variable case_sensitive be used as a parameter for those actions that accept it so that case sensitivity is consistent throughout your application, if required.

Comments

- If the result variable does not exist, the action creates it as a local variable.
- The action numbers the search positions starting with one. You cannot specify '0' as a search position.
Send_ADSI

Use the Send_ADSI action to send Server Display Control (SDC) and Feature Download Management (FDM) data to an ADSI telephone.

Parameters

FDM/SDC script name (Input, Required, Default=none)
The name of the ADSI SDC or FDM script which contains the compiled data to be sent to an ADSI telephone.

ADSI function name (Input, Optional, Default=none)
The name of the function to be executed on an ADSI telephone as defined in the SDC script file.

This parameter is required for SDC scripts and should be left blank for FDM scripts.

Transmission mode (Input, Required, Default=voice)
The acceptable values for this parameter are voice or data. There is a difference in the protocol used to send ADSI information in the two modes. Data mode is faster than voice mode but you cannot play any voice segments while in data mode. To use data mode the phone must already have been set to data mode by an SDC command or a soft key return sequence within your ADSI script.

Return Codes

Data sent (RC=0)
The data has been successfully sent to the ADSI telephone.

Not enabled (RC=1)
An attempt was made to download to a non ADSI telephone.

Download denied (RC=2)
An attempt to download an FDM file has been rejected by the caller.

No available resource (RC=3)
The ADSI feature is not installed or there is no ADSI resource currently available.

Error (RC=4)
An error has occurred. For example you may have specified a script file that does not exist, or a transmission error may have occurred when the data was downloaded. An appropriate message is given.

HUP (RC=HUP)
The caller has hung up.

System Variables

The following optional variables can be used to define a common server and application on which you can store ADSI scripts and associated parameters for several applications to use.

ADSI_cmn_srvr (Input)
The server name of the DirectTalk/2 node that contains the ADSI scripts and their associated parameters.
**ADSI_cmn_appl (Input)**

The name of the application which contains the ADSI scripts and their associated parameters.

**Comments**

Substitution parameter variables that are used within an ADSI script must be defined somewhere in the Voice program before the Send_ADSI statement that sends the script.
Send Alert

DirectTalk/2 Version 2.1 provides this system action for the sending of a NetView alert.

Parameters

**Caller name (Input, Required, Default=None)**

The name of the caller sending the alert. For example, the name of a step in the application processing. The name can be up to 8 characters in length.

**Alert message (Input, Required, Default=None)**

The alert message text. If the message is longer than 56 characters in length, then the message is truncated to fit the output area. This means that the truncation point varies according to the space available, but at least 56 characters will appear. Special characters such as newlines or carriage returns must not be included in the message.

Return Codes

**Alert sent (RC=0)**

The alert was sent.

**Alerts not enabled (RC=1)**

Alerts are not enabled; alert not sent.

**Error (RC=2)**

This code is returned if the action encounters any error that prevents the alert being sent.

Comments

The GSI node name and application name are sent with the alert, if available. The following alert codes are used for the alert:

**Alert Description**

Operator notification

**Probable Cause**

Software program

**Recommended Action**

Perform problem determination procedure

The message has format:

```
EXH4590 %s
```

where %s is the **Alert Message**.
Send.Keys_to_Scr

Note: You must have the optional Communications Feature to use this action.

Use the Send.Keys_to_Scr action to simulate keystrokes to specify data on the virtual screen and to interact with the host application.

For example, the information can be a transaction name, an account number, or a program function key. Use the regular keys of the keyboard to type the data into a field. Use an escape character '@' or '^', and the key codes shown in Table 21 on page 267 and Table 22 on page 268, to send the special display control keys.

Parameters

Keystrokes (Input, Required, Default=none)
The keystrokes you want to send to the host application.

Starting row (Input, Optional, Default=none)
Specifies the row in the field you want to send the keystrokes. If the row is specified, the starting column must also be specified. If not specified, the action places the keystrokes at the cursor location.

Starting column (Input, Optional, Default=none)
Specifies the column in the field you want to send the keystrokes to. If starting column is specified the starting row must also be specified. If not specified, the action places the keystrokes at the cursor location.

Wait timeout (Input, Optional, Default=60)
Specifies the time, in seconds, the action waits for the host to unlock the keyboard after sending an AID key. A timeout of 1 to 300 seconds may be specified. If not specified the action waits for 60 seconds. The operation of this parameter is modified by the setting of the key_immediate system variable, see the section describing system variables below.

Note: Starting row, Starting column and Wait timeout parameters are ignored for ASCII terminals.

Return Codes

Keystrokes sent (RC=0)
The action sent the keystrokes you specified.

Keys not sent (RC=1)
The action could not send the keystrokes because an error condition exists. This may be due to a number of conditions such as:

- An attempt was made to send keystrokes to an invalid area of the screen. Either a 3270 protected field or at an attribute location.
- A non-numeric entry was made in a numeric only field (5250 only).
- An attempt was made to send keystrokes to a locked keyboard. This is usually a temporary condition due to slow host operation or due to the host application.
- An attempt was made to send more than 256 characters to an ASCII terminal.
Not connected (RC=2)
The action could not access the screen because there is no session currently active with the host application. This return code can indicate that the host is not currently running.

Session error (RC=3)
The action encountered a problem with the session with the host application.

Session disconnected (RC=4)
The action could not send the keystrokes because the session with the host had been disconnected.

Keyboard locked (RC=5)
After the action sent an AID key, the host did not unlock the keyboard, within the time specified by the Wait timeout parameter. The host application or session is either slow to respond or may be having problems or you may have specified a Wait timeout value that is too small for your system.

Note: This return code does not apply to ASCII terminals.

System Variables

CM_key_reset
If CM_key_reset is set to 'NO' in any case mix the automatic reset which normally precedes the Send_Keys_to_Scr action is disabled.

If the CM_key_reset variable is not defined or set to 'YES', the keystrokes sent are preceded by a reset key. This has the effect of automatically resetting any input inhibit conditions that can be reset at this time.

The setting of this variable to 'NO' overcomes a problem experienced with the 5250 SysReq key causing premature exit from the 5250 SysReq state before keystrokes can be sent. This system variable is ignored for ASCII terminals.

key_immediate (input)
A value of 'Y' or 'YES' in any case mix forces the action to ignore processing of the keyboard lock states. The action returns immediately with the Keystrokes sent return code (RC=0). When using this value, the application should check that the host is ready for further keystrokes, before attempting to send any keys to the screen. This can be done through the use of the Search_Screen, Wait_Scr_Update, or Refresh_OIA actions.

Any other value, or if key_immediate is undefined, and results in normal lock processing as described below.

This system variable is ignored for ASCII terminals.

Keyboard Lock Processing
When using 3270 or 3250 terminal emulation sessions, the keyboard is locked after an AID (attention) key is sent to the host, and it may take some time for the host acknowledgement to unlock the keyboard. The behavior of this action varies depending on the setting of the key_immediate system variable and whether 'Retry sending keys' has been checked in the emulator settings notebook.
If the `keyImmediate` variable is set to 'YES', and the application is connected to a valid host session, the action always returns the Keystroke sent return code (RC=0), immediately after attempting to send the keys specified in the `keystrokes` parameter.

If the `key Immediate` variable is not set to 'YES', the behavior of the action depends on the keyboard lock state when the action is invoked, and whether 'Retry sending keys' is selected. The various options are:

**Keyboard not locked when Send_Keys_to_Scr is invoked**

The action sends the keys to the screen and, if an AID key is included, attempts to send the keys to the host. The action returns the Keystrokes sent return code (RC=0) if the host unlocks the keyboard within the time specified by the `Wait_timeout` parameter, or the Keyboard locked return code (RC=5), if the host does not unlock the keyboard in this time. If you receive the Keyboard locked return code, you should take the same steps as when using `keyImmediate='Y'`, to ensure that the host unlocks the keyboard.

**Keyboard locked when Send_Keys_to_Scr is invoked, and 'Key Retry' is checked in the emulator setup.**

The action attempts to send the keys for up to one minute (4 minutes for EHLLAPI). If the host has not unlocked the keyboard in this time, the action returns the Keys not sent return code (RC=1). If the keys are successfully sent within this time, the action returns the Keystrokes sent return code (RC=0).

**Keyboard locked when Send_Keys_to_Scr is invoked, and 'Key Retry' is not checked in the emulator setup.**

The action returns immediately with the Keys not sent return code (RC=1), without attempting to send the keystrokes.

**Note:** When an AID key is sent to the host, it locks the keyboard for some period of time.

If the keyboard is still locked by the host when a second attention key is sent, you will experience unpredictable results, including the loss of any data sent prior to the attention key and a host response of up to 4 minutes. Therefore, it is important that, when setting `keyImmediate` to yes, your application ensures that the host is ready for an attention key. You can do this using the Search_Screen or Wait_Scr_Update actions.

Alternatively you can set the 'KEY_RETRY' keyword to 'NO' in your emulator configuration file. See the online README file provided with DirectTalk/2 for details of how to set this parameter.

Additionally, if `keyImmediate` is not set, and your application receives the Keyboard locked return code, you should take the same steps to ensure that the host has unlocked the keyboard.
Comments

- You must establish a connection to the host with the Connect_Screen action before using this action.

- Since the action places the data at the cursor location if you do not specify the row and column, you can use the Tab keys to simplify the process of specifying data into multiple screen fields. For example, a host screen could have three input fields (Transaction ID, Account number, and PIN). After collecting this information from the caller, you can use the following sequence to fill the screen:

  1. Make sure the cursor is in the first field.
  2. text=transaction (Variable) – the input from the caller.
  3. text=' @T' – use the Tab key to move to the next field.
  4. text=account_number (Variable) – the input from the caller.
  5. text=' @T' – the Tab key.
  6. text=pin (Variable) – the input from the caller.
  7. text=' @E' – the Enter key.

As an alternative, if the cursor automatically moves to the next field when the current field is full, you can omit the Tab steps. You can also concatenate all of the input into a single string and use the Send_Keys_to_Scr action to send the string.

Table 21 and Table 22 on page 268 list the key codes and their meanings.

- For ASCII terminals a maximum of 256 characters can be sent with this action. If you need to send more data than this, use multiple Send_Keys_to_Scr actions.

### Table 21. 3270/5250 Emulator keyboard symbols for the Send_Keys_to_Scr action

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Key</th>
<th>Symbol</th>
<th>Key</th>
<th>Symbol</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>@@</td>
<td>@</td>
<td>@2</td>
<td>PF2 *</td>
<td>@g</td>
<td>PF16 *</td>
</tr>
<tr>
<td>@C</td>
<td>Clear *</td>
<td>@3</td>
<td>PF3 *</td>
<td>@h</td>
<td>PF17 *</td>
</tr>
<tr>
<td>@E</td>
<td>Enter *</td>
<td>@4</td>
<td>PF4 *</td>
<td>@i</td>
<td>PF18 *</td>
</tr>
<tr>
<td>@F</td>
<td>Erase EOF</td>
<td>@5</td>
<td>PF5 *</td>
<td>@j</td>
<td>PF19 *</td>
</tr>
<tr>
<td>@N</td>
<td>New Line</td>
<td>@6</td>
<td>PF6 *</td>
<td>@k</td>
<td>PF20 *</td>
</tr>
<tr>
<td>@P</td>
<td>Print **</td>
<td>@7</td>
<td>PF7 *</td>
<td>@l</td>
<td>PF21 *</td>
</tr>
<tr>
<td>@R</td>
<td>Reset</td>
<td>@8</td>
<td>PF8 *</td>
<td>@m</td>
<td>PF22 *</td>
</tr>
<tr>
<td>@T</td>
<td>Tab</td>
<td>@9</td>
<td>PF9 *</td>
<td>@n</td>
<td>PF23 *</td>
</tr>
<tr>
<td>@A</td>
<td>Attn</td>
<td>@a</td>
<td>PF10 *</td>
<td>@o</td>
<td>PF24 *</td>
</tr>
<tr>
<td>@U</td>
<td>Page Up +</td>
<td>@b</td>
<td>PF11 *</td>
<td>@x</td>
<td>PA1 *</td>
</tr>
<tr>
<td>@D</td>
<td>Page Down +</td>
<td>@c</td>
<td>PF12 *</td>
<td>@y</td>
<td>PA2 *</td>
</tr>
<tr>
<td>@X</td>
<td>Field Exit +</td>
<td>@d</td>
<td>PF13 *</td>
<td>@z</td>
<td>PA3 *</td>
</tr>
<tr>
<td>@H</td>
<td>Help +</td>
<td>@e</td>
<td>PF14 *</td>
<td>@S</td>
<td>SysReq</td>
</tr>
<tr>
<td>@1</td>
<td>PF1 *</td>
<td>@f</td>
<td>PF15 *</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

* AID key symbol
** Supported only with EHLLAPI emulators
+ 5250 terminal emulation.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Symbol</th>
<th>Meaning</th>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>^2</td>
<td>NUL</td>
<td>^T</td>
<td>DC4</td>
<td>^H</td>
<td>BS</td>
</tr>
<tr>
<td>^3</td>
<td>ESC</td>
<td>^Y</td>
<td>EM</td>
<td>^J</td>
<td>LF</td>
</tr>
<tr>
<td>^4</td>
<td>FS</td>
<td>^U</td>
<td>NAK</td>
<td>^K</td>
<td>VT</td>
</tr>
<tr>
<td>^5</td>
<td>GS</td>
<td>^l</td>
<td>HT</td>
<td>^L</td>
<td>FF</td>
</tr>
<tr>
<td>^6</td>
<td>RS</td>
<td>^O</td>
<td>SI</td>
<td>^Z</td>
<td>SUB</td>
</tr>
<tr>
<td>^7</td>
<td>US</td>
<td>^P</td>
<td>DLE</td>
<td>^X</td>
<td>CAN</td>
</tr>
<tr>
<td>^8</td>
<td>DEL</td>
<td>^A</td>
<td>SOH</td>
<td>^C</td>
<td>ETX</td>
</tr>
<tr>
<td>^Q</td>
<td>DC1</td>
<td>^S</td>
<td>DC3</td>
<td>^V</td>
<td>SYN</td>
</tr>
<tr>
<td>^W</td>
<td>ETB</td>
<td>^D</td>
<td>EOT</td>
<td>^B</td>
<td>STX</td>
</tr>
<tr>
<td>^E</td>
<td>ENQ</td>
<td>^F</td>
<td>ACK</td>
<td>^N</td>
<td>SO</td>
</tr>
<tr>
<td>^R</td>
<td>DC2</td>
<td>^G</td>
<td>BEL</td>
<td>^M</td>
<td>CR</td>
</tr>
</tbody>
</table>
Send_TDD

Use the Send_TDD action to send ASCII data, stored in a DirectTalk/2 database segment, to a caller as TDD data. The action will retrieve the text from the specified Voice System database record. The action sends the text to completion unless the caller presses any telephone key or hangs up the telephone.

For any application using Send_TDD, permanently assign a TDD resource to the application for the duration of each call by using the Assign_Resource and Free_Resource actions.

**Note:** The data transmitted cannot be interrupted by TDD data received from the caller.

**Parameters**

- **Database server (Input, Required, Default=none)**
  The network name of the database server.

- **Filename (Input, Required, Default=none)**
  The file name of the Voice Systems database that contains the TDD record.

- **Record key (Input, Required, Default=none)**
  The key of the record that contains the ASCII text to be sent.

**Return Codes**

- **TDD sent (RC=0)**
  TDD data was converted and sent.

- **Key pressed (RC=1)**
  This indicates the receipt of a DTMF digit. This means that the outgoing TDD data was interrupted and may not be complete. The caller has probably interrupted the TDD data by pressing a key on the phone pad.

- **Error (RC=2)**
  An error condition occurred. You may have specified a filename or record key that does not exist.

- **Caller hung up (RC=HUP)**
  The caller hung up the telephone.

**Comments**

- The TDD data is sent as BAUDOT.
- Table 20 on page 244 is the ASCII/BAUDOT conversion table used by DirectTalk/2.
- Any ASCII characters not found in Table 20 on page 244 will be converted as blanks.
- If the **Key pressed** return code is returned, the DTMF digit can be retrieved using Get_a_Tone or Get_Tone_String.
**Send_TDD_String**

Use the Send_TDD_String action to send ASCII data to a caller as TDD data. The action will send the text contained in a DirectTalk/2 variable. The action sends the text to completion unless the caller presses any telephone key or hangs up the telephone.

For any application using Send_TDD_String, permanently assign a TDD resource to the application for the duration of each call using the Assign_Resource and Free_Resource actions.

**Note:** The transmitted data cannot be interrupted by TDD data received from the caller.

**Parameter**

Variable name or literal (Input, Required, Default=none)

The ASCII string to be sent as TDD data.

**Return Codes**

TDD sent (RC=0)

TDD data was converted and sent.

Key pressed (RC=1)

This indicates the receipt of a DTMF digit. This means that the outgoing TDD data was interrupted and may not be complete. The caller has probably interrupted the TDD data by pressing a key on the phone pad.

Error (RC=2)

An error condition occurred. You may have specified a variable that does not exist.

Caller hung up (RC=HUP)

The caller hung up the telephone.

**Note:** A '<' character is transmitted as a carriage return.

**Comments**

- The TDD data is sent as BAUDOT.
- Table 20 on page 244 is the ASCII/BAUDOT conversion table used by DirectTalk/2.
- Any ASCII characters not found in Table 20 on page 244 will be converted as blanks.
- A newline character will be converted as the CRLF (carriage return, linefeed) character sequence.
- If the Key pressed return code is returned, the DTMF digit can be retrieved using Get_a_Tone or Get_Tone_String.
Set_Timeout

Use the Set_Timeout action to set the system variable `timeout_value` directly. This system variable is used in subsequent calls to the Wait_for_Hang action.

Parameter

Value (Input, Required, Default='10')

Specify the time in seconds to set the `timeout_value` variable to.

Return Codes

Timeout Set (RC=0)

The action assigned a new timeout value.

Error (RC=1)

The action could not assign the value to the variable. The input data variable may not exist.

Comments

- The `timeout_value` system variable cannot be set by the Set_Variable action as the system uses the `timeout_value` variable for display only.
- The `timeout_value` variable is normally set by the Play_Module action and is used later by the Get_Tone_String action. The variable should not be set to a very large value as it may prevent the application performing other tasks such as refreshing host screens.
Set_Variable

Use the Set_Variable action to assign a new value to a variable.

Parameters

Variable (Output, Required, Default=none)
The name of the variable to assign a value to.

Value (Input, Required, Default=none)
The data to assign to the target variable.

Return Codes

Variable set (RC=0)
The action assigned the new value to the variable.

Error (RC=1)
The action could not assign the value to the variable. The input data variable may not exist.

Comments

• You must enclose literal data (character and numeric data) in single quotes.
• If the variable does not exist in the local or global variable pool, the action creates the variable as a local variable.
• An example of the Set_Variable action is in the CALC sample application.
**Start_Clock**

Use the Start_Clock action to start a timer you specify at 0.

**Parameter**

Clock number (Input, Required, Default=none)
The number of the clock you want to start. You can specify a number from 1 through 8.

**Return Code**

Clock started (RC=0)
The action started the clock you specified.

**Comments**

- Each application contains its own set of independent clocks, numbered from 1 to 8. Therefore, you must start and stop a clock in the same application. (You cannot start a clock in one application, link to a subapplication using Link_to_Appl, and stop the clock.)
- You do not need to use the Stop_Clock action before you use the Start_Clock action.
- The clock resolution is in seconds.
Stop_Clock

Use the Stop_Clock action to store the duration from the last Start_Clock action in the variable you specify.

Parameters

Clock number (Input, Required, Default=none)
Number of the clock you want to time the duration.

Variable for result (Input, Required, Default=none)
Variable to receive the duration in seconds.

Return Code

Clock stopped (RC=0)
The action stored the amount of time the clock ran in the result variable parameter. The clock continues to run.

Comments

- You must stop a clock in the same application that you started it, or the results stored will not be correct.
- You can use the Stop_Clock action as many times as you want after initially starting the clock using the Start_Clock action.
- Each Stop_Clock action stores the duration from the last Start_Clock action, but will not actually halt the timer.
- Use the Start_Clock action to reset the duration.
Switch_Screen

**Note:** You must have the optional Host Communications Feature to use this action.

Use the Switch_Screen action to change from one host session that the application is currently using to another host session.

**Parameters**

**Session name (Required, Input, Default=none)**

The session identifier. Use the variable you specified for the `Session name var` parameter on the Connect_Screen action to specify the session to switch to.

**Return Codes**

**Switched (RC=0)**

The switch was successful.

**Session not found (RC=1)**

The action could not change the host session the application is currently using. The value you specified for the `Session name` parameter does not correspond to an active session.

**Session error (RC=2)**

The action encountered a problem with the session with the host application. This return code can indicate that the host is not currently active or may not be communicating with the DirectTalk/2 system.

**Session disconnected (RC=3)**

The action could not change the host session because the session with the host is disconnected.

**Comments**

- You must use Connect_Screen actions to establish connections to multiple host applications before using this action.
- Figure 6 on page 276 contains a sample flow for the Switch_Screen action.
Connect_Screen (Emulator = 3270_server,  
    Session name var = 3270_sess_name)  
    :  
        (logon processing for 3270 host)  
    :  
Connect_Screen (Emulator = 5250_server,  
    Session name var = 5250_sess_name)  
    :  
        (logon processing for 5250 host)  
    :  
Wait_for_Call  
    :  
        (get input from caller)  
    :  
Switch_Screen (Session name = 3270_sess_name)  
    :  
        (get information from 3270 screens)  
    :  
Switch_Screen (Session name = 5250_sess_name)  
    :  
        (get information from 5250 screens)  
        (play information back to caller)  
    :  
Hang up and loop back to wait for call.

Figure 6. Sample Action Flow for Switch_Screen
Take_a_Message

Use the Take_a_Message action to record telephone messages from callers and store the messages in a mailbox. You must have the optional Voice Messaging Feature to use this action.

Message recording stops when:

- The caller presses a tone key.
- The maximum duration is reached.
- A period of silence is detected that is equal to the Telephony Server, Voice Configuration parameter Record Term Silence.
- The caller hangs up.

Parameters

Mailbox number (Input, Required, Default=user)
The mailbox number in which to store the message. You can specify the mailbox number by using the <prefix>_mailbox variable returned by the Search_Directory action.

Type of greeting (Input, Required, Default='User')
The type of greeting the caller receives. You can specify the following types of greeting:

- 'none'
  The action skips the greeting generation.

- 'user'
  The action:
  - Determines if it has access to the personal greeting you specify in the tmsg_g/n_server, tmsg_g/n_db, and tmsg_g/n_greet variables.
  - If the action has access, the action plays the greeting you specify to the caller.
  - If the action does not have access, it plays a system greeting.

- 'system'
  The action plays a greeting in the form:
  The party you called <tmsg_g/n_name> at <tmsg_phone> is not available. At the tone, please leave a message.

  If the action cannot access the recorded name segment or the tmsg_phone variable is not available, it does not include them in the greeting.

Max duration (Input, Required, Default=none)
The maximum duration of the recorded message. You can specify the recording time by using the value of the <prefix>_rtime variable in the Search_Directory action.
Return Codes

Okay, caller hung up (RC=0)
The caller hung up, but has left a message. The action stores the message.

Okay, caller on line (RC=1)
A message has been left and the caller has not hung up.

Mailbox full (RC=2)
The action cannot put a message in the mailbox because the user's mailbox is full. This return code is returned immediately and the application must inform the caller of the situation, as appropriate.

Error (RC=3)
You did not specify all the required variables and parameters or the action cannot access a server or database you specified.

Greeting exited (RC=4)
The caller pressed the key sequence specified in the variable tmsg_grtg_exit during the greeting. This allows access to special application menus by authorized users.

Caller hung up (RC=HUP)
The caller hung up before leaving a message.

Required System Variables

Note: The values for the required system variables are defined and set to default values in the Voice Application Developer Setup file during voice messaging installation and configuration.

mailbox_server (Input)
The network name of the mailbox server the action uses to access the mailbox.

tmsg_msg_server (input)
The network name of the database server the action uses to access the message database where the message text is stored.

tmsg_msg_db (Input)
The name of the message database.

Optional System Variables

tmsg_g/n_server
The network name of the database server used to access the recorded name and personal greetings. If you do not specify this server, the action uses a system greeting.

tmsg_g/n_db
The name of the database containing the recorded names and personal greetings. If you do not specify the database name, the action uses a system greeting.
tmsg_g/n_greet
The key of a personal greeting. You can specify the key by using the value in the <prefix>_grtgkey variable returned by the Search_Directory action. If you do not record a personal greeting, the action builds a system greeting.

tmsg_g/n_name
The key for the recorded name. You can specify the key by using the value in the <prefix>_namekey variable returned by the Search_Directory action. If you do not record names using the DirectTalk/2 Directory Administration manager, the action uses a system greeting without a name.

tmsg_grtg_exit
This variable can be set before the action is called and can be used to provide a fastpath to the message recording in two different ways:

- If the variable is empty, or its value is null, any DTMF tone causes the greeting to be terminated and the input tone to be played.
- If the variable is set to a string which starts with 'F' (F007 for example), then receipt of tones which do not match the string following the 'F' (007 in the example), causes the greeting to be terminated and the input tone to be played.

In both these cases, the caller can record a message immediately.

If the variable is set to any string of valid DTMF characters, any received DTMF tones that match the value of the variable (or the string following the 'F' if the first character is F) cause the action to end with a return code of 'Greeting exited'. The valid DTMF characters are 0 through 9, #, *, A, B, C, and D. If the received tones do not match the string, the greeting is replayed from the start, except in the special case of a string beginning with 'F', which is described above.

tmsg_phone
The telephone number of the person receiving the message. You can specify the phone number using the value in the <prefix>_phone variable in the Search_Directory action. The action uses this variable to build a system greeting. If you do not specify a value for this variable, the action uses a greeting without a number.

tmsg_page_type
The type of paging system the action uses when it stores the message. The types of paging systems are:

- V The action calls the number you specify in the tmsg_page_phone variable and leaves a recorded message.
- D The action calls the number you specify in the tmsg_page_phone variable and puts the digital information in the tmsg_page_data variable.

tmsg_page_phone
The telephone number of the recipient's paging system. If you do not specify a value for this variable, the action does not page.
tmsg_page_data
The digital information to send to the digital paging system you specify.
You must specify a value for this variable if you specify a value of ‘D’
for the tmsg_page_type variable.

tmsg_grtg_exit
The keys the caller can press to exit the action using the Greeting exited
return code.

Comments
• The Take_a_Message action uses several prerecorded voice segments. The
  names of these segments all begin with tmsg_. These segments are stored in
  the system voice segment database.

  You may rerecord these segments or you can override them by recording them
  in the voice application or common voice segment databases. Use the Voice
  Segment Editor for application SYSTM to find the text of each segment. If you
  reword the segments, make sure that you do not change the meaning or intent
  of the segments so that the segments become misleading.

  A caller may interrupt the greeting by pressing any key. This will jump to the
  prompt tone for recording the caller’s message, and will not exit the action. If
  the key, or key sequence, pressed is the same as defined in the tmsg_grtg_exit
  variable, then the action will return the Greeting exited return code.

  An example of the Take_a_Message action appears in the TAKE sample
  application.

  If the caller presses a pushbutton to indicate the end of a message, the action
  plays a segment asking what the caller wants to do. The caller can:
  – Cancel the message.
  – Rerecord the message. The action repeats the recording process.
  – Leave the message.
  – Play the message. The action plays the message the caller recorded and
    repeats the instructions.
  – Continue recording. The action resumes recording the message.

  In order to use the message paging facility, you need to set up a dedicated
  telephone line for paging requests. This is done as part of the Voice
  Messaging feature configuration which is described in the IBM CallPath
  DirectTalk/2 Installation Guide

  If the paging facility is in operation, but the telephone being paging is engaged,
  the system will wait for a period of time before trying again. The default value
  for this delay is 5 seconds. You can change the default value by adding the
  following line into the MBSRV.CFG configuration file, directly before the
  MAILBOX_VOICE_LINE entry:

  INTER_CALL_DELAY=n

  where n is the delay time in seconds.

  If the paged telephone is still engaged, the request is cancelled.

  If a caller hangs up whilst the greeting is being played, and your DirectTalk/2
  system is using silence to detect a hang-up tone, it is possible that a silent
  message will be stored in the mailbox.
**Update_Record**

Use the Update_Record action to replace the data in an existing record in a DirectTalk/2 application database with new data.

**Parameters**

**Database server name (Input, Required, Default=none)**
The network name of the database server that the action uses to access the application database.

**File name (Input, Required, Default=none)**
The name of the application database in which to update the record.

**Record key (Input, Required, Default=none)**
The key for the record the action should update. This value must be the same length as the key length defined when the database was created.

**Record data (Input, Required, Default=none)**
The data the action should use to replace the current data. The data can be of variable length. The maximum length of the data is the record length defined when the database was created. If the length of the data exceeds the defined record length, the system will log an error message.

**Return Codes**

**Record updated (RC=0)**
The action replaced the data in the record.

**Record not found (RC=1)**
The action could not find the record in the database.

**Error (RC=2)**
An error condition occurred. You specified an incorrect database server name or file name, or the record data is too long.

**Record locked (RC=3)**
The record you want to update is currently locked.

**System Variables**

**close_database (Input)**
To close all databases after updating a record with this action, set the close_database variable to Y or y. Note that this option may cause severe performance degradation. The default setting for close_database is N or n.

**db_record_wait (Input)**
If the first character of the db_record_wait variable is set to Y or y, and the record to be deleted is currently locked by another session, the Update_Record action waits until the lock is freed before deleting the record.

If the first character of the db_record_wait variable is not set to Y or y and the record to be deleted is currently locked by another session, the Update_Record action does not delete the record and returns the Record locked return code.
If the record to be deleted is not currently locked by another session, the Update_Record action updates the record regardless of the value of the db_record_wait variable.
VR_Get_String

**Note:** You must have the optional Voice Recognition Feature to use this action. Before using this action, refer to “VR_Get_String and VR_Get_Yes_No” on page 117.

Use the VR_Get_String action to enable the application to recognize one or more single words spoken by the caller, and verify that the recognized words are the ones that the caller actually said. VR_Get_String initiates a dialog with the caller to resolve ambiguous responses, so that when it returns, the response data has been verified by the caller.

**Parameters**

**Minimum words (Input, Required, No Default)**
The minimum number of words the caller must say.

**Maximum words (Input, Required, No Default)**
The maximum number of words the caller can say. You must specify the maximum number of words to be equal to or greater than the minimum number of words.

If minimum words does not equal maximum words, a variable length string may be input, but the end of the string is indicated when the caller stops speaking, not when caller says “stop”, as in the Get_Voice_Resp action. (The description of how stop is handled is part of the explanation of the Ended by caller return code.)

**VR subvocabulary (Input, Required, No Default)**
The vocabulary set of valid, recognizable word responses. You can specify any vocabulary set defined for a particular language. See the DirectTalk/2 language information manual for the language you are using for the supported voice recognition vocabulary sets.

At a minimum, the subvocabulary specified here must contain the digits zero through nine, plus stop. If possible, it should also contain some nonsense string (as a “none of the above”, that indicates that the caller said a word that is not in the vocabulary). This action will not work properly unless the specified subvocabulary meets this condition.

**Yes/No subvocabulary (Input, Required, No Default)**
The vocabulary set of valid, recognizable word responses. You can specify any vocabulary set defined for a particular language. See the DirectTalk/2 language information manual for the language you are using for the supported voice recognition vocabulary sets. The subvocabulary must contain at least yes, no, and stop, rather than digits.

**Return Codes**

**Response okay (RC = 0)**
The caller spoke the words that satisfy the parameters you specified for the action and verified that the words were received correctly. The action assigns the letters that correspond to each word the caller spoke to the last_voice_resp system variable.
Too few words (RC-1)
The caller spoke fewer than the minimum number of words. The action assigns the characters for the words that the caller spoke up to this point to the last_voice_resp system variable.

Too many words (RC-2)
The caller spoke more than the maximum number of words. The action assigns the characters for the words that the caller spoke up to this point to the last_voice_resp system variable.

Ended by caller (RC-3)
One of two things happened:
- The caller said ‘stop’, and does not wish to start over.
- There was a successful or ambiguous recognition of a word in the vocabulary other than the digits zero through nine, oh, or stop. For example, help or cancel in a vocabulary that contains those words.

In either case, any digits spoken by the caller are stored in the last_voice_resp variable, and the character corresponding to stop or the unknown response is stored in the resp_term_char variable. This permits an application to determine the reason for this result and process accordingly. This is especially useful for providing application-specific help.

Re-enter string (RC-4)
The caller said “stop”, and wishes to start over. Typically, you might want to limit the number of times that a caller can request to start again. It is good practice to repeat the prompt that preceded this step to refresh the caller’s memory as to what is required.

No VR line (RC-5)
This could occur if the application is running on a voice line not configured for use with voice recognition or if there are fewer voice recognition lines than telephone lines and all the voice recognition lines are in use.

Parameter error (RC-6)
The value of one of the input parameters is not valid, or is missing. There will be a log message that describes the condition that caused this return code.

Entry failed (RC-T1)
The caller was unable to speak and verify a string of digits that meet the input criteria. There are a number of conditions that can cause this. For example:
- There were one or more ambiguous responses which could not be resolved.
- The caller made multiple mistakes such as speaking too softly and speaking over the digit prompt, and did not correct the behavior after help messages were played.
- During verification of the input, the caller indicated that the input as understood by the action was incorrect.

You can use the special step number (−1) as the next step to automatically repeat the previous voice logic module. The number of times the module can be repeated is set when it is played for the first
time (Play_Module). The action sets the \textit{entry\_fail\_flag} variable to 1 each time this return code is passed back. The value of this flag can be tested in a voice logic module to optionally play additional help to the caller.

The action assigns the characters for the responses that the caller spoke up to this point to the \textit{last\_voice\_resp} variable.

\textbf{Last repeat (RC\textsubscript{-T2})}

The action repeated the prompt a specified number of times (Play_Module action), but the caller did not respond completely. The voice logic module defines the number of repeats that are permitted before the action returns this return code. The action places the caller’s response prior to this return code in the \textit{last\_voice\_resp} global system variable.

\textbf{Caller hung up (RC\textsubscript{-HUP})}

The caller hung up the phone.

\section*{System Variables}

\textbf{evr\_no\_beep (Input)}

If this variable is set to ‘yes’ there is no beep between input characters.

\textbf{last\_voice\_resp (Output)}

Contains the character corresponding to each digit in the string spoken by the caller. If the string is terminated by stop or an unknown word, only the digits spoken before the input was terminated are included. The termination character is not included. The action resets this variable each time it runs.

\textbf{resp\_term\_char (Output)}

Contains the character value corresponding to the word that caused the Ended by user return code.

\textbf{entry\_fail\_flag (Output)}

The action sets this variable to 1 when it passes back the Entry failed return code.

\section*{Comments}

The \texttt{VR\_Get\_String} action initiates a dialog with the caller to receive a string of digits. It manages the interaction and provides context-sensitive help where needed (for example, if the user makes a mistake, it will provide additional information specific to the particular mistake). After getting the string of digits, it will ask the caller to verify that the recognition is correct, and allow the caller to correct any ambiguous responses. Therefore, when the \texttt{Response\ ok\_ay} return code is passed back, it means that the caller has said the correct number of digits, and has verified that DirectTalk/2 recognized them correctly.

The action can recover from many mistakes that the caller may make (for example, speaking before the digit prompt), but the caller is not permitted to make mistakes indefinitely. The action returns the \texttt{Entry\ failed} return code if the caller makes too many mistakes, and does not correct the behavior.

If the caller wishes to re-enter the string, or makes a mistake, they can say “stop” and the action will return the \texttt{Re\textemdash enter\ string} return code. This allows your application to replay the prompt and give the caller another chance.
VR_Get_Yes_No

Note: You must have the optional Voice Recognition Feature to use this action.

Before using this action, refer to “VR_Get_String and VR_Get_Yes_No” on page 117.

The VR_Get_Yes_No action initiates a dialog with the caller to receive a response of “yes” or “no”. It manages the interaction and provides context-sensitive help where needed (for example, if the user makes a mistake, it will provide information specific to the particular mistake that was made). If the response from the caller is ambiguous, it attempts to determine what the correct response should be.

Parameters

Yes/No subvocabulary (Input, Required, No Default)
The vocabulary set of valid, recognizable word responses. You can specify any vocabulary set defined for a particular language. See the DirectTalk/2 language information manual for the language you are using for the supported voice recognition vocabulary sets. The subvocabulary must contain at least yes, no, and stop, rather than digits.

Return Codes

Caller said ‘yes’ (RC=0)
The caller spoke the word ‘yes’ in response to the prompt.

Caller said ‘no’ (RC=1)
The caller spoke the word ‘no’ in response to the prompt.

Other response (RC=2)
The caller spoke a word which was recognized as part of the subvocabulary, but is neither ‘yes’ nor ‘no’. The character which corresponds to the word spoken by the caller is in the ‘last_voice_resp’ system variable.

Response unresolved (RC=3)
The action was unable to resolve the response spoken by the caller. When this return code is passed back, you may want to play a help message to the caller and repeat your prompt.

Parameter error (RC=4)
The value of one of the input parameters is not valid, or is missing. There will be a log message that describes the condition that caused this return code.

No VR line (RC=5)
This could occur if the application is running on a voice line not configured for use with voice recognition or if there are fewer voice recognition lines than telephone lines and all the voice recognition lines are in use.

Time out (RC=T1)
The caller did not respond to the prompt. System variable ‘timeout_flag’ will be set to ‘1’. Application can use special step number (-1) to cause the prompt to be automatically replayed. The number of times to play the prompt was defined when the voice logic module was created.
**Last repeat (RC=T2)**

The action repeated the prompt a specified number of times (Play_Module action), but the caller did not respond completely. The voice logic module defines the number of repeats that are permitted before the action returns this return code. The action places the caller’s response prior to this return code in the last_voice_resp global system variable.

**Caller hung up (RC=HUP)**

The caller hung up the telephone.

**System Variables**

- **evr_beep_flag**
  
  If set to 'yes' there is no beep before a word is input.

- **last_voice_resp (Output)**
  
  Contains the character corresponding to the response spoken by the caller. The action resets this variable each time it runs.

- **timeout_flag (Output)**
  
  Set to ‘1’ when the Time out return code is passed back.

**Comments**

The VR_Get_Yes_No action initiates a dialog with the caller to receive a “yes” or “no” response. It manages the interaction and provides context-sensitive help where needed (for example, if the user makes a mistake, it will provide additional information specific to the particular mistake). After getting the response, it will ask the caller to verify that the recognition is correct, and allow the caller to correct any ambiguous responses. Therefore, when either the Caller said ‘yes’ or Caller said ‘no’ return code is passed back, it means that the caller has said the correct response, and has verified that DirectTalk/2 recognized it correctly.

The action can recover from many mistakes that the caller may make (for example, speaking before the prompt), but the caller is not permitted to make mistakes indefinitely. The action returns the Response unresolved return code if the caller makes too many mistakes, and does not correct the behavior.

For additional help on use and configuration, see “VR_Get_String and VR_Get_Yes_No” on page 117.
Wait_for_Call

Use the Wait_for_Call action to wait for and answer incoming calls.

Parameters

Number of rings (Input, Optional, Default='1')
The number of consecutive rings the Wait_for_Call action receives on the application telephone line before answering the telephone. If you set this parameter to a number less than 1, the action will answer on the first ring.

Time to wait for call (Input, Optional, Default='0')
The time, in minutes, the action waits for a call. If you do not specify a time period or if you specify a value of 0, the action waits for a call indefinitely.

Return Codes

Phone answered (RC=0)
The action answered the telephone after the number of rings you specified.

Wait expired (RC=1)
The amount of time you specified for the action to wait for a call expired before the action received a call. The telephone remains on hook.

Application quiesce (RC=2)
A request has been received to stop the application.

Comments

- Use the 'Time to wait for call' parameter in situations where you need to perform intermittent processing during periods of inactivity. For example, many host systems will log off a user (in this case, your voice application) who does not perform some activity within a certain period of time. You can use the Time out parameter to ensure that the host system does not log off your application during times when no one is calling.

  Note: The use of the 'Time to wait for call' parameter in applications that are to be used on more than one telephone line can put a heavy demand on system resources as multiple timeouts will need to be handled very close together. For this reason you should only use this parameter if it is needed to enable the application to run successfully, as in the instance described above.

- When you begin the voice application, the application telephone line is initialized to the default state specified during configuration (usually on hook or ready to answer a call). When this action answers a call, the telephone line is set to busy. When the application finishes processing or the caller hangs up, use the Hang_up_Phone action to hang up the telephone before using the Wait_for_Call action again.

- Upon execution, the Wait_for_Call action tests the stop_application flag. If a quiesce is indicated by the flag, the Wait_for_Call action immediately exits with the Application Stop return code. If a quiesce is not indicated by the flag, the Wait_for_Call action continues processing.

  You cannot execute a Wait_for_Call action during a call without hanging up on the caller. The stop_application flag therefore has no meaning in this situation and is ignored.
• The Wait_for_Call action stores any received ANI or DNIS digits in the system global variable ANI_DNIS_data.

• If your DirectTalk/2 system is configured for a T1 connection then the Number of rings parameter is accepted as 1, regardless of the value specified in the Wait_for_Call action.

• For analog lines, if the time required for the specified number of rings is greater than the timeout value, the telephone will not be answered by DirectTalk/2 because the timeout will occur first. The number of rings should be set to either 0 or 1 to avoid this problem.

• With the Dialogic D/41D hardware in particular, it is possible that an incoming call is not answered if the call has been ringing for the ‘Number of rings’ before the Wait_for_Call action is called. There are a number of actions developers can take to ensure that this does not happen:
  – If the number of rings before answer is always one, Set the Telephony Server, Network Interface parameter, ‘Interring Delay’ to 1. (This makes each ring appear as a new call.)
  – If the number of rings before answer needs to be greater than one, ensure that the application is not in the on-hook state with no Wait_for_Call active for the time required for the number of rings. If the application may be doing processing for an extended period of time, use the Hang_Up_Phone action, with OffHook set to ‘yes’, to take the phone off hook, thus preventing incoming calls.
Wait_for_Hang

Use the Wait_for_Hang action to monitor a call and detect when the caller hangs up. If the call is a referral call, this action will monitor the agent and the caller and will detect hang-up from either party.

Return Codes

**Multiple Hangups (RC=0)**
Disconnect has been detected from more than one party in the extended call.

**Timeout (RC=1)**
The time specified in the system variable `timeout_value` has expired.
The `timeout_value` can be set with either the Set_Timeout action, or with the Play_Module action.

**Agent hung up (RC=T2)**
The voice system detected that the agent disconnected.

**Caller hung up (RC=HUP)**
The voice system detected that the caller disconnected.

System Variables

The system variables and their settings after this action are discussed in “E1 Extend Call System Variables” on page 81.

Comments

- Control remains with this action until at least one party in the call hangs up, or the time out expires.
- If the call is not a referral call, only **Caller hung up (RC=HUP)** and **Timeout (RC=1)** can be returned.
- The values returned in the system variables depend on the mode of operation of the system as shown in Table 9 on page 83.
**Wait_Scr_Update**

Use the *Wait_Scr_Update* action to perform one search per second for the number of seconds you specify until the indicated text appears on the screen. You must have the optional Communications Feature to use this action.

You must specify the string of characters you want the action to wait for. You may also specify the starting row and column where the action is to search for the data.

The action searches from the starting row and column to the end of the screen. You can specify row 1, column 1 to see if a string appears anywhere on the screen.

**Parameters**

**Text to wait for (Input, Required, Default=none)**

The text you want the action to wait for on the virtual screen.

**Number of seconds (Input, Required, Default=none)**

The number of seconds you want the action to wait for the text to appear on the virtual screen.

**Starting row (Input, Required, Default=’1’)**

Specifies the row in the field in which to begin searching for the information.

**Starting column (Input, Required, Default=’1’)**

Specifies the column in the field in which to begin searching for the information.

**Return Codes**

**Text found (RC=0)**

The action found the text.

**Time out (RC=1)**

The text you specified did not appear on the screen in the amount of time you specified. This return code may indicate an unexpected host screen or that the host is hung up.

**Not connected (RC=2)**

The action could not access the screen because there is no session currently active with the host application. This return code can indicate that the host is currently not running.

**Session error (RC=3)**

The action encountered a problem with the host application session. This return code can indicate that the host is currently not running.

**Session disconnected (RC=4)**

The action could not retrieve the information because the session with the host has been disconnected. This return code can indicate that the host is currently not running.
System Variables

last_found_row
The starting row on the screen of the found text. This variable is only updated when RC=0.

last_found_col
The starting column on the screen of the found text. This variable is only updated when RC=0.

Comments

- You must establish a session with the host application using the Connect_Scr action prior to using this action.
- This action distinguishes between upper and lower case characters.
- You can use this search action to wait for a specific screen to be presented by the host application. For example, during the logon process, the order of screens may vary.
Appendix A. Sample Voice Application Form

This sample form consists of five parts.

- Use Part 1 (Table 23 on page 298) of the design form to define the purpose and tasks of the voice application.
- Use Part 2 (Table 24 on page 299) of the design form to specify the steps for the voice program.
- Use Part 3 (Table 25 on page 300) of the design form to specify the voice logic modules you will need.
- Use Part 4 (Table 26 on page 301) of the design form to specify the name and the text of the voice segments for the application.
- Use Part 5 (Table 27 on page 302) of the design form to specify the name and the text of the text segments for the application.
<table>
<thead>
<tr>
<th>Table 23. Application Design Form</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Part 1: Application Design</strong></td>
</tr>
<tr>
<td>Application name:</td>
</tr>
<tr>
<td>Designer:</td>
</tr>
<tr>
<td>Purpose:</td>
</tr>
<tr>
<td>Intended callers:</td>
</tr>
<tr>
<td>Application languages:</td>
</tr>
<tr>
<td>Input from callers:</td>
</tr>
<tr>
<td>&lt;_&gt; Pushbutton only</td>
</tr>
<tr>
<td>&lt;_&gt; Speech only</td>
</tr>
<tr>
<td>&lt;_&gt; Both</td>
</tr>
<tr>
<td>&lt;_&gt; None</td>
</tr>
<tr>
<td>Types of calls:</td>
</tr>
<tr>
<td>&lt;_&gt; Incoming only</td>
</tr>
<tr>
<td>&lt;_&gt; Outgoing only</td>
</tr>
<tr>
<td>&lt;_&gt; Both</td>
</tr>
<tr>
<td>Accept messages from callers:</td>
</tr>
<tr>
<td>&lt;_&gt; Yes</td>
</tr>
<tr>
<td>&lt;_&gt; No</td>
</tr>
<tr>
<td>Optional features:</td>
</tr>
<tr>
<td>&lt;_&gt; Voice Messaging</td>
</tr>
<tr>
<td>&lt;_&gt; Voice Recognition</td>
</tr>
<tr>
<td>&lt;_&gt; Host Communications</td>
</tr>
<tr>
<td>&lt;_&gt; Text-to-Speech</td>
</tr>
<tr>
<td>Step</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Go to Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 (T1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 (T2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 (HUP)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 25. Application Design Form (Voice Logic Modules)

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Type</th>
<th>Operand 1</th>
<th>Cond</th>
<th>Operand 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part 3: Voice Logic Modules
<table>
<thead>
<tr>
<th>Voice Segment Name</th>
<th>Voice Segment Text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Text Segment Name</td>
<td>Text Segment Text</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Appendix B. Sample Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MENU, Get Date and Time Application</td>
<td>305</td>
</tr>
<tr>
<td>CALC, Audible Calculator Application</td>
<td>305</td>
</tr>
<tr>
<td>TAKE Application</td>
<td>305</td>
</tr>
<tr>
<td>GET Application</td>
<td>305</td>
</tr>
<tr>
<td>PAGE Application</td>
<td>306</td>
</tr>
<tr>
<td>VRTTS, Voice Recognition Application</td>
<td>307</td>
</tr>
<tr>
<td>WMSCH, Employee Scheduling Application</td>
<td>307</td>
</tr>
<tr>
<td>HOST, Host Application</td>
<td>309</td>
</tr>
<tr>
<td>Sample ADSI Application</td>
<td>311</td>
</tr>
<tr>
<td>TRAN, Extend Call Application</td>
<td>311</td>
</tr>
</tbody>
</table>
Appendix B. Sample Applications

This appendix contains brief descriptions and information about the sample applications that are shipped with DirectTalk/2. You may want to print the sample applications to use as a reference while reading this appendix.

MENU, Get Date and Time Application

This application contains all the elements of the TIME application that is created in Chapter 4, “Creating a simple voice application” on page 13, together with a link to the CALC sample application.

The REXX source of this application is also included with DirectTalk/2

CALC, Audible Calculator Application

This application prompts the caller for numbers and operations, computes the answer, and tells the caller the result.

The REXX source of this application is also included with DirectTalk/2

TAKE Application

This application allows a caller to leave a recorded message, using the Voice Messaging feature. It demonstrates the preparation for and the usage of the Take_a_Message system action. Before running the application, the Mailbox Manager has to be used to build a voice messaging directory of mailbox users.

The overall structure of the application is as follows:

1. Wait for a call.
2. Welcome the caller and ask for the extension of the party being called to be entered as DTMF tones. Note that a more advanced application could use voice recognition to obtain the extension. The party being called must have been defined earlier as a mailbox user via the Mailbox Manager.
3. The voice messaging directory is searched for the entry corresponding to the phone number entered. From this directory entry the mailbox number in which to store the message is obtained. The id of any personal greeting and recorded name is also obtained.
4. The user greeting of the party being called is played, or the system greeting with the recorded name is played if the user greeting does not exist. The caller is then then able to leave a recorded message.

GET Application

This sample application allows a caller to retrieve any new or old messages from a mailbox, using the Voice Messaging feature. It demonstrates the preparation for and the usage of the Get_Messages system action.
Before running the application, the Mailbox Manager has to be used to build a voice messaging directory of mailbox users, and must include an entry for the user retrieving messages.

The overall structure of the application is as follows:

1. Wait for a call.
2. Welcome the caller and ask for the extension of the mailbox used to be entered as DTMF tones. Note that a more advanced application could use voice recognition to obtain the extension. The user must have been defined earlier as a mailbox user via the Mailbox Manager.
3. The voice messaging directory is searched for the entry corresponding to the phone number entered.
4. The caller is asked to enter the password for the entry as DTMF tones and the password entered is checked. The password was defined when the directory entry was created or updated via the Mailbox Manager (or updated by the user via an application such as in “PAGE Application”).
5. The messages are played back to the caller. The Get_Messages action being used allows any messages to be manipulated. For example, a message can be deleted, replayed or saved.

PAGE Application

This sample application allows a caller to change information in the voice messaging directory. The information that can be changed consist of the password and the paging data, such as the paging type and phone. It demonstrates the preparation for and the usage of the Put_User_Info system action.

Before running the application, the Mailbox Manager has to be used to build a voice messaging directory of mailbox users, and must include an entry for the user updating the information.

The overall structure of the application is as follows:

1. Wait for a call.
2. Welcome the caller and ask for the extension of the mailbox used to be entered as DTMF tones. Note that a more advanced application could use voice recognition to obtain the extension. The user must have been defined earlier as a mailbox user via the Mailbox Manager.
3. The voice messaging directory is searched for the entry corresponding to the phone number entered.
4. The caller is asked to enter the password for the entry as DTMF tones and the password entered is checked. The password was defined when the directory entry was created or updated via the Mailbox Manager (or updated by the user by a previous run of the PAGE example).
5. The caller is asked to enter as DTMF tones the paging information being updated and the update of the directory entry is then performed.
VRTTS, Voice Recognition Application

This sample application allocates separate telephone lines for voice response, and tests the responses for all valid values using a series of Compare_Chars actions. Alternatively, the Branch action could be used.

WMSCH, Employee Scheduling Application

This sample application allows people who work in a business such as a fast-food restaurant or retail store to call and retrieve information about their work schedules. They can access information for the current week or for either of the next two weeks. If employees find a problem with their schedule, the application transfers them to the manager in charge of the schedules. Callers with rotary phones can use this application via the DirectTalk/2 Voice Recognition Feature. (You can run the application even if you have not installed the Voice Recognition Feature.)

If the manager is not available, the employee can leave a message. The manager has a separate, password-protected menu for changing employee schedules and adding and deleting employees from the schedule database. The manager’s menu can also be used to listen to the recorded messages.

The sample application illustrates the use of the following DirectTalk/2 features:

- Voice messaging
- Voice recognition
- Local database actions
- String-processing actions
- Statistics actions
- Linking to subapplications
- Processing a multilevel menu
- Transferring a call
- Sharing voice segments and voice logic modules

This application contains the following five DirectTalk/2 voice programs:

**WMSCH**
The initial application. It retrieves employee schedules and processes the manager menu using tone input.

**WMREC**
The subapplication that retrieves the employee schedule using voice recognition instead of tone input.

**WMSSN**
The subapplication used to prompt the manager to input a social security number.

**WMUPD**
The subapplication used to update the schedule database when the manager wants to make a change.

**WMCOM**
The application that contains common voice logic modules and voice segments that are shared by WMSCH, WMREC, WMSSN, and WMUPD.
To run this sample application using the Debug utility, you must add the following statements to your Voice Application Developer Setup:

- `cmn_logic_appl` with a value of WMCOM
- `cmn_logic_srvr` with a value of your network name
- `cmn_segmt_appl` with a value of WMCOM
- `cmn_segmt_srvr` with a value of your network name

The employee schedules are kept in the following three DirectTalk/2 databases:

**WEEK1.SCH**
Contains the data about the current week

**WEEK2.SCH**
Contains the data about the next week

**WEEK3.SCH**
Contains the data about the following week.

Each database contains start and end times for each employee for one week. The key is the employee’s social security number. When a record is retrieved, it appears in the application with the following format:

<table>
<thead>
<tr>
<th>Column</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Social security number</td>
</tr>
<tr>
<td>10</td>
<td>Employee name as text</td>
</tr>
<tr>
<td>40</td>
<td>Start time for day 1 (HHMM)</td>
</tr>
<tr>
<td>44</td>
<td>End time for day 1</td>
</tr>
<tr>
<td>48</td>
<td>Start time for day 2</td>
</tr>
<tr>
<td>52</td>
<td>End time for day 2</td>
</tr>
<tr>
<td>...</td>
<td>Start and end times for days 3 through 7</td>
</tr>
</tbody>
</table>

The database also contains the dates for the week being processed. They are in a record with a social security number of ********. This record has the same basic format as the others, but it contains dates (YYYYMMDD) instead of start and end times.

When running this sample application in production, you should specify that all of the linked applications be preloaded by DirectTalk/2. Preloading applications improves performance of the links and reduces the amount of memory required by the system. The VSEXEC.CFG entry for this sample application is:

```
TYPE=EXECUTOR_START
   NAME=SCHED1 STARTUP_FILE=SCHED.CTL MSG_LANG=E TR_ADAPTER=255
   RESTARTS=10; 
TYPE=PRELOAD APPL=WMSCH; 
TYPE=PRELOAD APPL=WMREC; 
TYPE=PRELOAD APPL=WMUPD; 
TYPE=PRELOAD APPL=WMSSN; 
```

**Note:** Since the WMCOM application is only used for storing common voice segments and voice logic modules, there will never be a link to WMCOM. You do not need to preload the WMCOM application.
This sample application uses the new call detail statistics and also keeps track of a piece of unique statistical data. This method was chosen so that the call detail statistics collection could be turned off by simply changing the application variables, then stopping and starting the application. This sample enables the call detail statistics by using the Application Variables option to set the cs_record variable to a value of 'Y'.

To use the new termination report to get information about where in the application callers hang up, the WMSCH application sets the cs_procedure variable at the beginning of major sections of the voice program (see step 30 in the sample application). After the employee chooses to listen to the schedule, DirectTalk/2 sets the cs_procedure variable to employee_sched. If the caller hangs up during this processing, the statistics file contains the following as part of the statistics record:

```cs_procedure(employee_sched)```

The call-transfer processing uses variables to specify the key sequences that need to be dialed when the application is run using a ROLM switch. The key sequences may differ for other switches. You probably need to specify new key sequences to run the sample application in your office. Select Application Variables in the Options pull-down to change the following information for the WMSCH application and the WMREC application:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>caller_on_hold</td>
<td>&amp;,*9</td>
<td>Flash-hook, then put the caller on hold.</td>
</tr>
<tr>
<td>mgr_phone</td>
<td>*7nnnn</td>
<td>Transfer the call to the manager’s extension. ‘*7’ is the key sequence for a transfer. Replace the ‘nnnn’ with an extension that represents the manager’s phone.</td>
</tr>
<tr>
<td>opr_phone</td>
<td>&amp;,*7nnnn</td>
<td>Perform a blind transfer to the operator’s extension. ‘&amp;,*7’ does a flash-hook and a transfer. Replace the ‘nnnn’ with an extension that represents the operator.</td>
</tr>
<tr>
<td>pickup_caller</td>
<td>&amp;,*1</td>
<td>If the manager does not respond, get the caller back on the line before taking a message.</td>
</tr>
</tbody>
</table>

In order to use the messaging capabilities used in the sample application, you must install the DirectTalk/2 Voice Messaging Feature. In addition, you must build a directory containing an entry for a user with a user identification of 1000 (or whatever value you use for mgr_mbox_number). Refer to the IBM CallPath DirectTalk/2 Administrator’s Guide for information on how to build a directory.

**HOST, Host Application**

The sample host application illustrates the DirectTalk/2 host interface actions, including the Switch_Screen action. This action enables an application to work with multiple host sessions. In this example, the application works with a 3270 host session and an AS/400 host session.
The application uses the updated Wait_for_Call action to check if the host is still active when a timeout occurs.

The sample application also uses transaction messaging. Through the use of transaction messaging, DirectTalk/2 can help you improve service to customers who call during peak calling periods or after normal business hours. Using this technique, you can design an application that records voice messages from customers and assigns each message a data key or identifier. This key provides an easy and efficient way of notifying you of these messages and retrieving them.

For example, suppose a customer wants to discuss a particular invoice with a service representative. When the customer calls, however, all of the service representatives are busy handling other calls. Instead of putting the customer on hold until a service representative is available, your DirectTalk/2 application can handle the call. The application can:

1. Request the customer’s order number
2. Locate the appropriate record in the host database
3. Record a voice message from the customer
4. Assign a unique data key to the message

The application can then update the record in the host database to indicate that a message is waiting.

The next available service representative receives a prompt with the host record that indicates that the customer left a message. When the service representative looks at the host record, the host database sends the data key to the DirectTalk/2 application, which then calls the representative and plays the message when the representative answers the phone.

To design a DirectTalk/2 application that uses transaction messaging, use the following voice-related actions:

**Delete_Voice**
To delete voice messages after they are received

**Play_Voice**
To retrieve and play voice messages

**Record_Voice**
To record voice messages and assign a data key to each message

This sample application uses these actions to perform transaction messaging.

The application also contains several comment statements that identify parts of the application that you need to expand to use this application in production. However, you can use the application as is by using the debug utility in the Voice Application Developer. Many of these comments relate to exception handling, such as what should be done if the host goes down. You should determine the correct implementation based on your environment.

The following are suggested actions for host error conditions:

- The host goes down:
  1. Disconnect_Scr.
  2. Connect_Sceen.
– If the Connect_Scr action is successful, reset and continue the transaction.
– If the Connect_Scr is unsuccessful, notify the caller of the failure and transfer to a service attendant.

• Emulator unavailable:
  – Notify the caller of the problem and transfer the caller to a service attendant.

• Screen timeout:
  – Retry the operation.
    – If the retry is successful, continue the transaction.
    – If the retry is unsuccessful, run the Disconnect_Scr action and transfer the caller to a service attendant.

Sample ADSI Application

The sample SDC script source file MAIL.SDC is a text mailbox application. Also included are the other files required to run the application. They are the trexx file MAIL.TRX, together with the two command files MAILCOPY.CMD and MAILDEL.CMD that are called by the program to handle the database, and the text data file MAIL.TXT.

The example FDM script source file SFDM.FDM, together with its downloading trexx program SFDM.TRX, is included with DirectTalk/2. When SFDM.FDM has been compiled and downloaded to a telephone, an incoming call causes the telephone to display “CALL ARRIVED”, while dialing a number causes the telephone to display “DIALING OUT”.

TRAN, Extend Call Application

This is a simple Call Transfer application.

The application performs the following tasks:

1. Asks the caller to enter the agent’s telephone number and then phones the agent
2. Announces the caller to the agent and asks them how to proceed with the call. The agent can choose to either take the call as a transfer or referral call, or they can refuse to take the call.
3. If the agent refuses the call the application returns to the caller and asks them to supply the telephone number of another agent, or to end the call.
# Part 7. Country Programming Considerations

## Appendix C. Programming Considerations for Australia
- Physical Requirement ........................................ 317
- Ringing Signal Detection ................................... 317
- Identification Message (Incoming) ....................... 318
- Identification Message (Outgoing) ....................... 318
- Decadic Dialling ........................................... 318
- Repetition of a Call ...................................... 318

## Appendix D. Programming Considerations for Belgium
- Ringing Signal Detection ................................... 319
- Identification Signal ....................................... 320
- Duration of the Loop Condition ......................... 320
- Dial Tone Detection ........................................ 320
- Dialing International Numbers ......................... 321
- Pulse Dialing ............................................. 321
- DTMF Tone Dialing ....................................... 321
- Repetition of a Call ...................................... 322

## Appendix E. Programming Considerations for Denmark
- Identification Signal ....................................... 323
- Duration of the Loop Condition ......................... 324
- General Outbound .......................................... 324
- Dial Tone Reception ....................................... 324
- Dialing with Loop Pulsing ............................... 324
- Dialing with DTMF Pulses ............................... 325
- Duration of the Call ...................................... 325
- Repetition of a Call ...................................... 325

## Appendix F. Programming Considerations for France
- Ringing Signal Detection ................................... 327
- Identification Signal ....................................... 328
- Duration of the Loop Condition (Inbound) ............ 328
- Answering Incoming Calls ................................ 329
- Dial Tone ................................................... 329
- Dial Tone Reception ....................................... 330
  - Detection of Called Party’s Answer ................... 330
- Loop Pulse Dialing ....................................... 330
- Dialing with DTMF Pulses ............................... 331
- Switching of Transmission Equipment ................ 331
- Duration of Loop Condition ............................. 331
- Repetition of a Call ...................................... 332

## Appendix G. Programming Considerations for Germany
- Ringing Signal Detection ................................... 333
- Identification Signal ....................................... 333
- Duration of the Loop Condition ......................... 334
- General ...................................................... 334
- Dial Tone Reception ....................................... 335
- Dialing with Loop Pulsing ............................... 335
Automatic Dialing .................................................. 358

Appendix M. Programming Considerations for South Africa ........ 359
Ringing Detection .................................................. 359
Automatic Answering .............................................. 360
Call Termination ................................................... 360
Maintaining Connection .......................................... 360
Recording .......................................................... 361
Timed Break (Flash) Signal ....................................... 361
Dialing ............................................................. 362
Pulse Dialing ........................................................ 362
Multifrequency Dialing (DTMF) .................................. 362
Automatic Dialing .................................................. 363
Remote Supervisory Functions .................................... 363
Outbound Dialing Exchange Delays ............................... 363
Repertory Dialling .................................................. 364

Appendix N. Programming Considerations for Spain .................. 365
Ringing Signal Detection ........................................... 365
Identification Signal ................................................ 366
Duration of the Loop Condition .................................... 366

Appendix O. Programming Considerations for Sweden .................... 369
Answer Function ..................................................... 369
Ringing Signal ....................................................... 369
Voice-activated Circuits ............................................ 370
Dial Tone ............................................................ 371

Appendix P. Programming Considerations for Switzerland ............ 373
Inbound Calls ........................................................ 373
Outbound Calls ..................................................... 373

Appendix Q. Programming Considerations for the United Kingdom .. 375
Two-way Recording .................................................. 375
Readiness ........................................................... 376
Exhaustion of Recording Medium .................................. 376
Paytone ............................................................. 376
Auto-answering ....................................................... 377
Answertone .......................................................... 377
Auto-clearing ........................................................ 377
Call Initiation ........................................................ 378
    Proceed indication conditions ................................... 378
Call Progress Monitoring ........................................... 379
Repeat Attempts ..................................................... 380
Appendix C. Programming Considerations for Australia

This appendix discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Australia.

DirectTalk/2 holds Austel permit number A91/56B/0373 for connection to the Australian telephone network. The following requirements are based on the Austel Specification TS002. References to this specification appear in bold. For example, 5.5.7.3.

Warning: It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the Austel certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide.

The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

Table 30: Inbound and Outbound Calls

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Ringing Signal Detection&quot;</td>
<td>&quot;Identification Message (Outgoing)&quot; on page 318</td>
</tr>
<tr>
<td>&quot;Identification Message (Incoming)&quot; on page 318</td>
<td>&quot;Decadic Dialling&quot; on page 318</td>
</tr>
<tr>
<td></td>
<td>&quot;Repetition of a Call&quot; on page 318</td>
</tr>
</tbody>
</table>

Physical Requirement

The supplied label is to be attached to the exterior of the PS/2 in a visible position. If no label is attached please contact your IBM representative for supply. The text of the label reads:

![Label Text]

Austel Permit A91/56B/0373
Ringer Equivalent Number = 0.7

Ringing Signal Detection

5.5.7.2

Incoming calls must be answered in not less than 2 seconds from arrival of a ringing signal. To meet this requirement, the 'number of rings' parameter in a Wait_for_Call action must be set to a value of 2 or more.
Identification Message (Incoming)

5.5.7.2 (b)

When an incoming call is answered, the application must play an "appropriately worded voice message". To meet this requirement, a suitable Play_Module action must follow a Wait_for_Call action for the 'Answered' return code (RC=0). If it is expected that incoming calls will normally be machine-generated, the incoming message may be replaced by a 2100Hz tone of duration 3 seconds (a suitable voice segment may be recorded comprising this tone).

Identification Message (Outgoing)

5.5.7.6

When an outgoing call is answered, the application must play a "message to identify the caller". To meet this requirement, a suitable Play_Module action must follow a Place_a_Call action for the 'Answered' return code (RC=0). It is recommended that a 2-second pause should precede the message to allow for STD pips. If it is intended that the call will be answered by an automatic device, the message may be replaced by the relevant tone (modem or fax).

Decadic Dialling

Outward tone dialling is permitted for DirectTalk/2 in Australia, but not decadic (pulse) dialling with Dialogic Micro Channel hardware. Therefore you must not include a 'P' character in digit strings for the Place_a_Call or Put_Tone_String actions. You must not set the PR_INIT_TONE_TYPE parameter to 'P' (Pulse).

Repetition of a Call

5.5.7.3

If service tones are not detected (that is, in a Put_Tone_String action), only 3 call attempts may be made to the same number in succession (that is, 1 call plus 2 retries). The application must then delay for 30 minutes before the next retries (up to three in succession). This pattern of three attempts, 30-minute pause, may continue indefinitely.

If service tones are detected (i.e. Place_a_Call action) the same process applies but up to 10 calls may be placed in succession. This pattern of ten attempts, 30-minute pause, may continue indefinitely.
Appendix D. Programming Considerations for Belgium

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Belgium.

The following requirements are based on:

- RTT/RN/SP/204
  Réseau téléphonique automatique (PSTN).

- RTT/RN/SP/206
  Equipements auxiliaires spéciaux (répondeurs, envoyeurs automatiques, transmetteurs de signaux).

- PSTN-Compendium Compliance Test Description
  Conference of European Post and Telecommunication (CEPT), 1987.

Reference to these specifications appear in bold. For example: 6.2.3.

Warning: It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all applications comply with the Belgian RTT certification. IBM has merely assisted with the interpretation of these requirements in order to provide guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide. The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

Table 31. Inbound and Outbound Calls

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Ringing Signal Detection&quot;</td>
<td>&quot;Dial Tone Detection&quot; on page 320</td>
</tr>
<tr>
<td>&quot;Identification Signal&quot; on page 320</td>
<td>&quot;Dialing International Numbers&quot; on page 321</td>
</tr>
<tr>
<td>&quot;Duration of the Loop Condition&quot; on page 320</td>
<td>&quot;Pulse Dialing&quot; on page 321</td>
</tr>
</tbody>
</table>

Ringing Signal Detection

RTT 6.2.3

The closing of the loop (phone off hook) cannot start until detection of the second pulse of the ringing current. Calls should therefore not be answered on the first ring. The Wait_for_Call action in DirectTalk/2 allows an application developer to specify how many rings to wait before answering a call. When developing a voice application, specify a value of 2 for this parameter.
Identification Signal
RTT 6.4.1

When answering an incoming call, applications must present an identification signal in the form of a tone, or play a verbal announcement. To satisfy this requirement, applications must be designed to run the Play_Module action to play a greeting or prerecorded tone immediately after answering a call with the Wait_for_Call action.

Duration of the Loop Condition
RTT 6.5.1

For answering-recorder devices, the loop condition must not be maintained for more than 5 minutes after the start of the recording. This delay must be controlled by a timer.

This only applies to the recording phase of the session, not the interactive part. Any call to the Record_Voice action in a DirectTalk/2 application must set the Duration parameter to a value less than 300 seconds (five minutes).

For systems intended for recording emergency calls, a minimum record time of 95 to 120 seconds per message is required for safety reasons.

Dial Tone Detection
RTT 5.1.3

For automatic dialing, if dial tone is not received within 3 seconds, the line must be cleared.

You must define the dial tone as one of DTONE-L, DTONE-I, or DTONE-X using the Tone Detection function. Additionally, the Place_a_Call action must use the corresponding character (L, I, or X) at the appropriate place in the dialed string, usually as the first character. Your DirectTalk/2 system has been configured to meet timing requirements.

You may change the amount of time you wait for the dial tone and the amount of time for valid dial tone using the Max Dial Tone Wait and Valid Dial Tone parameters. If no valid dial tone is detected within the specified time limit, the Place_a_Call action returns the No dial tone return code. In response to this, your application must immediately execute the Hang_up_Phone action.

If a valid dial tone is detected, DirectTalk/2 will begin dialing.
Dialing International Numbers

RTT 5.2.2

Some old PABXs send an international dial tone after reception of the 00 international prefix. This dial tone comprises three frequencies (900 Hz, 1020 Hz, and 1140 Hz), each emitted with a duration of 330 ms, and must be detected before dialing starts.

The presence of the three frequencies is important; the order is not.

When dialing an international number, use a two-step process:

1. Use the Place_a_Call action to dial the international prefix. The presence of the international dial tone will cause the Operator Intercept return code to be passed back.
2. Use another Place_a_Call action to dial the rest of the phone number.

Pulse Dialing

RTT 5.3.2 and 5.3.3

At a typical frequency of 10 Hz, the break period must be 66% typical, with a range of 63 through 70%. Your DirectTalk/2 system has been configured to do this.

In general, when using pulse dialing, the interdigit delay must be greater than 400 ms. Equipment with automatic pulse dialing capabilities must have an interdigit delay of between 500 ms and 1000 ms. Your DirectTalk/2 system has been configured to meet this requirement. If you need to adjust the interdigit delay for pulse dialing, change the Pulse Interdigit Delay parameter. Refer to the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide for information about how to change telephony parameters.

DTMF Tone Dialing

RTT 5.4

The duration of each DTMF tone must be between 65 and 75 ms, and the delay between consecutive tones must be between 65 and 75 ms. Your DirectTalk/2 system has been configured to meet these requirements.

The time period from the last digit dialed until no answer occurs, at which time the line must be cleared, must not exceed 80 seconds. This time period corresponds to the number of rings before no answer is returned. Your DirectTalk/2 system is configured to meet this timing requirement. All applications must run the Hang_up_Phone action immediately after receiving the No ring return code from a Place_a_Call action.
Repetition of a Call

RTT 5.7

The wait time between successive attempts to call the same number must be:

- At least 5 seconds between the first and second attempts
- At least 1 minute between subsequent attempts

A maximum of four call attempts is allowed within 1 hour of the initial attempt (15 attempts are allowed for alarm systems).

A manual intervention is required before further call attempts are made to the same call address before the expiration of the 1-hour period from the first attempt.

DirectTalk/2 has no built-in features that enforce compliance with this requirement. It is up to the application designer to ensure that all outbound applications comply with these rules.
Appendix E. Programming Considerations for Denmark

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Denmark.

The following requirements are based on the Danish PTT specifications:

- Proposal for Compliance Test Description.
- Circular No 27A–Technical Requirements for Private Equipment Connected to the Public Switched Telephone Network.

References to these specifications appear in bold. For example, 6.4.

**Warning:** The DirectTalk/2 system has been certified for attachment to the Danish public switched telephone network (PSTN). It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the Danish PTT certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*. The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls.

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Identification Signal”</td>
<td>“General Outbound” on page 324</td>
</tr>
<tr>
<td>“Duration of the Loop Condition” on page 324</td>
<td>“Dial Tone Reception” on page 324</td>
</tr>
<tr>
<td></td>
<td>“Dialing with Loop Pulsing” on page 324</td>
</tr>
<tr>
<td></td>
<td>“Dialing with DTMF Pulses” on page 325</td>
</tr>
<tr>
<td></td>
<td>“Duration of the Call” on page 325</td>
</tr>
<tr>
<td></td>
<td>“Repetition of a Call” on page 325</td>
</tr>
<tr>
<td></td>
<td>“Identification Signal” on page 326</td>
</tr>
</tbody>
</table>

### Identification Signal

6.4

DirectTalk/2 applications must generate an announcement to indicate to the caller that the call has been answered. To do this, use the Play_Module action within 3 seconds of answering the call with the Wait_for_call action. The Wait_for_call action should answer the call after 1 ring by setting the Number of rings parameter to 1. The Phone answered return code should go immediately to a Play_Module action.
An identification signal in the form of a code signal must be made up of one or more frequencies in the range 800 Hz through 2200 Hz and must be emitted for a period of 3 or 4 seconds.

As an exception from this requirement, you can use DTMF signals as the identification signal.

### Duration of the Loop Condition

**6.5, 6.5.2**

If the caller has not responded to a prompt within 15 seconds, the DirectTalk/2 application must hang up the call. Currently, voice logic modules have a default timeout value of 10, with a repeat value of 3. To conform with this requirement, you must change these values. For example, change the timeout value to 5 seconds.

### General Outbound

**5.1, 5.1.3**

When making outbound calls, DirectTalk/2 must wait for dial tone to be present before dialing. You can alter the amount of time DirectTalk/2 waits for dial tone to be present by changing the Maximum Dial Tone Wait parameter value. Refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*.

### Dial Tone Reception

**5.2, 5.2.2**

The dial tone is a continuous AC voltage in the frequency range 400 Hz through 450 Hz, with a level of 0 to –30 dBm.

Dialing must begin no more than 2 seconds after the dial tone is received.

DirectTalk/2 has been configured to meet these requirements.

### Dialing with Loop Pulsing

**5.3, 5.3.2, 5.3.3**

When dialing with loop pulsing, the break period must be 56 ms through 80 ms, and the make period between two break pulses 27 ms through 41 ms.

You can alter these settings by changing the Pulse Dial Break and Pulse Dial Make parameter values, respectively. See the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*.

DirectTalk/2 has been configured to meet these requirements.

The duration of the interdigit delay tone must be in the range of 800 ms through 1000 ms.
You can alter this setting by changing the Pulse Interdigit Delay parameter value. See the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide.

DirectTalk/2 has been configured to meet these requirements.

Dialing with loop pulsing of a number with no more than 9 digits must last no more than 16 seconds.

DirectTalk/2 has been configured to meet this requirement. If you alter Pulse Dial Break, Pulse Dial Make, or Pulse Interdigit Delay, you must ensure that this requirement is still met by the new values.

To perform pulse dialing, set the Initial Tone Type parameter to Pulse. Additionally, DirectTalk/2 can be set to perform pulse dialing by including a ‘P’ in the dialed string in the Put_Tone_String or Place_a_Call actions. The character ‘T’ in the dialed string will set DirectTalk/2 to perform DTMF dialing.

### Dialing with DTMF Pulses

#### 5.4

The duration of the DTMF tone must be 65 ms through 105 ms, excluding rise time.

The interdigit delay must be such that the total time is greater than 200 ms. Total time is defined as:

\[ \text{rise time} + \text{emission} + \text{fall time} + \text{pause} \]

You can alter this setting by changing the Pulse Interdigit Delay parameter value. See the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide.

DirectTalk/2 has been configured to meet these requirements.

### Duration of the Call

#### 5.6, 5.6.2

If the caller has not responded to a prompt within 15 seconds, the DirectTalk/2 application must hang up the call. Currently, voice logic modules have a default timeout value of 10, with a repeat value of 3. To conform with this requirement, you must change these values. For example, change the timeout value to 5 seconds.

### Repetition of a Call

#### 5.7

The DirectTalk/2 application must disconnect (hang up) for 1 second between each successive call attempt.

The number of unsuccessful call attempts to one telephone number is limited to 10 attempts.
Private equipment that is designed for automatic start and performance of calls, but not designed to detect whether the correct connection has been obtained, must make no more than 10 automatic calls between each manual intervention in the operation.

Application developers must ensure that their application meets these call repetition limits.

Identification Signal

5.8

Equipment with an automatic calling function must play a prompt within 5 seconds from the end of the last dialed digit. The prompt must be repeated continuously until the called party has responded or disconnected.

Application developers must ensure that their application meets this call identification signal requirement.
Appendix F. Programming Considerations for France

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in France.

The following requirements are based on the following French RTT specifications:

- ST/PAA/TPA/AGH/1108
  Interface réseau analogique des matériels téléphoniques et télématicques.
- ST/PAA/TPA/AGH/1257
  Appel automatique.
- ST/PAA/TPA/AGH/1764
  Réponse automatique.

References to this specification appear in bold. For example, 6.2.3.

**Warning:** The DirectTalk/2 system has been certified for attachment to the French public switched telephone network (PSTN). It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the French RTT certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*. The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

**Table 33. Inbound and Outbound Calls**

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ringing Signal Detection” on page 328</td>
<td>“Answering Incoming Calls” on page 329</td>
</tr>
<tr>
<td>“Identification Signal” on page 328</td>
<td>“Dial Tone” on page 329</td>
</tr>
<tr>
<td>“Duration of the Loop Condition (Inbound)” on page 328</td>
<td>“Dial Tone Reception” on page 330</td>
</tr>
<tr>
<td></td>
<td>“Loop Pulse Dialing” on page 330</td>
</tr>
<tr>
<td></td>
<td>“Dialing with DTMF Pulses” on page 331</td>
</tr>
<tr>
<td></td>
<td>“Switching of Transmission Equipment” on page 331</td>
</tr>
<tr>
<td></td>
<td>“Duration of Loop Condition” on page 331</td>
</tr>
<tr>
<td></td>
<td>“Repetition of a Call” on page 332</td>
</tr>
</tbody>
</table>
Ringing Signal Detection

6.2.3

Calls must be answered between 5 and 15 seconds from the start of the ring signal.

In the special case of some automatic call rerouting apparatus, or the automatic switching to an operator position, this delay can be reduced to 1 second.

In the special case of remote control systems, this delay can be between 35 and 45 seconds in order to avoid intercepting calls that are not intended for the remote control system.

In DirectTalk/2 the amount of time from the beginning of the ring signal to the time the call is answered is controlled by the Number of rings parameter on the Wait_for_Call action. You must set this parameter to allow for the required amount of time after the start of the ring signal. For example, if the ring and the silence that follows take 2 seconds, you need to set the Number of rings parameter to at least 3 in order to meet the requirement that the phone be answered between 5 and 15 seconds after the start of the ring signal.

Identification Signal

6.4.1

On answering an incoming call, an application must play an identification signal in the form of either a tone, or a verbal announcement. The tone must be 2100 Hz, and must comply with CCITT V.25. A verbal announcement must indicate that the number called has actually answered the call.

To meet this requirement, you must run the Play_Module action immediately after answering the call. The voice segment that you play must contain a message or recorded tone.

Duration of the Loop Condition (Inbound)

6.5.1

The terminal equipment must automatically release the line at the end of the message and at least in one of the following cases:

- The called party did not send any signal for more than 3 minutes. DirectTalk/2 contains timer actions that allow you to keep track of how long your application is on the line.
- The busy tone, or the following tone is present on the line:

  | Frequency | 440 ± 15 Hz |
  | Level     | -10 through -25 dBm |
  | Cadence   | 500 ± 50 ms ON, 500 ± 50 ms OFF |

To detect this tone, configure the Tone Detection feature with the above definition as a HANGUP tone. You may also define the hangup pattern parameters as shown below.
Set this parameter:                      To this setting:

Hangup Minimum Silence               40
Hangup Maximum Silence               60
Hangup Minimum Nonsilence            40
Hangup Maximum Nonsilence            60
Hangup Repeat Count                  4

- You should also set the Silence Before Hangup parameter to a value of between 6 and 12.

Notes:

1. Before releasing the line, a ‘thank you’ message of about 5 seconds must be sent.
2. When the end of the recording medium is reached, a relevant tone or message must be sent. Your DirectTalk/2 system is already configured to do this.
3. Short silences (less than 5 seconds) can be foreseen in order to allow the reception of remote-controlled commands.
4. Remote-controlled machines must release the line no more than 40 seconds after the last message is played back, unless other commands are received.

Answering Incoming Calls

2.3.2., 2.3.3.

When the calling equipment is also performing the automatic answering function, incoming calls must be given priority.

Systems that cannot prevent automatic call attempts, even after detection of an incoming call or an ongoing connection, must be connected to outgoing lines only.

Dial Tone

PTT 5.1.3

All dial tones sent by the public network must be detected before starting or proceeding with dialing.

The dial tone wait time must be between 6 and 12 seconds. If no dial tone has been detected after that time, the line must be released.

Dialing must occur, at the latest, 3 seconds after the beginning of the dial tone. The dial tone validation signal must be between 1 and 2 seconds long, as long as the tone contains no interruptions longer than 30 ms.

You must define the dial tone as one of DTONE-L, DTONE-I, or DTONE-X using the Tone Detection function. Additionally, the Place_a_Call action must use the corresponding character (L, I, or X) at the appropriate place in the dialed string, usually as the first character.
Your DirectTalk/2 system has been configured to meet this requirement.

You may change the amount of time you wait for the dial tone and the amount of
time for valid dial tone using the Max Dial Tone Wait and Valid Dial Tone
parameters. If no valid dial tone is detected within the specified time limit, the
Place_a_Call action returns the No dial tone return code. In response to this, your
application must immediately execute the Hang_up_Phone action. If a valid dial
tone is detected, DirectTalk/2 will begin dialing.

For information about how to change the telephony parameters, refer to the section
on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation
Guide*.

---

### Dial Tone Reception

**PTT  5.1.3**

Terminal equipment calling automatically to systems with manual answering must
do one of:

- Send, no longer than 3 seconds after the end of dialing, a spoken message no
  longer than 3 minutes.
- Send, no later than 3 seconds after the end of dialing, a calling tone (as
  defined above). The tone should not be sent for more than 3 minutes.
- Send a spoken message no later than 1 second after reception of the called
  party’s answer.
- Transfer to a live operator’s position no later than 1 second after detection of
  the called party’s answer.

All these requirements can be met as long as your application runs a Play_Module
action (for spoken messages), Place_a_Call action, or Put_Tone_String action (for
transfers) immediately after the Wait_for_Call action returns the Answered return
code.

### Detection of Called Party’s Answer

The called party’s answer must be detected by the end of the ring back tone, as a
silence of between 4 and 5 seconds, even if a single tone, as short as 300 ms has
been received. Your DirectTalk/2 system has been configured to meet this
requirement.

---

### Loop Pulse Dialing

**PTT  5.3.2, 5.3.3**

Dialing pulses must meet the following characteristics:

- **Break Period**: 66 ± 7 ms
- **Make Period**: 33 ± 4 ms
- **Cycle Time**: 100 ± 10 ms

Your DirectTalk/2 system has been configured to meet this requirement.
Dialing with DTMF Pulses

PTT 5.4

For equipment with automatic dialing with DTMF pulses, the tone duration must be between 65 and 90 ms, with a pause of between 65 and 90 ms.

Your DirectTalk/2 system has been configured to meet this requirement. If your telephone network environment makes it necessary to change the configuration, the duration of the DTMF tone pulse is controlled by the DTMF Interdigit Delay parameter.

Loop interruptions used to cause register recall must meet the same requirements as dialing pulses, but with a duration of 270 ± 50 ms. Your DirectTalk/2 system has been configured to meet this requirement. If your telephone network environment makes it necessary to change the configuration, the duration interrupt that causes register recall is controlled by the Wink Length parameter.

The time period from the last digit dialed until no answer occurs, at which time the line must be cleared, must not exceed 90 seconds. This time period corresponds to the number of rings before no answer is returned.

To meet this requirement, your application must run the Hang_up_Phone action immediately after receiving the No answer return code from a Place_a_Call action.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide.

Switching of Transmission Equipment

PTT 5.5

Terminal equipment automatically calling to systems with automatic answering must generate, within 3 seconds after the end of dialing, a calling tone complying with CCITT recommendations V.25 or T.30.

If an application is intended for this type of environment, the tone must be recorded to comply with this requirement. Play the prerecorded tone, using a Play_Module action, immediately after receiving the Answered return code from a Place_a_Call action.

Duration of Loop Condition

PTT 5.6.1

The terminal equipment must automatically release the line at the end of the message and at least in one of the following cases:

- The called party did not send any signal for more than 3 minutes. DirectTalk/2 contains timer actions that allow you to keep track of how long your application is on the line.
The busy tone, or the tone already described in “Duration of the Loop Condition (Inbound)” on page 328, is present on the line. You may also define the hangup pattern parameters as shown below.

<table>
<thead>
<tr>
<th>Set this parameter:</th>
<th>To this setting:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hangup Minimum silence</td>
<td>40</td>
</tr>
<tr>
<td>Hangup Maximum silence</td>
<td>60</td>
</tr>
<tr>
<td>Hangup Minimum nonsilence</td>
<td>40</td>
</tr>
<tr>
<td>Hangup Maximum nonsilence</td>
<td>60</td>
</tr>
<tr>
<td>Hangup Repeat Count</td>
<td>4</td>
</tr>
</tbody>
</table>

You should also set the Silence Before Hangup parameter to a value of between 6 and 12.

---

**Repetition of a Call**

**PTT 5.7**

Unsuccessful or incorrectly dialed calls (wrong calls) can be repeated five times, with a delay of 1 through 12 minutes between successive attempts. If the expected connection has not been made after the sixth attempt, it must not be attempted again without manual intervention.

If the application can distinguish between wrong calls and correctly dialed calls, a new series of six call attempts can be made every hour, as long as the call is identified as being unsuccessful. However, if two wrong calls are made, that number must not be attempted again without manual intervention.
Appendix G. Programming Considerations for Germany

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Germany.

The following requirements are based on the German PTT specifications:

- **FTS 1TR2.**
- **PSTN–Compendium Compliance Test Description**, German contribution to NET 4.

References to these specifications appear in bold. For example, **6.2**.

**Warning:** The DirectTalk/2 system has been certified for attachment to the German public switched telephone network (PSTN). It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the German PTT certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*. The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ringing Signal Detection”</td>
<td>“General” on page 334</td>
</tr>
<tr>
<td>“Identification Signal” on page 334</td>
<td>“Dial Tone Reception” on page 335</td>
</tr>
<tr>
<td>“Duration of the Loop Condition” on page 334</td>
<td>“Dialing with Loop Pulsing” on page 335</td>
</tr>
<tr>
<td></td>
<td>“Dialing with DTMF Pulses” on page 336</td>
</tr>
<tr>
<td></td>
<td>“Duration of the Call” on page 337</td>
</tr>
<tr>
<td></td>
<td>“Repetition of a Call” on page 337</td>
</tr>
</tbody>
</table>

**Ringing Signal Detection**

**PTT 6.2**

All DirectTalk/2 applications must start at least 7 seconds after the beginning of the first ringing signal.

To meet this requirement, set the Number of rings parameter of the Wait_for_Call action to a value greater than or equal to 2.
Identification Signal

PTT 6.4

DirectTalk/2 applications must generate an announcement to indicate to the caller that the call has been answered within 2.5 seconds of answering the call.

To do this, use the Play_Module action immediately after receiving the Phone answered return code from the Wait_for_call action.

Duration of the Loop Condition

PTT 6.5, 6.5.1

When recording voice, DirectTalk/2 is limited by the size of the disk. DirectTalk/2 plays a message when recording space has been exhausted, then disconnects the caller.

DirectTalk/2 must hang up the call if the following signals occur for at least 8 seconds:

- Speech signals with a level less than –52 dBm.
- Signals with a single frequency in the range 375 Hz through 550 Hz, with a level of up to –3 dBm.
- Multi-frequency dialing signals with an overall level less than –52 dBm.
- White noise of up to 20 KHz with a level less than –52 dBm.

When configuring DirectTalk/2 you must configure the Telephony Server as follows:

- For Silence Before Hangup, set the value to 8 seconds.
- For Hangup Pattern, the values should represent the Hangup pattern, if applicable.

General

PTT 5.1, 5.1.3

When making outbound calls, DirectTalk/2 must wait for dial tone to be present before dialing. If dial tone is not present within 10 seconds, DirectTalk/2 must disconnect (hangup). You can alter this setting by changing the Max. Dial Tone Wait Time parameter value. See the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide. Your application should execute a Hangup action immediately after receiving the No ring return code from the Place_a_Call action.

When placing a call and receiving the No ring return code, an application is allowed 6 attempts to make a connection. After 6 unsuccessful attempts, the application must no longer seize the line for dialing until the equipment has been checked by an operator. One way to implement this is to have an application execute a Hang_up_Phone and Return_from_Appl action after a No ring return code. The Session Monitor should be configured with Restarts=6. This will ensure the application is shut down after 6 failed attempts.
Dial Tone Reception

**PTT  5.2, 5.2.2**

The dial tone is a continuous tone in the range 390 through 490 MHz, with a power of $-4$ to $-27$ dBm.

The dial tone evaluation time must be between 2 and 5 seconds. Once the valid dial tone has been detected, dialing must start within 200 through 2000 ms.

You must define the dial tone as one of DTONE-L, DTONE-I, or DTONE-X using the Tone Detection function. Additionally, the Place_a_Call action must use the corresponding character (L, I, or X) at the appropriate place in the dialed string, usually as the first character. Your DirectTalk/2 system has been configured to meet these requirements.

You may change the amount of time you wait for the dial tone and the amount of time for valid dial tone using the Max Dial Tone Wait and Valid Dial Tone parameters. If no valid dial tone is detected within the specified time limit, the Place_a_Call action returns the No dial tone return code. In response to this, your application must immediately execute the Hang_up_Phone action. If a valid dial tone is detected, DirectTalk/2 will begin dialing.

Dialing with Loop Pulsing

**PTT  5.3, 5.3.2, 5.3.3**

The interdigit pulse delay must be between 760 ms and 920 ms. DirectTalk/2 has been configured to meet these requirements. To perform pulse dialing, set the Initial Tone Type parameter to Pulse. Additionally, DirectTalk/2 can be set to perform pulse dialing by including the character ‘P’ in the dialed string in the Put_Tone_String or Place_a_Call actions. If you include the character ‘T’ in the dialed string then DirectTalk/2 will perform DTMF dialing.

When dialing behind a PBX, use the Put_tone_String action to dial the Central Exchange access number. Define the PBX and Centrex dial tones as DTONE-L and DTONE-I, using the Tone Detection feature. When using the Place_a_Call action, use the “L” and “I” characters to wait for the appropriate dial tones. It is recommended that the intended number is preceded by at least one pause character (usually “,”) to ensure that the PBX is prepared to dial the number.

When connected to the Central Exchange, you must dial the digits within 3 seconds of detecting the dial tone. To meet this requirement, your voice program should not have more than 3 seconds of pause before the first digit of the phone number specified in the Place_a_Call action.

Appendix G. Programming Considerations for Germany  335
Dialing with DTMF Pulses

PTT 5.4

The duration of DTMF tones must be between 80 and 90 ms.

The interdigit DTMF delay must be between 80 ms and 90 ms.

DirectTalk/2 has been configured to meet these requirements.

If no answering signal (or message) is received within 60 seconds after the last digit is dialed, the line must be released.

DirectTalk/2 supports this using the Place_a_Call action and its return codes. Your application must ensure that this requirement is met.

If the equipment is designed to wait for the end of ring back and then play a welcome message, such a welcome message can be played for up to 180 seconds. If no interaction with the called party has been detected after 180 seconds, the line must be released.

When writing DirectTalk/2 applications, you must ensure that your welcome message is no longer than 180 seconds. Also, DirectTalk/2 voice logic modules must be configured so that the total duration of repeating a prompt does not exceed 180 seconds. The current default for waiting for input is 10 seconds.

If the equipment is waiting for an interaction with the called party without playing any message, the limit of 60 seconds from the end of dialing applies, even if the ring back and the end of ring back have been detected.

To meet this requirement, you must either set the Silence Before Hangup parameter to 60 or your application must start a timer using the Start_Clock action before the Place_a_Call action. If, after the Answer return code is received, the timer reaches 60 seconds, the application must disconnect using the Hangup action.

When dialing behind a PBX, use the Put_tone_String action to dial the Central Exchange access number. Define the PBX and Centrex dial tones as DTONE-L and DTONE-I, using the Tone Detection feature. When using the Place_a_Call action, use the “L” and “I” characters to wait for the appropriate dial tones. It is recommended that the intended number is preceded by at least one pause character (usually “,”) to ensure that the PBX is prepared to dial the number.

When connected to the Central Exchange, you must dial the digits within 3 seconds of detecting the dial tone. To meet this requirement, your voice program should not have more than 3 seconds of pause before the first digit of the phone number specified in the Place_a_Call action.
Duration of the Call

PTT 5.6

The duration of the call is limited to 1 minute. It can be extended by 1 minute by manual means. That is, an interaction with the called party. For example, DTMF input or voice recognition input.

You must ensure that any application meets these call duration limits.

Repetition of a Call

PTT 5.7

If a dialing device can initiate call attempts independently, it must include a device for recognizing the busy tone. DirectTalk/2 is configured to detect busy cadences. However, DirectTalk/2 may also be configured to detect busy tones based upon cadence and frequency, using the Tone Detection feature.

If this dialing device recognizes that the called subscriber line is busy, a period of at least 1 minute must pass before the dialing information is retransmitted. This pause must be indicated by optical means, and the terminal equipment must release the line during this period.

If the Place_a_Call action returns a Busy return code, the application must disconnect the line, log a message, and wait for at least 1 minute before it redials the same number.

If the device can detect speech, and recognizes a free subscriber who does not answer within 60 seconds, the device can make up to 12 further call attempts.

If the Place_a_Call action returns a No Answer return code, the application can make up to 12 further call attempts.

Automatic Initiation of Outbound Calls

Unless the system is specifically certified against the requirements outlined in section 5.7, automatic initiation of outbound calls and automatic repetition of call attempts must not be performed. This means that, if your application places calls automatically (without an operator of any kind), as defined in section 5.7, the entire DirectTalk/2 system and application must be specifically certified by the PTT.

Outbound Calls

Outbound calls are covered by the generic product approval providing that:

- The call is initiated by a local or remote operator. For example, somebody calling the system and controlling it through a DTMF keypad (this is the case for a call transfer in an automated attendant or a message forwarding in a voice mail system). The call is then considered the result of an action by the operator.
- There must be a fixed relationship between the line on which the outgoing call is placed and the line from which the call is considered initiated. Line hunting is not allowed for products not considered a PABX. Possible schemes are:
- All outgoing calls are placed on the same line.
- Outgoing calls are placed on the initiating line. For example, call transfer.
- Calls initiated from line A are always placed on line B.
- There is no automatic redialing initiated by the system.
Appendix H. Programming Considerations for Italy

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Italy.

The following requirements are based on the Italian PTT specification series T11-02 through T11-04. References to this specification appear in bold. For example, 3.1.

**Warning:** The DirectTalk/2 system has been certified for attachment to the Italian public switched telephone network (PSTN). It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the Italian PTT certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*.

The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Automatic Answering”</td>
<td>“Disconnect Detection and Call Duration” on page 340</td>
</tr>
<tr>
<td>“Recording Medium Exhaustion” on page 340</td>
<td>“Dialing” on page 340</td>
</tr>
<tr>
<td>“Disconnect Detection and Call Duration” on page 340</td>
<td>“Pulse Dialing (Mechanical and Electronic)” on page 341</td>
</tr>
<tr>
<td></td>
<td>“Call Repetition” on page 341</td>
</tr>
</tbody>
</table>

**Automatic Answering**

**PTT 6.4, 6.4.1, 6.4.2**

When your DirectTalk/2 system answers a call automatically, the system must generate an announcement or tone to indicate to the caller that the call has been answered. This identification signal must be generated within 3 seconds of answering the call. Your application must play a Play_Module, Put_Tones, or Put_Tone_String action immediately after receiving the Phone answered return code from the Wait_for_Call action.
Recording Medium Exhaustion

**PTT 3.3.02**

If an application is to record messages, it must inform the caller or user that the recording medium is exhausted. Once the medium is exhausted, the application must no longer seize the line for incoming calls.

Your DirectTalk/2 system has been configured to meet this requirement.

Additionally, if your application is to record messages, the `RESTART=` parameter in the session monitor configuration must be zero. This will prevent an application from restarting and seizing a line when the recording medium has been exhausted.

Disconnect Detection and Call Duration

**PTT 5.6.1, 5.6.2, 6.5.1, 6.5.2**

Your DirectTalk/2 system must be configured to detect dial tone and continuous silence as a means of determining that the caller has hung up. Further, if no input is received from the called party for a period of 20 seconds, the line must be disconnected. While this is a requirement for only the first 1.5 minutes after connection, it is good practice at all times.

To meet this requirement, ensure that all your voice logic modules have a number of repeats and timeout values so that, when the two are multiplied, they are less than or equal to 20.

You should also set the Silence Before Hangup parameter to 20 seconds or less during DirectTalk/2 configuration. It is recommended that this value is greater than the Voice Logic Module timeout value specified in your applications.

For information about configuring your DirectTalk/2 system, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

Dialing

**PTT 5.1.2, 5.1.3, 5.2.2, 5.8**

The DirectTalk/2 system must not start dialing out until the dial tone has been received from the telephone exchange and has been recognized as such. Dialing by means of preset waiting times is not permitted. After the connection is made, a signal or message must be sent via the line within 5 seconds. The signal or message must be repeated until the called party has responded.

You must define the dial tone as one of DTONE-L, DTONE-I, or DTONE-X using the Tone Detection function. Additionally, the Place_a_Call action must use the corresponding character (L, I, or X) at the appropriate place in the dialed string, usually as the first character. Your DirectTalk/2 system has been configured to meet this minimum requirement.

You may change the amount of time you wait for the dial tone and the amount of time for valid dial tone using the Max Dial Tone Wait and Valid Dial Tone parameters. If no valid dial tone is detected within the specified time limit, the
Place_a_Call action returns the No dial tone return code. In response to this, your application must immediately execute the Hang_up_Phone action. If a valid dial tone is detected, DirectTalk/2 will begin dialing. If the call is answered, your application must immediately execute a Play_Module action to play a greeting message and the T1 return code should have a value of −1.

For more information, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*.

### Pulse Dialing (Mechanical and Electronic)

**PTT 5.3.2**

When dialing with loop pulsing, the number of pulses corresponds to the digit value except that 10 pulses corresponds to zero.

Your DirectTalk/2 system has been configured to meet these requirements.

To perform pulse dialing, set the Initial Tone Type parameter to Pulse. Additionally, DirectTalk/2 can be set to perform pulse dialing by including a ‘P’ in the dialed string in the Put_Tone_String or Place_a_Call actions. The character ‘T’ in the dialed string will set DirectTalk/2 to perform DTMF dialing.

### Call Repetition

**PTT 5.7**

If a connection cannot be made, automatic redial is permitted, subject to the following conditions:

- The waiting time between the first and second dialing attempt must be greater than 5 seconds. Between subsequent dialing attempts, the waiting time must be greater than 1 minute.
- The same number can be dialed at most 4 times within a period of one hour. Alarm and telemetry calls can be dialed 15 times within one hour.
Appendix I. Programming Considerations for the Netherlands

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in the Netherlands.

The following requirements are based on the Dutch PTT specification series T11-02 through T11-04. References to this specification appear in bold. For example, 3.1.

Warning: It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all applications comply with the Dutch PTT certification. IBM has merely assisted with the interpretation of these requirements in order to provide guidance.

For information about how to change the telephony parameters, refer to section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide.

The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ringing Detection”</td>
<td>“Dialing” on page 345</td>
</tr>
<tr>
<td>“Clearing” on page 344</td>
<td>“Register Recall (R-R) Signal” on page 346</td>
</tr>
<tr>
<td>“Number of Rings” on page 344</td>
<td>“Pulse Dialing (Mechanical and Electronic)” on page 346</td>
</tr>
<tr>
<td>“Automatic Answering” on page 345</td>
<td>“Pulse Dialing (Electronic)” on page 347</td>
</tr>
<tr>
<td>“Disconnect Detection” on page 345</td>
<td>“Multifrequency Dialing (DTMF)” on page 347</td>
</tr>
</tbody>
</table>

Ringing Detection

PTT 3, 3.1, and 3.2.1

The ringing signal consists of signals applied to the telephone line with one of two sequences:

- Sequence 1
  Signal for 1 second, pause for 4 seconds.
- Sequence 2
  Signal for 0.4 seconds, pause for 0.2 seconds, signal for 0.4 seconds, pause for 4 seconds.
DirectTalk/2 can be configured to detect either sequence. Table 37 shows which parameter should be set to what setting in order to achieve each sequence:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Inter-ring delay</th>
<th>Minimum Ring Off</th>
<th>Minimum Ring On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence 1</td>
<td>80 (default)</td>
<td>35 (default)</td>
<td>5 (default)</td>
</tr>
<tr>
<td>Sequence 2</td>
<td>80</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Notes:**
1. Partial ringing signal cycles of 200 ms must be registered.
2. Ringing signals of less than 100 ms will not be registered.

For information about how to change parameters, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*.

**Clearing**

**PTT 2, 3.8**

Clearing is understood to be the transition from the hold state to the idle state or, in other words, the interruption of the direct current loop connection for longer than 1 second. If the external loop is interrupted for less than 150 ms, the connection must be maintained.

DirectTalk/2 has been configured to detect that the caller hung up, based on a drop in current of 1 second. You can make this smaller and still meet the PTT requirement. If you need to reduce the length of the drop in current that signifies that the caller has hung up, change the Minimum Look Current Off parameter. You must not specify a value lower than 2 for this parameter.

For further information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

**Number of Rings**

**PTT 3.2**

It is recommended that automatic equipment answer calls within a certain number of rings (no more than eight). The Wait_for_Call action allows you to specify the number of rings your application will wait before answering. In your applications, this value should be less than eight (the default is one).
Automatic Answering

PTT 3.3

When your DirectTalk/2 system answers a call automatically, the system must generate an announcement or tone to indicate to the caller that the call has been answered. To do this, use a Play_Module action to play a greeting or a prerecorded tone immediately after answering the phone with the Wait_for_Call action.

Disconnect Detection

PTT 3.4, 3.7.3, 3.9

Your DirectTalk/2 system must be configured to detect dial tone and continuous silence as a means of determining that the caller has hung up. The PTT requirements state that a dial tone must be detected within 2 seconds after it begins. To do this, define the dial tone characteristics as a HANGUP tone using the Tone Detection function. If the defined HANGUP tone is detected, the DirectTalk/2 actions will return a HUP return code. If you do not set this parameter, you must design your application to remain online for no longer than 120 seconds.

If your application is recording the caller’s voice, you must configure your system to detect a disconnection after 5 seconds of continuous silence. To do this, set the Silence Before Hangup parameter to 5 seconds during DirectTalk/2 configuration.

For information about configuring your DirectTalk/2 system, refer to the IBM CallPath DirectTalk/2 Installation Guide.

Dialing

PTT 2, 10.2, and 10.2.1

Following the establishment of the DC connection, the DirectTalk/2 system must not start dialing out until the dial tone has been received from the telephone exchange and has been recognized as such. It must wait for at least 5 seconds and at most 40 seconds for the dial tone. Dialing by means of preset waiting times is not permitted. If no dial tone is detected within this period after the moment of seizure, the system must clear the connection (hang up). After the connection is made, a signal or message must be sent via the line.

You must define the dial tone as one of DTONE-L, DTONE-I, or DTONE-X using the Tone Detection function. Additionally, the Place_a_Call action must use the corresponding character (L, I, or X) at the appropriate place in the dialed string, usually as the first character. Your DirectTalk/2 system has been configured to meet this minimum requirement.

You may change the amount of time you wait for the dial tone and the amount of time for valid dial tone using the Max Dial Tone Wait and Valid Dial Tone parameters. If no valid dial tone is detected within the specified time limit, the Place_a_Call action returns the No dial tone return code. In response to this, your application must immediately execute the Hang_up_Phone action. If a valid dial
tone is detected, DirectTalk/2 will begin dialing. If the call is answered, your application must immediately run a Play_Module action to play a greeting message.

For more information, refer to the IBM CallPath DirectTalk/2 Installation Guide.

When dialing a trunk, international, or 06 number, DirectTalk/2 must wait, after transmitting the trunk, international, or 06, for the next dial tone before continuing with the dialing procedure. To satisfy this requirement, you must define the additional international dial tone using the Tone Detection function. The dialed string for the Place_a_Call action must then contain the appropriate dial tone character in the appropriate place. For example, the dialed string:

```
L/zerodot6I123456
```

would wait for the dial tone as defined by DTONE-L, dial the numbers 06, wait for the dial tone defined by DTONE-I, and finally dial the numbers 123456.

**Register Recall (R-R) Signal**

**PTT 6**

The purpose of the R-R signal is to enable subscribers to use certain facilities offered by the telephone network during a call, such as the enquiry call facility. If the equipment is fitted with a register recall facility, the requirements relating to register recall must be met.

The R-R signal takes the form of an interruption of the DC connection (loop interruption) of at least 90 ms and, at most, 130 ms. During this interruption, the line current must not exceed 0.5 ma.

DirectTalk/2 is configured to meet the minimum requirement for the R-R signal (90 ms). If you want to change this, modify the Flash Time parameter. For further information, refer to the IBM CallPath DirectTalk/2 Installation Guide.

**Pulse Dialing (Mechanical and Electronic)**

**PTT 7.2 and 7.4**

The frequency at which the pulses are transmitted must be between 9 and 11 Hz. The duration of a pulse is 61.5% through 64.5% of the period (where period = pulse duration + interval).

The digit interval between two successive pulse series must be at least 700 ms.

Your DirectTalk/2 system has been configured to meet these requirements.
Pulse Dialing (Electronic)

PTT 8.2, 8.3, and 8.4

The digit interval between two successive pulse series must be no more than 3 seconds. Your DirectTalk/2 system has been configured to meet this requirement. If you need to change this interval, modify the Pulse Interdigit Delay parameter. For more information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

If an incidental line interruption current occurs with a duration of more than 1 second, dialing must be stopped.

An accidental line current interruption of at most 150 ms between the pulse series, caused by external conditions, must not affect the dialing function.

Your DirectTalk/2 system has been configured to handle these loop drop conditions as stated in the requirements.

Multifrequency Dialing (DTMF)

PTT 9.14

The sending time and (pause) interval of the dialing signals must each be longer than 65 ms. The sending time is the time that the dialing signal is present without interruption. In the case of automatic multifrequency dialing, the digit period (digit + pause) must not exceed 250 ms.

Your DirectTalk/2 system has been configured to meet the minimum requirement. If you need to change this, update the Pulse Interdigit Delay parameter. For more information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

Automatic Dialing

PTT 10.2.2, 10.2.3, and 10.2.4

If a busy tone or congestion indication is received, the line must be cleared (go on-hook) within 20 seconds. You should define the busy and congestion tones as types BUSY using the Tone Detection function. This means that after *Place_a_Call* returns the Busy return code, your application must immediately run the *Hang_up_Phone* action.

If a connection cannot be made, automatic redial is permitted, subject to the following conditions:

- The waiting time between the first and second dialing attempt must be greater than 5 seconds. Between subsequent dialing attempts, the waiting time must be greater than 1 minute.
- The same number can be dialed at most 15 times within a period of one hour.
Appendix J. Programming Considerations for New Zealand

This appendix discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in New Zealand.

DirectTalk/2 holds Telepermit number PTC 212/91/017 for connection to the New Zealand telephone network. The following requirements are based on Telecom Corporation of New Zealand specification PTC 100.

**Warning:** It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the Telepermit certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the *inst.*

The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Recording Time Expiration&quot;</td>
<td>&quot;Identification Message&quot;</td>
</tr>
<tr>
<td>&quot;Decadic Dialling&quot; on page 350</td>
<td>&quot;Call Logging&quot; on page 350</td>
</tr>
</tbody>
</table>

**Recording Time Expiration**

If an application is to record messages, it must inform the caller when the time for recording a message has expired. Your DirectTalk/2 system has been configured to meet this requirement automatically when using the optional Voice Messaging action Take_a_Message. However to meet this requirement for any Record_Voice action, the application must play a warning message if recording time is exceeded. That is, a suitable Play_Module action must follow a Record_Voice action for the 'Maximum time' return code (RC=1).

**Identification Message**

When an outgoing call is answered, the application must play an identification message (verbal announcement). To meet this requirement, a suitable Play_Module action must follow a Place_a_Call action for the 'Answered' return code (RC=0).
Decadic Dialling

Outward tone dialling is permitted for DirectTalk/2 in New Zealand, but not decadic (pulse) dialling. Therefore you must not include a ‘P’ character in digit strings for the Place_a_Call or Put_Tone_String actions.

Call Logging

The call log incorporated in this equipment cannot account for incorrect information or marginal timing errors which may occur during normal call processing; therefore it may not register all answered calls. The call log therefore may not agree with the Telecom account, which may include calls not shown on the log.

Physical Requirements

The physical requirements are as follows:

- The supplied label is to be attached to the exterior of the PS/2 in a visible position. If no label is attached please contact your IBM representative for supply. The text of the label reads as follows:

<table>
<thead>
<tr>
<th>Label Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>• TELEPERMIT — This DirectTalk/2 voice processing system is permitted by Telecom Corporation of New Zealand Limited for connection to its network.</td>
</tr>
<tr>
<td>• PTC 212/91/017</td>
</tr>
<tr>
<td>• This Telepermit is issued to IBM NZ Ltd, subject to the conditions of Specification PTC 100 and any further conditions stated in letter CPAS 212/91/017 dated 29th August 1991.</td>
</tr>
<tr>
<td>• This plug-in card shall only be used with the type of equipment specified in the user instructions.</td>
</tr>
<tr>
<td>• Ringer Approximate Loading (RAL) = 0.7</td>
</tr>
</tbody>
</table>

The plug-in card referenced above is the IBM-supplied telephony network interface card(s). These may be plugged into system units.

- Immediately disconnect the equipment should it become physically damaged, and arrange for its disposal or repair.
- Disconnect the Telecom connection before disconnecting the power connection prior to relocating the equipment, and reconnect power first.
- The PC which houses this card is classified as a computer rather than a communication terminal. As with all mains powered electrical equipment, there is a legal requirement for it to meet the requirements of the New Zealand Wiring Regulations. It is the responsibility of the PC supplier rather than the Telepermit System to ensure that these requirements are met.
- For reasons of electrical safety, the PC which houses this card must be earthed. The operation of this card in a PC which is not earthed may negate the users rights under the Telecom terms of service.
Appendix K. Programming Considerations for Norway

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Norway.

The following requirements are based on the Norwegian PTT specifications:

- *Introduction to the Norwegian National Part of the Initial NET-4.*
- *Type Approval Regulation of Subscriber Terminal Equipment—Group of Analog Subscriber Interface to the Public Switched Telephone Network.*

References to these specifications appear in bold. For example, 6.4.

**Warning:** The DirectTalk/2 system has been certified for attachment to the Norwegian public switched telephone network (PSTN). It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the Norwegian PTT certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the *IBM CallPath DirectTalk/2 Installation Guide*. The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls.

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Identification Signal&quot;</td>
<td>&quot;Dial Tone Detection&quot; on page 352</td>
</tr>
<tr>
<td>&quot;Duration of the Loop Condition&quot; on page 352</td>
<td>&quot;Dialing with Loop Pulsing&quot; on page 353</td>
</tr>
<tr>
<td></td>
<td>&quot;Dialing with DTMF Pulses&quot; on page 353</td>
</tr>
<tr>
<td></td>
<td>&quot;Duration of the Call&quot; on page 353</td>
</tr>
<tr>
<td></td>
<td>&quot;Repetition of a Call&quot; on page 353</td>
</tr>
<tr>
<td></td>
<td>&quot;Identification Signal&quot; on page 354</td>
</tr>
</tbody>
</table>

**Identification Signal**

6.4

DirectTalk/2 applications must generate an announcement to indicate to the caller that the call has been answered. To do this, use the Play_Module action within 3 seconds of answering the call with the Wait_for_call action. The Wait_for_call action should answer the call after 1 ring by setting the Number of rings parameter to 1. The Phone answered return code should go immediately to a Play_Module action.

© Copyright IBM Corp. 1992, 1996
Duration of the Loop Condition

6.5, 6.5.2

If the caller does not respond to a prompt within 20 seconds, the DirectTalk/2 application must hang up the call. Currently, voice logic modules have a default timeout value of 10, and a repeat value of 3. To conform with this requirement, you must change these values. For example, change the repeat value to 2.

DirectTalk/2 must hang up the call within 10 seconds if:

- The speech signal level is less than \(-48\) dBm for more than 20 seconds.
- There is a hangup signal which is a continuous tone or pulses in the frequency range of 340 Hz through 500 Hz.

When you configure DirectTalk/2 you must configure the Telephony Server as follows:

- Set Silence Before Hangup to 10 seconds.
- Use the Tone Detection function to define the hangup signal as a HANGUP tone type.

Dial Tone Detection

5.1, 5.1.3, 5.2, 5.2.2

When making outbound calls, DirectTalk/2 must wait for dial tone to be present. You can alter the amount of time that DirectTalk/2 waits for dial tone to be present by changing the Maximum Dial Tone Wait parameter.

The dial tone is a continuous AC voltage in the frequency range 350 Hz through 500 Hz, and with a level of 0 to -30 dBm.

You must define the dial tone as one of DTONE-L, DTONE-I, or DTONE-X using the Tone Detection function. Additionally, the Place_a_Call action must use the corresponding character (L, I, or X) at the appropriate place in the dialed string, usually as the first character. Dialing must begin no more than 2 seconds after the dial tone is received.

DirectTalk/2 has been configured to meet these requirements.

You may change the amount of time you wait for the dial tone and the amount of time for valid dial tone using the Max Dial Tone Wait and Valid Dial Tone parameters. If no valid dial tone is detected within the specified time limit, the Place_a_Call action returns the No dial tone return code. In response to this, your application must immediately execute the Hang_up_Phone action. If a valid dial tone is detected, DirectTalk/2 will begin dialing.
Dialing with Loop Pulsing

5.3, 5.3.2, 5.3.3

When dialing with loop pulsing, the interdigit pause must be in the range of 800 to 1000 ms.

You can alter this setting by changing the Pulse Interdigit Delay parameter.

DirectTalk/2 has been configured to meet these requirements.

To perform pulse dialing, set the Initial Tone Type to Pulse. Additionally, DirectTalk/2 can be set to perform pulse dialing by including the character ‘P’ in the dialed string in the Put_Tone_String or Place_a_Call actions. The character ‘T’ in the dialed string will set DirectTalk/2 to perform DTMF dialing.

Dialing with DTMF Pulses

5.4

The interdigit pause must be between 65 ms and 130 ms, excluding fall time.

You can alter this setting by changing the DTMF Interdigit delay parameter.

DirectTalk/2 has been configured to meet these requirements.

Duration of the Call

5.6, 5.6.2

If the caller has not responded to a prompt within 20 seconds, the DirectTalk/2 application must hang up the call. Currently, voice logic modules have a default timeout value of 10, with a repeat value of 3. To conform to this requirement, you must change these values. For example, change the timeout value to 2.5 seconds.

Repetition of a Call

5.7

The equipment can repeat the call attempt automatically only after the detection of a busy or congestion tone.

Between each successive call attempt, the DirectTalk/2 application must pause for 5 seconds.

The number of call attempts to one number must be limited to 10 attempts, except from alarm and telemetry equipment, to which a maximum of 15 attempts applies.

Application developers must ensure that their application meets these call repetition limits.
Identification Signal

5.8

Equipment with an automatic calling function must play a prompt within 5 seconds of the end of the last dialed digit. The prompt must be repeated continuously until the called party has responded or disconnected.

Application developers must ensure that their application meets this call identification signal requirement. Due to the unpredictable timing of a call connection, one way to do this is to use the Put_Tone_String action instead of the Place_a_Call action. This allows you to dial a number and immediately play the prompt. However, Put_Tone_String does not perform call analysis.
Appendix L. Programming Considerations for Portugal

This appendix discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Portugal.

The following requirements are based on the Portuguese PTT Technical Specifications for Access to the Public Switched Telephone Network, 25.01.51.001 Issue 1. References to this specification appear in bold. For example, 3.1.

**Warning:** It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all applications comply with the Portuguese PTT certification. IBM has merely assisted with the interpretation of these requirements in order to provide guidance.

For information about how to change the telephony parameters, refer to section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*.

The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ringing Detection”</td>
<td>“Dialing” on page 357</td>
</tr>
<tr>
<td>“Number of Rings” on page 356</td>
<td>“Pulse Dialing” on page 357</td>
</tr>
<tr>
<td>“Automatic Answering” on page 356</td>
<td>“Multifrequency Dialing (DTMF)” on page 358</td>
</tr>
<tr>
<td>“Call Termination” on page 356</td>
<td>“Call Termination” on page 356</td>
</tr>
<tr>
<td>“Maintaining Connection” on page 357</td>
<td>“Maintaining Connection” on page 357</td>
</tr>
<tr>
<td>“Recording Conversation” on page 357</td>
<td>“Recording Conversation” on page 357</td>
</tr>
</tbody>
</table>

**Ringing Detection**

4.1

The ringing signal is applied to the telephone line with the following characteristics:

400 Hz signal for 1 second, pause for 5 seconds.

DirectTalk/2 is configured to detect this sequence.
Number of Rings

5.3.8.1

It is required that automatic equipment answer calls between one and ten rings. The Wait_for_Call action allows you to specify the number of rings your application will wait before answering. In your applications, this value should be ten or less (the default is one).

Automatic Answering

5.3.8.2

When your DirectTalk/2 system answers a call automatically, the system must generate an announcement or tone to indicate to the caller that the call has been answered. Additionally, this must be at least 2.5 seconds in length. To do this, use a Play_Module action to play a greeting or a prerecorded tone immediately after answering the phone with the Wait_for_Call action. The greeting should be at least 2.5 seconds in length.

Call Termination

5.3.5.1, 5.3.6.1, 5.3.8.3, 5.11.2

The DirectTalk/2 system must disconnect if an established call (inbound or outbound) is inactive for 1.5 minutes.

Your application must ensure that, if no response has been received from the caller, or no output has been generated to the caller, within 1.5 minutes that a Hang_up_Phone action is executed.

Your DirectTalk/2 system must be configured to detect hangup tone and continuous silence as a means of determining that the caller has hung up. If an established call (inbound or outbound), meets one of the following conditions for 20 seconds, DirectTalk/2 must disconnect within 10 seconds:

- The level of the information on the lines is less than -48 db
- Dialling, engaged, or inaccessible signals are received

The signals are applied to the telephone line with the following characteristics:

<table>
<thead>
<tr>
<th>Type</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialling</td>
<td>400 Hz signal, continuous</td>
</tr>
<tr>
<td>Special Dialling</td>
<td>425 Hz for 1 second, pause for 200 ms</td>
</tr>
<tr>
<td>Engaged</td>
<td>400 Hz for 500 ms, pause for 500 ms</td>
</tr>
<tr>
<td>Inaccessible</td>
<td>400 Hz for 200 ms, pause for 200 ms</td>
</tr>
</tbody>
</table>

These tones may defined using the General Tone Detection definitions. These definitions should take into account 20 second duration maximum above. You may disconnect before the 20 seconds has expired and still meet the PTT requirements. For information about configuring your DirectTalk/2 system, refer to the IBM CallPath DirectTalk/2 Installation Guide.
Maintaining Connection

5.4.5.2

The DirectTalk/2 system must be able to absorb a loop current drop up to 110 ms without altering its normal operation or disconnecting. Your DirectTalk/2 system has been configured to meet the minimum requirement. If you need to change this, update the Minimum Loop Current Off parameter. For more information, refer to the IBM CallPath DirectTalk/2 Installation Guide.

Recording Conversation

5.3.9

When recording a telephone conversation, the DirectTalk/2 system must generate a 1400 Hz tone, amplitude -20 dBm to -15 dBm, duration 350ms to 500ms every 15s during the conversation. Your DirectTalk/2 system cannot currently meet this requirement. You may not use the DirectTalk/2 system to record complete telephone conversations especially if the recording is unannounced. You may, however, record messages or information for which a caller has been prompted.

Dialing

5.3.1.1, 5.3.1.2, 5.3.2.1, 5.3.2.2

Following the establishment of the DC connection (phone offhook), the DirectTalk/2 system must not start dialing out until the dial tone has been received from the telephone exchange. The DirectTalk/2 system must also disconnect if it does not detect the dial tone within 10 seconds of taking the phone offhook. Additionally, once dial tone is detected, DirectTalk/2 must begin dialling digits within 3 seconds.

Your DirectTalk/2 system has been configured to meet this minimum requirement. The Place_a_Call action will wait for dial tone before sending the first digit of the number being dialed. If no dial tone is detected within the specified time limit, the No dial tone return code is passed back. In response to this, your application must immediately run the Hang_up_Phone action. If the call is answered, your application must immediately run a Play_Module action to play a greeting message.

You can control the length of time that DirectTalk/2 waits before returning an indication that there is no dial tone by changing the Maximum Dialtone Wait parameter. For more information, refer to the IBM CallPath DirectTalk/2 Installation Guide.

Pulse Dialing

5.6.1.2, 5.6.1.3, 5.6.1.7

The frequency at which the pulses are transmitted must be between 9 and 11 Hz. The break period is 63% through 70% of the period (where period = pulse duration + interval) with a nominal value of 66 2/3%.

The digit interval between two successive pulse series must be between 600 and 1000ms.
Your DirectTalk/2 system has been configured to meet this requirement. If you need to change the digit interval, modify the Pulse Interdigit Delay parameter. For more information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

**Multifrequency Dialing (DTMF)**

5.6.2.2, 5.6.2.7, 5.6.2.8

The interval between digits must be between 65 and 150 ms.

Your DirectTalk/2 system has been configured to meet the minimum requirement. If you need to change this, update the DTMF Interdigit Delay parameter. For more information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

Your application must not issue the DTMF A, B, C, D tones.

**Automatic Dialing**

5.3.4.5.1, 5.3.4.5.2

If a connection cannot be made, automatic redial is permitted, subject to the following conditions:

- The waiting time between dialling attempts must be greater than 60 seconds.
- The same number can be dialed at most 5 times within a period of one hour. Alarm and remote control calls have maximum limit of 15 attempts.

If connection with the called party has not been established within these limits, the DirectTalk/2 system may not call them again until manual intervention has occurred to begin a new set of call attempts.

You must ensure that your application meets these requirements.
Appendix M. Programming Considerations for South Africa

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in South Africa.

The following requirements are based on the South African SAPT Specification TA/001/Issue 1, TA/002/Issue 1, TA/003/Issue 1, and TA/004/Issue 1. The SAPT Specification TA/005 through TA/013 and 3C22/001 do not currently apply to DirectTalk/2. References to this specification appear in bold. For example, TA/001/3.1.

Warning: It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all applications comply with the South African PTT certification. IBM has merely assisted with the interpretation of these requirements in order to provide guidance.

For information about how to change the telephony parameters, refer to section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide.

The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Ringing Detection”</td>
<td>“Dialing” on page 362</td>
</tr>
<tr>
<td>“Automatic Answering” on page 360</td>
<td>“Pulse Dialing” on page 362</td>
</tr>
<tr>
<td>“Call Termination” on page 360</td>
<td>“Multifrequency Dialing (DTMF)” on page 362</td>
</tr>
<tr>
<td>“Maintaining Connection” on page 360</td>
<td>“Automatic Dialing” on page 363</td>
</tr>
<tr>
<td>“Recording” on page 361</td>
<td>“Remote Supervisory Functions” on page 363</td>
</tr>
<tr>
<td>“Timed Break (Flash) Signal” on page 361</td>
<td>“Outbound Dialing Exchange Delays” on page 363</td>
</tr>
<tr>
<td>“Repertory Dialling” on page 364</td>
<td></td>
</tr>
</tbody>
</table>

Ringing Detection

4.1

The ringing signal is applied to the telephone line with the following characteristics:

400 Hz signal for 1 second, pause for 5 seconds.

DirectTalk/2 is configured to detect this sequence.
Automatic Answering

TA/001/4.3.4, TA/002/2.2

When your DirectTalk/2 system answers a call automatically, the system must generate an announcement to indicate to the caller that the call has been answered. The announcement must include the telephone number and the fact that a machine has answered the call. To allow for line settling time, any meaningful announcement of tone must not be transmitted for 2 seconds after answering the call (phone offhook).

To do this, use a Pause action with a value of 2 immediately after answering the phone with the Wait_for_Call action.

Call Termination

TA/001.4.4, TA/002/2.7

The DirectTalk/2 system must disconnect if an established call (inbound or outbound) is inactive for 40 seconds.

Your application must ensure that if no response has been received from the caller or no output has been generated to the caller within 40 seconds that a Hang_up_Phone action is executed.

Your DirectTalk/2 system must be configured to detect hangup tone and continuous silence as a means of determining that the caller has hung up. If an established call (inbound or outbound), meets one of the following conditions:

- The level of the information on the lines is less than -45 db
- Dialling, engaged or unobtainable signals are received

These tones may defined using the General Tone Detection definitions. definitions should take into account 20 second duration maximum above. For information about configuring your DirectTalk/2 system, refer to the IBM CallPath DirectTalk/2 Installation Guide.

Under no circumstances is the DirectTalk/2 system allowed to hold a line for more than 3 minutes after one of the above conditions has been met.

Maintaining Connection

TA/001/4.2.4.4

The DirectTalk/2 system must be able to absorb a loop current drop up to 110 ms without altering its normal operation or disconnecting. Your DirectTalk/2 system has been configured to meet the minimum requirement. If you need to change this, update the Minimum Loop Current Off parameter. For more information, refer to the IBM CallPath DirectTalk/2 Installation Guide.
Recording

TA/002/2.3, TA/002/2.4, TA/002/2.5, TA/002/2.6, TA/002/2.8

The DirectTalk/2 system must announce that a message is to be recorded and the caller's message may now follow. That is, a caller must always be informed that they are being recorded and only their messages may be recorded.

You must ensure that your application meets this requirement.

When recording, a caller must never be allowed to think that they are being recorded when, in fact, the recording has stopped for one of the following reasons:

- Fixed recording time expired
- Recording media full
- Recording stopped due to silence period

Your application must ensure that when the Maximum time return code is received, that an announcement is played to the caller to tell them that the recording is finished. Additionally, you DirectTalk/2 system has been configured to handle the case of the recording media being full. If you need to change this, update the Recording Exhaustion Time parameter. For more information, refer to the IBM CallPath DirectTalk/2 Installation Guide. Your application may, alternately, stop answering calls when the recording medium is full.

A caller may not be allowed to record over a previously recorded message until the DirectTalk/2 system has acknowledged the message.

Your application must let the caller know that the message has been stored and then should offer choices for deleting, rerecording or saving the message.

Finally, the recording of a two way conversation is permitted as long as it is recorded by one of the parties directly involved in the conversation. Recording by outside parties is unlawful and a criminal act.

Timed Break (Flash) Signal

TA/001/4.2.5.8

The purpose of the Timed Break signal is to enable subscribers to use certain facilities offered by the telephone network during a call.

The Timed Break signal takes the form of an interruption of the DC connection (loop interruption) of at least 53 ms and, at most, 103 ms.

DirectTalk/2 is configured to meet the minimum requirement for the Timed Break signal and is set at 60 ms. If you want to change this, modify the Flash Time parameter. For further information, refer to the IBM CallPath DirectTalk/2 Installation Guide.
Dialing

**TA/001/4.2.2, TA/001/4.2.3.1**

Following the establishment of the DC connection (phone offhook), the DirectTalk/2 system must not start dialing out until the dial tone has been received from the telephone exchange. Additionally, for DirectTalk/2 systems connected to a PABX, applications must dial the access code and wait for the public exchange dial tone before dialling.

Your DirectTalk/2 system has been configured to meet this minimum requirement. The Place_a_Call action will wait for dial tone before sending the first digit of the number being dialed. If no dial tone is detected within the specified time limit, the No dial tone return code is passed back. In response to this, your application should run the Hang_up_Phone action. If the call is answered, your application should immediately run a Play_Module action to play a greeting message.

For dialling from behind a PABX, define the public exchange dial tone using the Tone Detection definitions and then use the appropriate letter code in the dialled string of the Place_a_Call action.

You can control the length of time that DirectTalk/2 waits before returning an indication that there is no dial tone by changing the Maximum Dialtone Wait parameter. For more information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

Pulse Dialing

**TA/001/4.2.4.1, TA/001/4.2.4.3**

The frequency at which the pulses are transmitted must be between 9 and 11 Hz. The break period is 67% through 72% of the period (where period = pulse duration + interval).

The digit interval between two successive pulse series must be between 850 and 1150ms.

Your DirectTalk/2 system has been configured to meet this requirement. If you need to change the digit interval, modify the Pulse Interdigit Delay parameter. For more information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.

Multifrequency Dialing (DTMF)

**TA/001/4.2.5.6**

The interval between digits must be at least 65ms.

Your DirectTalk/2 system has been configured to meet the minimum requirement. If you need to change this, update the DMTF Interdigit Delay parameter. For more information, refer to the *IBM CallPath DirectTalk/2 Installation Guide*.
Automatic Dialing

TA/001/4.2.6

If a connection cannot be made due to the called party being engaged (busy), automatic redial is permitted, subject to the following conditions:

- The waiting time between dialling attempts must be greater than 60 seconds.
- The same number can be dialed at most 10 times in any one series

You must ensure that your application meets these requirements.

Remote Supervisory Functions

TA/004/2.1, TA/004/2.2, TA/004/2.3, TA/004/2.4

Remote Supervisory Functions entail the use of DirectTalk/2 as an interrogation system, control system, alarm or alerting, etc.

Calls must be made only to recipients who will accept and be able to interpret any messages sent on the line. If a call is not placed to automatic receiving equipment, the DirectTalk/2 system must transmit a message within 2 seconds of the last dialled digit. Additionally, the message must be repeated with not more than 5 seconds silence between messages. If a sequence of consecutive phone number is dialled, there must be at least 10 seconds between the end of one dialling sequence and the start of the next. The maximum call duration is 3 minutes unless the called party or equipment controls the length of the call.

If your DirectTalk/2 system is to be used for Remote Supervisory functions, You must ensure that your application meets these requirements. In addition, in order to achieve the message transmission within 2 seconds of dialling, your application will have to use the Put_Tone_String action without call analysis to place all calls.

Outbound Dialing Exchange Delays

TA/001/4.2.7,

Due to switching delays in certain Telkom exchanges, it is possible that there may be a delay of up to 10 seconds from the time that dialling is complete until ringing begins. Additionally, for DTMF dialling, there may be an additional delay of 1.5 seconds per digit.

Your DirectTalk/2 system has been configured to meet this requirement. If you need to change this, update the parameter. For more information, refer to the IBM CallPath DirectTalk/2 Installation Guide.
Repertory Dialling

The DirectTalk/2 system must meet the following requirements in order to operate as a repertory dialler:

- It must operate in conjunction with a standard Telkom telephone
- It must not hold the line for more than 40 seconds after dialling or more than 50 seconds from the start of dialling
- It must release line holding conditions when the handset is lifted

You must ensure that your application meets these requirements.
Appendix N. Programming Considerations for Spain

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Spain.

The following requirements are based on the Spanish PTT specifications:

- *Especificaciones tecnicas de acceso a la red telephonica commutada.*
- *Initial NET 4–Spanish Contribution.*

References to these specifications appear in bold. For example, **6.2**.

**Warning:** The DirectTalk/2 system has been certified for attachment to the Spanish public switched telephone network (PSTN). It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the Spanish PTT certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide.*

The following list shows which items in this chapter refer to inbound calls:

- “Ringing Signal Detection”
- “Identification Signal” on page 366
- “Duration of the Loop Condition” on page 366

In Spain, DirectTalk/2 is not allowed to make outbound calls.

---

**Ringing Signal Detection**

**PTT 6.2**

The loop condition will be established after application of a ringing signal with an open circuit voltage from 35 through 75 V rms, a frequency from 20 through 30 Hz, and a sequence that is made up of a signal duration from 1 through 1.5 seconds and a pause duration of 3 seconds. To achieve this:

<table>
<thead>
<tr>
<th>This parameter:</th>
<th>Has been set to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-ring Delay</td>
<td>80</td>
</tr>
<tr>
<td>Minimum Ring Off</td>
<td>28</td>
</tr>
<tr>
<td>Minimum Ring On</td>
<td>8</td>
</tr>
</tbody>
</table>

Your DirectTalk/2 system has been configured to meet this requirement.
The loop condition must not be established after application of any of the following signals:

- Signal duration 600 ms or less, pause of 1000 ms or longer
- Continuous ringing signal
- Three series of 10 ‘break’ of 67 ms with nine ‘make’ of 33 ms, and with a ‘make’ of 450 ms between each two series

Your DirectTalk/2 system has been configured to meet these requirements. However, if you make any parameter changes you must avoid setting the Inter-ring Delay, Minimum Ring Off and Minimum Ring On parameters to values that would cause detection of these signals.

Note: There are other requirements, that only concern electrical characteristics of the ring detection circuit and do not impact the application software or the parameter setting.

If you need to change any parameters, refer to the IBM CallPath DirectTalk/2 Installation Guide.

Identification Signal

PTT 6.4

On answering an incoming call, an application must play an identification signal in the form of either a tone, or a verbal announcement. The tone must be 2100 Hz, and must comply with CCITT V.25. A verbal announcement must indicate that the number called has actually answered the call.

To meet this requirement, you must run the Play_Module action immediately after answering the call. The voice segment that you play must contain a message or the recorded tone.

Duration of the Loop Condition

PTT 6.5

Terminal equipment prepared to maintain the loop condition without any control related to the information transferred or received through the telephone line must establish the quiescent condition within a period no longer than 1.5 minutes from the change to loop condition or from the change to automatic control.

Terminal equipment prepared to maintain the loop condition with a control related to the information transferred or received through the telephone line must establish the quiescent condition within a period no longer than 1.5 minutes from when the last information was transferred or received.

Terminal equipment prepared to maintain the loop condition with a control related to the information transferred or received through the telephone line must maintain the loop condition (within the limits described above) with a signal as low as 11 mV (open circuit) or $N_{43}$ dBm (on 600 Ohms):

- For any frequency from 300 through 3400 Hz, or
- For frequencies specified in the user’s manual
Terminal equipment in the loop condition must change to the quiescent state within a period no longer than 10 seconds after one of the following events has occurred:

- The limit of 1.5 minutes in previous requirements has expired.
- A signal lower than 6.17 mV (open circuit) or –48 dBm (across 600 Ohms) in the frequency range 300 through 3400 Hz has been applied for a continuous period of 20 seconds.
- A single frequency, continuous or intermittent, or a series of single frequencies, with or without pauses, in the frequency range 320 through 480 Hz, with a voltage greater than 49 mV (open circuit) or –30 dBm (across 600 Ohms) has been applied to the line for a period of 20 seconds.

Practically, the above requirement means that the system must release the line on reception of one of the following signal patterns for 20 seconds:

- A continuous tone. Set the Nonsilence Before Hangup parameter to 20.
- A tone duration from 200 through 235 ms, combined with a pause duration from 150 through 600 ms. If this pattern is found in your environment, define hangup pattern as follows:

<table>
<thead>
<tr>
<th>Set this parameter:</th>
<th>To this setting:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hangup Minimum Silence</td>
<td>15</td>
</tr>
<tr>
<td>Hangup Maximum Silence</td>
<td>60</td>
</tr>
<tr>
<td>Hangup Minimum Nonsilence</td>
<td>20</td>
</tr>
<tr>
<td>Hangup Maximum Nonsilence</td>
<td>24</td>
</tr>
<tr>
<td>Hangup Repeat Count</td>
<td>4</td>
</tr>
</tbody>
</table>

- A tone duration of 1.5 seconds, combined with a pause duration of 3 seconds. If this pattern is commonly found in your environment, define the hangup pattern as follows:

<table>
<thead>
<tr>
<th>Set this parameter:</th>
<th>To this setting:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hangup Minimum Silence</td>
<td>290</td>
</tr>
<tr>
<td>Hangup Maximum Silence</td>
<td>310</td>
</tr>
<tr>
<td>Hangup Minimum Nonsilence</td>
<td>140</td>
</tr>
<tr>
<td>Hangup Maximum Nonsilence</td>
<td>160</td>
</tr>
<tr>
<td>Hangup Repeat Count</td>
<td>4</td>
</tr>
</tbody>
</table>

Alternatively, you may use the Tone Detection function to define any of the above hangup patterns if the frequency components are known. The tones can be defined as one of more HANGUP tone types. This method is more reliable, but you must know the specific frequencies of the tones.
Appendix O. Programming Considerations for Sweden

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Sweden.

The following requirements are based on the Swedish PTT specification 8211 A 112-1989 12 05. References to this specification appear in bold. For example, PTT 9.3.

**Warning:** It is the responsibility of either the application developer, the systems administrator, or both to ensure that all applications adhere to the Swedish PTT certification. IBM has merely assisted with the interpretation of these requirements in order to provide guidance.

**Note:** All these requirements refer to inbound calls. In Sweden, DirectTalk/2 cannot make outbound calls.

For information about how to change telephony parameters, refer to the section on advanced telephony configuration in the *IBM CallPath DirectTalk/2 Installation Guide*.

### Answer Function

**PTT 9.3**

The answering device (voice processing system) shall, in its normal state, answer no later than the second ringing signal. However, in other states, the answering device can answer no later than the fourth ringing signal in the periodic cycle.

You must specify a value of 1 or 2 for the Number of rings parameter in the *Wait_for_Call* action.

### Ringing Signal

**PTT 15.2.2**

The ringing signal consists of a first ringing signal that lasts about 300 ms plus a periodic signalling cycle. The character of this signalling cycle may comply with one of the following alternatives:

- 1-second signal and 5 seconds pause (normal case)
- 1-second signal and 9 seconds pause (exceptional case)
- 1-second double signal consisting of two 330 ms signals separated by a 330 ms pause with a 5-second pause between double signals.

The time that elapses from the first ringing signal to the next ringing signal varies randomly within the specified pause interval. The first ringing signal that is received can vary from 0.3 seconds through 1.3 seconds in length.

The tolerances for the times specified are in the region of 10%.
For subscriber carrier systems, the duration of the ringing signal can vary by ±15% from that of the incoming ringing signal from the host exchange or PBX.

DirectTalk/2 has been configured to detect ringing signals that meet the specifications described above. If your DirectTalk/2 system is in an environment with a non-standard ringing signal, you can change the configuration by modifying the values of the parameters listed in Table 42. Refer to the IBM CallPath DirectTalk/2 Installation Guide for information about how to do this.

Table 42. Ringing Signal Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-ring Delay</td>
<td>12</td>
</tr>
<tr>
<td>Minimum Ring Off</td>
<td>60</td>
</tr>
<tr>
<td>Minimum Ring On</td>
<td>8</td>
</tr>
</tbody>
</table>

Voice-activated Circuits

PTT 6.3.2

The following requirements are imposed for voice-activated circuits:

- The circuit shall be kept activated by speech signals having levels above –50dBm.
- When the voice-activated circuit has not been activated for a period of time no longer than 20 seconds, the equipment shall revert to the idle condition within a subsequent time period of 10 seconds.

The application developer must not allow the line to be silent for more than 20 seconds, that is, the application must play or record voice periodically. If 20 seconds of silence is detected, the application must hang up within the next 10 seconds. When this silence is detected, the actions pass back the Caller hung up return code. The application must then perform the Hang_up_Phone action within 10 seconds. You can configure the length of the silence by changing the Silence Before Hangup parameter. For details about how to change the configuration, refer to IBM CallPath DirectTalk/2 Installation Guide.

The application developer must take care that the application does not go for long periods without playing a recorded message to the caller. One area where this might occur is during interaction with a host application. If the response time is slow, or the voice application needs to navigate through many host screens to retrieve the desired information, you must interrupt this processing at least every 20 seconds and play a prompt to the caller (status, update, or reminder). For host interaction, the key_immediate variable allows DirectTalk/2 to retain control after sending keystrokes to the host.
Dial Tone

PTT 15.3.1

Normally, the dial tone has a frequency of 425 + 15 Hz. Frequencies within a range of 425 + 75 Hz can be encountered. Generally speaking, the level at the subscriber’s equipment lies between -5 dBm (435 mV across 600 ohms) and 25 dBm (43.5 mV across 600 ohms). The character of the dial tone can comply with one of the following alternatives:

- Continuous tone (normal case).
- Busy tone:
  
  250 + 25 ms, 250 + 25 ms
  
  pause

You should define these dial tones, busy tones and hangup tones by using the Tone Detection function. Define each tone as the type which corresponds to its use. DirectTalk/2 will automatically return the appropriate responses and return codes based upon their detection.

In the case where a signal other than the normal dial tone is being used, you will be detecting is a 250 ±25 ms tone followed by 250 ±25 ms of silence, or a 250 ±25 ms tone followed by 700 ms of silence.
Appendix P. Programming Considerations for Switzerland

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in Switzerland.

**Warning:** The DirectTalk/2 system has been certified for attachment to the Swiss public switched telephone network (PSTN). It is the responsibility of the customer (for example, the application developer, the systems administrator, or both) to ensure that all customer applications comply with the Swiss RTT certification. To assist the customer in developing DirectTalk/2 applications, IBM has summarized the following requirements for customer guidance.

---

**Inbound Calls**

Your DirectTalk/2 configuration meets all Swiss PTT requirements for inbound calls.

---

**Outbound Calls**

In Switzerland, DirectTalk/2 cannot make outbound calls.
Appendix Q. Programming Considerations for the United Kingdom

This chapter discusses certain considerations and requirements for DirectTalk/2 application developers and system administrators when installing or operating a DirectTalk/2 system in the United Kingdom.

The following requirements are based on the British Approvals Board for Telephony (BABT) specifications BS-6789 and BABT/SITS/87/31.

Warning: It is the responsibility of either the application developer, the systems administrator, or both to ensure that all applications adhere to BABT certification. IBM has merely assisted with the interpretation of these requirements in order to provide guidance.

For information about how to change the telephony parameters, refer to the section on advanced telephony configuration in the IBM CallPath DirectTalk/2 Installation Guide.

The following table shows which items in this chapter refer to inbound calls and which items refer to outbound calls:

<table>
<thead>
<tr>
<th>Inbound Calls</th>
<th>Outbound Calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Two-way Recording”</td>
<td>“Call Initiation” on page 378</td>
</tr>
<tr>
<td>“Readiness” on page 376</td>
<td>“Call Progress Monitoring” on page 379</td>
</tr>
<tr>
<td>“Exhaustion of Recording Medium” on page 376</td>
<td>“Repeat Attempts” on page 380</td>
</tr>
<tr>
<td>“Paytone” on page 376</td>
<td></td>
</tr>
<tr>
<td>“Auto-answering” on page 377</td>
<td></td>
</tr>
<tr>
<td>“Answertone” on page 377</td>
<td></td>
</tr>
<tr>
<td>“Auto-clearing” on page 377</td>
<td></td>
</tr>
</tbody>
</table>

Two-way Recording

BABT A.2.3.3 and A.2.3.4

In two-way recording, the voices of both the called and the calling party are recorded. In the case where DirectTalk/2 is one of the parties, this means that DirectTalk/2 records its own speech output as well as anything spoken by the caller.

If you intend to perform two-way recording, you must develop a new user action to do so. The DirectTalk/2 system does not provide an action that meets the BABT requirement.

You must also be aware that recording machines, and answering/recording machines with two-way recording facility, must apply a recording warning tone to the line when recording is in progress.
This warning tone is defined as signals of 1400Hz ± 7% of duration 350ms to 500ms every 15 ± 3 seconds at a level of −20dBm ± 3dBm.

**Note:** You must not allow the action to be disabled by the user.

---

### Readiness

**BABT 22.2.1**

Any application that you develop must be at all times ready to record when the caller is invited to do so. That is, any prompt to record must be followed immediately by an action to record the caller's voice.

### Exhaustion of Recording Medium

**BABT A.3.2**

When there are only 30 seconds of message storage time remaining, any application must do one of:

- Not answer subsequent calls
- Change the answering message so that the caller is not invited to leave a message.

DirectTalk/2 will monitor the amount of storage remaining for recorded voice. When the space remaining falls below the designated threshold, any attempt to record will cause DirectTalk/2 to play an error tone and, optionally, a message. The application will then stop. You can alter the text of the message, if you want to. When the amount of available storage is increased, the application can be restarted. Until this happens, the application will not take calls.

DirectTalk/2 has been configured to meet these requirements. To change the amount of recording time which triggers the Exhaustion processing, use the Recording Exhaustion Time parameter.

For more information, refer to the telephony configuration section of the *IBM CallPath DirectTalk/2 Installation Guide*.

To change the message that is played, rerecord the following system voice segment:

**pr_no_space**

Default text is: “This message has been truncated. Recording space has been exhausted.”

If a user wants to develop and use a different action for recording, you must adhere to the BABT requirements.

### Paytone

**BABT 4.2.3.5**

When DirectTalk/2 answers a call, it is possible that a paytone will be heard on the line. DirectTalk/2 does not detect paytone, therefore, immediately after answering the call, your application must play an announcement that will be heard even if
paytone is present. If the paytone persists for a period of time longer than your initial greeting, you may need to replay the greeting.

To ensure that you meet this requirement, be sure to use DirectTalk/2’s built-in timeout and repeat processing on your first input prompt. As a default, DirectTalk/2 is configured to repeat 3 times any prompt that requires input, with a 10-second pause. This will meet the requirement. If you change the repeat value and timeout, make sure that the sum of the length of your greeting and the timeout is greater than the maximum length of the pay tone (13.5 seconds).

Auto-answering

BABT 5.2.3.1.1 and 5.2.3.1.2

Your application must answer all calls within 15 rings. The Wait_for_Call action allows you to specify the number of rings that your application will wait before answering. In your application, this must be less than 15.

Answertone

BABT 5.3.2.1

Most DirectTalk/2 applications are designed for use by naive users. These applications play prerecorded voice messages to the caller to prompt for input and provide information. It is possible to design applications which are intended for use by very experienced users, and do not automatically play a greeting after answering the phone. If you design an application with this characteristic, you must play a tone (known as the answer tone) immediately after answering the call as a confirmation that a connection has been made, and that the system is ready. To do this in DirectTalk/2, record a tone which meets the requirements described below as a voice segment, and play that segment immediately after the Wait_for_Call action in your application.

The answer tone must have a frequency of between 1700 and 2500 Hz, and must persist for a minimum of 2.6 seconds at a level greater than -25 dBm. The answer tone must begin within 2.5 seconds after the call is answered.

Auto-clearing

BABT 6 and C.4

Where there is no conflict with conditions occurring during normal operation of an established call, it is recommended that clearing (application hangs up and cycles back to wait for another call) be initiated under each of the following conditions, irrespective of when it occurs:

- Within 5 seconds of receipt of dial tone (proceed indication and new proceed indication). Dial tone detection is controlled by defining the tone as a HANGUP tone using the Tone Detection function. This is the most reliable method because it includes the frequency component. If the frequency is not known, you may also set the Nonsilence Before Hangup parameter during DirectTalk/2 configuration.
• Within 5 seconds of receipt of ‘equipment engaged’ tone (path engaged indication). The equipment engaged tone can be detected by defining the tone as a HANGUP tone using the Tone Detection function. This is the most reliable method because it includes the frequency component. If the frequency is not known, you may also set the Nonsilence Before Hangup parameter during DirectTalk/2 configuration.

• Within 15 seconds of continuous silence (signals below –52 dBm) appearing on the line. Detection of continuous silence can be enabled by setting the Silence Before Hangup parameter during DirectTalk/2 configuration.

DirectTalk/2 has been configured to meet these requirements. If any of the above conditions is detected, the voice actions will return the Caller hung up return code. Your application must then perform its end-of-call tasks, run the Hang_up_Phone action, and loop back to the Wait_for_Call action.

Call Initiation

BABT 5.2 and BABT 5.3

Proceed indication conditions

The BABT imposes a number of requirements on applications which place outbound calls. If your application detects a proceed indication (generally a dial tone) within 3.5 seconds of taking the phone off hook to dial, then it must send the first dialled digit within 8 seconds of the time the phone is taken offhook. If there is no proceed indication, then your application may either go ahead and dial the number (again, the first digit must be sent within 8 seconds), or hang up within 25 seconds.

You must define the dial tone as one of DTONE-L, DTONE-I, or DTONE-X using the Tone Detection function. Additionally, the Place_a_Call action must use the corresponding character (L, I, or X) at the appropriate place in the dialed string, usually as the first character. DirectTalk/2 supports these requirements using the Place_a_Call action.

You may change the amount of time you wait for the dial tone and the amount of time for valid dial tone using the Max Dial Tone Wait and Valid Dial Tone parameters. If no valid dial tone is detected within the specified time limit, the Place_a_Call action returns the No dial tone return code. In response to this, your application must immediately execute the Hang_up_Phone action. If a valid dial tone is detected, DirectTalk/2 will begin dialing. Your application may then dial the string using the Put_Tone_String action (which does not attempt to detect dial tone), or hang up using the Hang_up_Phone action and then try again using the Place_a_Call action.

In order to meet the 8-second requirement for dialing the first digit, you should set the Maximum Dial Tone Wait parameter to 4 seconds.
Call Progress Monitoring

BABT 6.2, 6.3, 6.4, 6.5, and 7.1

After initiating an outbound call, your application must be able to respond to a number of conditions which may occur if a connection is not made. DirectTalk/2 provides access to detailed call analysis parameters which permit your application (using the Place_a_Call action) to detect these conditions. Table 44 on page 380 contains the BABT specifications for each of the possible conditions that may be encountered when dialing a call. The actual characteristics may differ, based on where your DirectTalk/2 system is located, and whether or not it is attached to a PBX. The default configuration of call analysis parameters provided with your system will meet the BABT specification.

If you determine that you need to make changes to your call analysis parameters, you can change them using advanced telephony configuration as described in the IBM CallPath DirectTalk/2 Installation Guide. Details on how call analysis works and how you can best determine appropriate values for the parameters is beyond the scope of this document. The Dialogic Corporation can provide tools and documentation to assist you with this task.

The BABT has requirements related to how quickly your application must respond (that is, hang up) when one of the conditions in Table 44 on page 380 encountered. These requirements, or modes, are based on the type of call being placed, and are described below:

**Mode 1**
A mode of operation of the application in which the timeout is not more than 1 minute, when each of the ineffective call conditions given in Table 44 on page 380 is received.

Mode 1 can be used for all types of call, except voice alert calls to public emergency authorities and the BT emergency (999) service.

**Mode 2**
A mode of operation of the application in which the timeout is not more than 3 minutes, when each of the ineffective call conditions given in Table 44 on page 380 is received.

Mode 2 is for use only for voice alert calls (other than voice alert calls to the BT emergency 999 service).
Mode 3  A mode of operation of the application in which the timeout is not more than 5 minutes and, for ineffective calls, not less than 4 minutes.

Mode 3 is for use only for voice alert calls to the BT emergency (999) service.

If an application is operating in Mode 1, the application must return to the offline state in 1 minute or less after the last digit of a call is made.

If an application is operating in Mode 2, the application must return to the off-line state in 3 minutes or less after the last digit of a call is made.

If an application is operating in Mode 3 with a fixed timeout, the application must return to the offline state no less than 4 minutes, and no more than 5 minutes, after the last digit of a call is made. For DirectTalk/2 to detect these as disconnect conditions, define each tone as a HANGUP tone using the Tone Detection function.

<table>
<thead>
<tr>
<th>Table 44. BABT Specification for Ineffective Call Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tone</td>
</tr>
<tr>
<td>------------------------------</td>
</tr>
<tr>
<td>Ring tone</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Number unobtainable tone</td>
</tr>
<tr>
<td>Special information tone</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Busy tone</td>
</tr>
<tr>
<td>Equipment engaged tone</td>
</tr>
<tr>
<td>Proceed indication (new</td>
</tr>
<tr>
<td>dial tone)</td>
</tr>
</tbody>
</table>

Repeat Attempts

BABT 7.2.1, 7.2.2, and 7.2.4

When an application repeatedly tries to connect to a number without direct user intervention, the number of repeat call attempts and the minimum duration between them must comply with the call pattern or patterns shown in the table below.

DirectTalk/2 contains no built-in support for any of the call patterns. These patterns must be implemented via application logic.
Notes:

1. The maximum number of calls for repeat attempt call pattern A is 5, and for call pattern B is 7.

2. For patterns C and D, no limit is specified for the value of \( n \).

3. The durations given in the above table are measured from closedown to reseizure.

In any one repeat attempt sequence, an application must not operate in a combination of call patterns A, B, and C.

Where an application intends to automatically repeat either of call patterns A or B, there must be only one sequence of the particular pattern to the same number within a two hour period from when the first call attempt was made.

Call pattern D must only be used if an application can recognize engaged tones. An application must not be able to operate according to pattern D if it cannot recognize and receive engaged tones.

If call pattern D is entered on receipt and recognition of engaged tones via one of the call patterns A, B, or C:

- The 5 second minimum duration for call pattern D must not be used.
- The application must revert to the remainder of the original call pattern of receipt and recognition of tones other than the engaged tone.

<table>
<thead>
<tr>
<th>Call attempt</th>
<th>Pattern A</th>
<th>Pattern B</th>
<th>Pattern C</th>
<th>Pattern D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial, 1st repeat</td>
<td>5 sec</td>
<td>5 sec</td>
<td>5 sec</td>
<td>5 sec</td>
</tr>
<tr>
<td>1st/2nd repeat</td>
<td>1 min</td>
<td>2 min</td>
<td>10 min</td>
<td>3 min</td>
</tr>
<tr>
<td>2nd/3rd repeat</td>
<td>1 min</td>
<td>2 min</td>
<td>10 min</td>
<td>3 min</td>
</tr>
<tr>
<td>3rd/4th repeat</td>
<td>2 min</td>
<td>2 min</td>
<td>10 min</td>
<td>3 min</td>
</tr>
<tr>
<td>4th/5th repeat</td>
<td>End of sequence (note 1)</td>
<td>10 min</td>
<td>3 min</td>
<td></td>
</tr>
<tr>
<td>5th/6th repeat</td>
<td>2 min</td>
<td>End of sequence (note 1)</td>
<td>10 min</td>
<td>3 min</td>
</tr>
<tr>
<td>6th/7th repeat</td>
<td>10 min</td>
<td>3 min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nth (see note 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This glossary defines all important terms and abbreviations used in this book that may be new or unfamiliar to you. If you do not find the term you are looking for, refer to the index or to the IBM Dictionary of Computing, New York: McGraw-Hill, 1994.

This glossary includes terms and abbreviations from:

- The American National Standard Dictionary for Information Systems, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.

- The Information Technology Vocabulary, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among the participating National Bodies of SC1.

### A

**action.** (1) A DirectTalk/2 function that performs an activity in a voice application. DirectTalk/2 provides a set of actions which can be extended by creating your own actions. (2) In SAA Common User Access, one of the defined tasks that an application performs.

**action bar.** In SAA Common User Access, the area at the top of the window that contains choices that give a user access to actions available in that window.

**active.** (1) Operational. (2) Pertaining to a node or device that is connected or is available for connection to another node or device.

**address.** A value that identifies a register, a particular part of storage, a data source, or a data sink. The value is represented by one or more characters. (T)

**ADSI.** Analog Display Service Interface.

**ADSI telephone.** A 'smart' phone, capable of interpreting and returning ADSI data.

**AID.** Attention identifier.

**alphanumeric.** Pertaining to a character set that contains letters, digits, and usually other characters such as punctuation marks. (A)

**Analog Display Service Interface (ADSI).** A Bellcore standard defining a protocol for data transmission over a voice grade telephony channel.

**American National Standard Code for Information Interchange (ASCII).** The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

**ANI.** Automatic number identification

**answer.** To respond to a call to complete the connection between data stations.

**application.** A collection of software components used to perform specific types of work on a computer.

**application control file.** A file containing all of the parameters required to run a voice application in production mode.

**application development.** The design and implementation of a voice processing application.

**Application Manager.** The program that runs a voice application in a production environment after it has been created through the Voice Application Developer.

**application variables.** Variables that are treated as local variables within a particular voice application. See also *local variable*.

**ASCII.** American National Standard Code for Information Interchange.

**attention identifier (AID).** (1) A code in the inbound 3270 data stream that identifies the source or type of data that follows. (2) A character in a data stream indicating that the user has pressed a key, such as the Enter key, that requests an action by the system.
B

blank. A part of a data medium in which no characters are recorded. (A)

break point. A flag set in a voice program which, when reached during debug, causes the application to stop processing.

buffer. A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (A)

Busy (BSY). A condition encountered when a station (or telephone) is offhook or is in the do-not-disturb mode. A busy tone implies a busy condition.

C

call. An attempt to reach a user whether or not the attempt is successful.

caller. The person using a DirectTalk/2 inbound voice application on the telephone.

cancel. (1) To end a task before it is completed. (2) An action that removes the current panel or window without processing it and returns to the previous panel.

case-sensitive. A condition in which entries for a field must conform to a specific lowercase, uppercase, or mixed-case format in order to be valid.

CBX. Computerized branch exchange.

character. A letter, digit, or other symbol that is used as part of the organization, control, or representation of data. A character is often in the form of a spatial arrangement of adjacent or connected strokes.

character string. (1) A sequence of bytes or characters associated with a single-byte character set. (2) A string of characters, such as a command and its parameters, used to communicate with the base operating system.

check. (1) A process for determining accuracy. (A) (2) An error condition. (3) To look for a condition.

clock. In personal computers, a function provided by several software applications and multifunctional boards that keeps track of time and date regardless of whether the computer is on or off.

code. (1) A set of instructions for the computer. (2) To write instructions for the computer. (3) A representation of a condition, such as an error code.

calculate. To examine two items to discover their relative magnitudes, their relative positions in an order or in a sequence, or whether they are identical in given characteristics. (I) (A)

component. (1) A functional part of the DirectTalk/2 voice processing application. (2) Hardware or software that is part of a functional unit.

computer. A functional unit that can perform substantial computations, including numerous arithmetic operations and logic operations without human intervention during a run. In information processing, the term computer usually describes a digital computer. A computer may consist of a stand-alone unit or may consist of several interconnected units. (T)

computerized branch exchange (CBX). An exchange in which a central node acts as a high-speed switch to establish direct connections between pairs of attached nodes.

concatenate. (1) To link together. (2) To join two character strings.

condition (COND). Represents the logical relationship between two values or variables that are specified in a voice logic module or voice program.

configuration. (1) The task of defining the devices, features, parameters, and programs for a system. (2) The arrangement and relationship of the components in a system or network.

confirmation. A prompt in a menu window wherein a program questions the user when the consequences of a user action are significant.

connection. In data communications, an association established between functional units for conveying information. (I) (A)

create. The function which generates a new DirectTalk/2 database file.

cursor. A pointer to an element of a set of results. (T) (2) A movable, visible mark used to indicate the position at which the next operation will occur on a display surface. (A)

D

data. The coded representation of information for use in a computer. Data has certain attributes such as type and length.

database. (1) A collection of data with a given structure for accepting, storing, and providing, on demand, data for multiple users. (T) (2) A collection of
interrelated data organized according to a database schema to serve one or more applications. (T)

**Database Server.** A DirectTalk/2 system server that creates, accesses, and manages DirectTalk/2 databases.

**debug.** (1) To detect and correct errors in a voice application while testing the application. (2) To detect, to locate, and to eliminate errors in computer programs. (T)

**delete.** (1) A function that enables a user to remove all or part of a previously entered text. (T) (2) To remove, for example, to delete a file.

**dial.** To use a dial or pushbutton telephone to initiate a telephone call. In telecommunication, this action is taken to attempt to establish a connection between a terminal and a telecommunication device over a switched line.

**directory.** (1) A table of identifiers and references to the corresponding items of data. (I) (A) (2) A listing of the files stored on a diskette.

**Directory Manager.** The DirectTalk/2 program used to create and maintain the directory of users who have mailboxes that can be accessed using the voice messaging feature.

**Directory Server.** A DirectTalk/2 system server that stores names and numbers for the Mailbox Server and maintains and controls a central directory.

**display.** The function which presents the contents of a DirectTalk/2 database file. The contents are presented in HEX format.

**DNIS.** Dialed number identification service.

**DTMF.** Dual tone multifrequency.

**DTMF digit.** The telephone tone digit that a caller presses.

**dual tone multifrequency (DTMF).** A generic term used to describe an acoustic signal from the key pad of a telephone to the serving switching equipment. Two combining analog tones are used to represent digits (0-9) and characters (#,*).

**E**

**edit.** To add, change, delete, or rearrange data and to perform operations such as code conversion and zero suppression.

**editor.** A computer program designed to perform such functions as rearrangement, modification, and deletion of data in accordance with prescribed rules. (A) DirectTalk/2's Voice Application Developer contains several component editors.

**emulate.** To imitate one system with another, primarily by hardware, so that the imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated system. (A)

**emulator.** A combination of programming techniques and special machine features that permit a computing system to run programs written for a different system.

**error.** A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. (I) (A)

**exit.** An action that ends the active application and removes all windows associated with it. An action that ends a function or application and removes from the screen all windows associated with that function or application.

**extended help.** In SAA Common User Access, a help action that provides information about the contents of the main task window from which a user requested help.

**F**

**fast path.** A way to navigate through a voice-processing system using keys on the telephone keypad.

**FDM.** Feature Download Management

**feature.** A part of an IBM product that may be ordered separately by the customer.

**Feature Download Management (FDM).** An ADSI protocol which enables a number of alternative key and screen overlays to be stored in an ADSI telephone and to be selected by predetermined events at the telephone.

**field.** (1) An area on a window used to contain data. (2) The smallest identifiable part of a record. (3) In SAA, an identifiable area on a screen.

**file.** A collection of related data that is stored and retrieved by an assigned name.

**file name.** (1) The name used by a program to identify a file. (2) The portion of the identifying name that precedes the extension.

**FIFO.** First in, first out.

**first in, first out (FIFO).** The basis on which queue calls are generally handled. New calls are added to the
bottom of the queue, as calls at the top of the queue are answered.

**flag.** (1) A variable indicating that a certain condition holds. (T) (2) A character that signals the occurrence of some condition, such as the end of a word. (A)

**flowchart.** A graphical representation of a process or the step-by-step solution of a problem, using suitably annotated geometric figures connected by flowlines, for the purpose of designing or documenting a process or program.

**force play.** An option in Play_Module that allows a voice logic module to be played to completion without being interrupted by a DTMF tone.

**function.** (1) A specific purpose of an entity, or its characteristic action. (A) (2) A machine action such as a carriage return or a line feed. (A) (3) In ADSI, an ADSI instruction, or group of instructions.

**function key.** A key that causes a specified sequence of operations to be performed when it is pressed.

**function key area.** In SAA Basic Common User Access, the area at the bottom of a panel that identifies function key assignments that are available on that window.

**G**

**global.** (1) Pertaining to information available to more than one program or subroutine. (2) Pertaining to all places in a document or file.

**global variable.** A variable defined in one portion of a computer program and used in at least one other portion of the computer program. (T)

**H**

**hardware.** (1) All or part of the physical components of an information processing system, such as computers or peripheral devices. (T) (A) (2) The equipment, as opposed to the programming, of a system.

**header.** (1) A block of text printed consistently at the top of one or more pages in a multipage document. (2) The portion of a message that contains control information for the message, such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

**help.** (1) Information about the item the cursor is on or about the entire window. (2) An action that gives information about the item the cursor is on, an application panel, or the help facility. (3) An action bar choice that allows a user to select various kinds of help information.

**highlight block.** The block (or blinking line) on the screen that indicates where user input will begin.

**host computer.** (1) In a computer network, a computer that provides end users with servers such as computation and databases and that usually performs network control functions. (2) The primary or controlling computer in a multiple computer installation.

**host interface.** Interface between a network and a host computer. (T)

**host system.** The data processing system to which a network is connected and with which the system can communicate.

**I**

**icon.** A pictorial representation of a choice for the user to select. Icons can represent things (such as a document or file) the user wants to work on. Icons can also represent the actions the user wants to perform.

**ID.** Identification.

**identification (ID).** The process that enables recognition of an entity by a system through personal, equipment, or organizational characters or codes.

**index.** A list of the contents of a file or of a document, together with keys or references for locating the contents.

**input.** (1) The information entered into a computer for processing or storage. (2) Information or data to be processed.

**insert.** The function used to add new steps to a voice program or statements to a voice logic module.

**installation.** The process of placing one or more DirectTalk/2 components on the fixed disk of a workstation.

**interface.** A shared boundary between two or more entities. An interface might be a hardware component to link two devices together or it might be a portion of memory or requesters accessed by two or more computer programs.

**item.** An element of a set of data; for example, a file may consist of a number of items such as records which in turn may consist of other items. (I) (A)
**K**

**key.** A record field used to identify the record and to access the record from within an indexed file.

**key pad.** An alphanumeric telephone key pad used for dialing and entering data to a voice application.

**keys help.** An action in the help panels that gives users a list of the common DirectTalk/2 key assignments.

**L**

**last repeat.** A return code that occurs when a voice logic module has been repeated a specified number of times.

**link.** (1) The physical medium of transmission, the protocol, and associated devices and programming used to communicate between computers. (2) To interconnect items of data or portions of one or more computer programs; for example, the linking of object programs by a linkage editor, linking of data items by pointers. (T)

**literal.** A character string whose value is given by the characters themselves; for example, the numeric literal \(7\) has the value of \('7'\), and the literal \'CHARACTERS'\ has the value CHARACTERS.

**load.** The function which places an ASCII text file into a DirectTalk/2 database file.

**local.** (1) Pertaining to that which is defined and used only in one subdivision of a computer program. (A) (2) Residing on the user's node or workstation.

**local variable.** A variable that is defined and used only in one specified portion of a computer program. (T)

**log.** To record messages on a system data file or storage device.

**logging.** The recording of data about specific events.

**logic.** The part of an application that provides comparisons and decision-making processing.

**M**

**mailbox.** A DirectTalk/2 file that holds the telephone messages of a recipient.

**Mailbox Server.** A DirectTalk/2 system server that creates mailboxes, stores and retrieves messages, and manages user profiles.

**menu.** (1) A displayed list of available machine functions for selection by a user. (2) A displayed list of items from which a user can make a selection.

**message.** The information not requested by users but presented to users by the computer in response to a user action or internal process.

**message log.** A file used to save or log certain types of messages and status information.

**message text.** The part of a message of concern to the party ultimately receiving the message, that is, the message exclusive of the header or control information.

**monitor.** Software or hardware that observes, supervises, controls, or verifies operations of a system. (A) Synonym for visual display unit.

**mouse.** A device that a user moves on a flat surface to position a pointer on the screen. It allows a user to select a choice or function to be performed or to perform operations on the screen, such as dragging or drawing lines from one position to another.

**mouse button.** A mechanism on a mouse that a user presses to select choices or initiate actions.

**N**

**network.** A configuration of data processing devices and software connected for information interchange.

**node.** In a network, a point at which one or more functional units connect channels or data circuits. (I)

**Node Manager.** A DirectTalk/2 menu-driven program used to monitor the status of system resources, including applications and phone lines in a production environment, and to issue commands to alter the status of resources, and to start and stop application sessions and phone lines.

**noise.** A disturbance that affects a signal and that can distort the information carried by the signal. (T)

**null.** A special value that indicates the absence of information.

**O**

**offhook.** The state of a telephone line when in use. When a telephone is answered on a public switched system, it is said to go offhook. Contrast with onhook.

**offline.** (1) Pertaining to the operation of a functional unit that takes place either independently of, or in parallel with, the main operation of a computer. (T) (2)
Neither controlled by, nor communicating with, a computer. Contrast with online.

onhook. The state of a telephone line when not in use. Contrast with offhook.

online. (1) Pertaining to a user’s ability to interact with a computer. A description of a user’s access to a computer by way of a screen. (2) Pertaining to the operation of a functional unit that is under the continuous control of a computer. Contrast with offline.

open. The function that connects a file, adapter, system, resource, or database object to a program for processing.

operand. (1) An entity on which an operation is performed. (2) Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor.

operating system. The software that controls the running of programs. An operating system may provide services such as resource allocation, scheduling, input/output (I/O) control, and data management.

Operating System/2 Extended Edition (OS/2). (1) A family of IBM-licensed program operating systems for personal computers. (2) A program that contains the features of OS/2 Standard Edition Version 1.3. In addition, this program contains an advanced relational Database Manager component, a Communications Manager component, and a LAN Requester component that provide intersystem communications, improved connectivity, terminal emulation, and access to shared network resources.

operator. A symbol that represents an operation to be performed; for example, the plus sign (+).

option. A selectable item on an action bar.


P

parameter. (1) The information supplied by a program or user to a command or function. (2) The data passed between programs or procedures.

password. A unique string of characters known to a computer system and to a user, who must specify the character string to gain access to a system and to the information stored within it. For DirectTalk/2, a utility that is used for communications between a requester and a server.

path. The route used to locate files on a disk or diskette. The route consists of a collection of drives and directories.

pointer. The symbol displayed on the screen that is moved by a pointing device, such as a mouse.

Portmaster*. A card that serves as an input/output adapter for the system unit, and provides communications to an S/370 or S/390 host system.

print. The function which transfers a textual representation of voice segments, voice logic modules, and other components of a voice application to a printer or an ASCII file.

prompt. (1) An action that users request while the cursor is in an entry field. (2) A displayed message that requests input from the user or gives operational information.

PS/2. Personal System/2.

pull-down. An extension of the action bar that displays a list of one or more choices that are available for a selected action bar choice.

Q

query. A request for information from a file based on specific conditions; for example, a request for availability of a seat on a flight reservation system. (T)

R

range. The set of values that a quantity or function may take. (I)

record. (1) A set of data treated as a unit. (T) (2) A set of one or more related data items grouped for processing.

record format. The definition of how data is structured in the records contained in a file. The definition includes record names, field names, and field descriptions such as length and data type.

record key. A field within the first block of each record in an indexed data set that is used in storing and retrieving records in the data set.

record length. The total number of characters represented by the combination of the number of characters for the record key and the number of characters for the record data.

request. A directive, by means of a basic transmission unit, from an access method that causes the network
control program to perform a data-transfer operation or auxiliary operation.

**resource.** Any of the data processing system elements needed to perform required operations, including storage, input/output units, one or more processing units, data, files, and programs. Synonymous with computer resource. (T)

**response.** (1) An answer to an inquiry. (2) In data communication, a reply represented in the control field of a response frame. It advises the primary or combined station of the action taken by the secondary or other combined station to one or more commands. (3) In SNA, a message unit that acknowledges receipt of a request. A response consists of a response header (RH) and possibly a response unit (RU).

**result.** An entity produced by the performance of an operation. (I) (A)

**return.** A dialog control action that saves data, if appropriate, and causes a transition to an application-defined dialog state.

**return code.** A value returned to a program to indicate the results of an operation requested by that program.

**run.** To cause a program, object, utility, or other machine function to be performed.

**S**

**script.** The outline of a DirectTalk/2 application dialogue with the user.

**scroll.** To move all or part of a display image vertically or horizontally so new data is displayed at one edge as preceding data is no longer displayed at the opposite edge.

**SDC.** Server Display Control

**segment.** The whole or partial part of a voice recording that can be played to callers.

**select.** To mark or choose an item in a window, action bar, or menu.

**server.** (1) On a local area network (LAN), a data station that provides facilities to other data stations. (2) An application on the host that processes Server-Requester Programming Interface (SRPI) requests.

**Server Display Control (SDC).** An ADSI control mode in which the ADSI telephone is controlled through a dialog with a Voice Response system.

**session.** (1) A logical connection between two stations or network addressable units (NAUs) that allows them to communicate. (2) The period of time during which a user can communicate with an interactive program.

**Statistics Server.** A DirectTalk/2 system server that creates log records on demand from applications and writes them to a file.

**step.** One operation in a voice program.

**string.** A sequence of elements of the same nature, such as characters, considered as a whole. (T)

**system administrator.** The person at a computer installation who installs, controls, and manages the use of the computer system.

**system variable.** A predefined DirectTalk/2 variable available to all applications. A value that may vary each time the application is run. See also variable and operand.

**target variable.** The variable into which the result of a particular action will be placed.

**task.** A set of one or more sequences of instructions treated by a control program as an element of work to be accomplished.

**TDD.** Telecommunication Devices for the Deaf

**telephony.** Transmission of speech or other sounds.

**Telecommunication Devices for the Deaf (TDD).** Devices, similar to a teletype, that allows hearing-impaired callers to use telephone services. Such devices translate text to TDD data and TDD data back to text, which can be displayed.

**Telephony Server.** A DirectTalk/2 system server that supplies telephony processing services. This processing includes playing and recording voice, tone generation, voice recognition, and text-to-speech.

**template.** On an personal computer, a line entered from the keyboard and stored in memory from which the line can be retrieved, used again, or modified.

**terminal.** (1) A functional unit in a system or communication network at which data may enter or leave. (T) (2) A point in a system of communication network at which data can enter or leave. (A) (3) In data communications, a device, usually equipped with a keyboard and display device, capable of sending and receiving information.
terminal emulation.  The capability of a microcomputer or personal computer to operate as if it were a particular type of terminal linked to a processing unit and to process data.

termination key.  The DTMF key that will be used to indicate that a caller has completed entering a string of DTMF tones.

text.  In text processing, a sequence of elements intended to convey a meaning, whose interpretation is essentially based upon the reader's knowledge of some natural language or artificial language. The elements may consist of characters, symbols, words, phrases, paragraphs, sentences, or tables. (T)

time out.  (1) An event that occurs at the end of a predetermined period of time that began at the occurrence of another specified event. (I) (2) A time interval allotted for certain operations to occur.

U

update.  To add, change, or delete items. To modify a master file with current information according to a specific procedure.

user.  A person who uses a resource on a computer.

utilities.  Those DirectTalk/2 functions that allow users to check their applications for errors, debug those applications, create user-defined actions for use in voice programs, and conduct phone line diagnostics.

V

value.  (1) A specific occurrence of an attribute; for example, "blue" for the attribute "color." (T) (2) A quantity assigned to a constant, a variable, a parameter, or a symbol.

variable.  (1) A quantity that can assume any of a given set of values. (A) (2) An entity that can assume a value.

voice application.  An application that receives or places calls, plays recorded voice segments, and responds to the person's input.

Voice Application Developer.  The DirectTalk/2 component that provides the environment for implementing voice-processing applications. An application developer is guided through menu-driven screens with online help to record and edit the greetings and menus that will be played to the caller, define the logic for the interaction with the caller, and debug the application.

Voice Logic Module Editor.  A DirectTalk/2 component that enables an application developer to create and update voice logic modules.

voice logic module.  A combination of voice logic statements of two types: (1) IF and (2) PLAY.

voice menu.  A part of an application that presents a caller with choices, one of which the caller can choose.

voice pattern.  The graphical representation of a voice segment.

voice program.  A DirectTalk/2 program that contains the steps, actions, voice logic modules, and voice segments that perform a voice application.

Voice Program Editor.  A component of DirectTalk/2's Voice Application Developer used to create and update a voice program.

voice segment.  The words and phrases you record to play to a caller using the voice application. A voice segment can consist of the phrases you record and the system voice variables.

Voice Segment Editor.  A component of DirectTalk/2's Voice Application Developer used to record and play words and phrases.

W

window.  An area of the screen with visible boundaries through which a panel or portion of a panel is displayed.

window title.  A title that identifies the window and associates it with an application.

workstation.  A terminal or personal computer, usually one that is connected to a mainframe or within a network, at which a user can run applications.
Index

Special Characters
<prefix>_dept variable  53
  in Search_Directory action 256
<prefix>_fnm variable  53
  in Search_Directory action 256
<prefix>_grtgkey variable  53
  in Search_Directory action 256
<prefix>_idno variable  53
  in Search_Directory action 256
<prefix>_lnm variable  53
  in Search_Directory action 256
<prefix>_loc variable  53
  in Search_Directory action 256
<prefix>_mailbox variable  53
  in Search_Directory action 256
<prefix>_namekey variable  53
  in Search_Directory action 257
<prefix>_node variable  53
  in Search_Directory action 257
<prefix>_pagedata  53
  in Put_User_Info action 241
  in Search_Directory action 257
<prefix>_pagephon variable  53
  in Search_Directory action 257
  Put_User_Info action 241
<prefix>_pagetype variable  53
  in Search_Directory action 257
  Put_User_Info action 241
<prefix>_phone variable  53
  in Search_Directory action 257
<prefix>_pswd variable  53
<prefix>_rtime variable  53
  in Put_User_Info action 241
  in Search_Directory action 257
<prefix>_userid variable  53
  in Search_Directory action 257

Numerics
3270_server variable  47
5250_server variable  47

A
Actions
Add_Record  160
Add_Stat_Item  161
alphabetic list  159
Assign_Resource  162
Branch  164
Calculate  166
Call_Extend_Cfg  168

Actions (continued)
Call_Extend_Init  170
Call_Extend_Ref  173
Call_Extend_Tran  175
Call_Referral  177
Call_Release  167
Call_Transfer  179
Check_Password  182
Clear_Tones  183
Close_Database  184
Comment  185
Compare_Chars  186
Compare_Numbers  187
Concatenate  188
Connect_Screen  189
Create_Variable  191
creating new  158
Delete_Record  192
Delete_Variable  193
Delete_Voice  194
Disconnect_Scr  195
Free_Resource  196
Get_a_Tone  210
Get_ADSI  197
Get_Length  199
Get_Messages  200
Get_Next_Record  204
Get_Record  206
Get_Screen_Data  208
Get_String  283
Get_Substring  209
Get_Tone_String  212
Get_Voice_Cont  215
Get_Voice_Resp  217
Get_Yes_No  286
Hang_up_Phone  221
Link_to_Appl  222
Log_a_Message  224
Log_Statistics  225
Pause  229
Place_a_Call  230
Play_Module  233
Play_Text  236
Play_Text_String  237
Play_Voice  238
Put_Tone_String  240
Put_User_Info  241
Receive_TDD  243
Record_Voice  248
Refresh_OIA  250
Remove_Stat_Item  254
Return_from_Appl  255
Actions (continued)
Search_Directory 256
Search_Screen 258
Search_String 260
Send_ADSI 261
Send_Alerts 263
Send_Keys_to_Scr 264
Send_TDD 269
Send_TDD String 270
Set_Timeout 271
Set_Variable 272
Start_Clock 273
Stop_Clock 274
Switch_Screen 275
Take_a_Messag 277
Update_Record 281
using 153
Wait_for_Call 288
Wait_for_Hang 290
Wait_Scr_Update 291
actions, voice
add password function 86
Add_Record action 160
parameters 160
Add_Stat_Item action 161
ADSI 147
ADSI application, creating 148
ADSI_cmn_appl variable 47
in Send_ADSI command 261
ADSI_cmn_srvr variable 47
in Send_ADSI command 261
ADSI_script_srvr variable 47
analog display services interface 147
analog display services interface (ADSI) actions 158
 Get_ADSI 197
Send_ADSI 261
ANI_DNIS_data variable 47
Application Control files
 Creating 38
application field 85
Application Variables
 creating or changing 44
application_name variable 47
in Link_to_Appl action 223
in Return_from_Appl action 255
applications, ADSI, creating 148
ASCII_server variable 47
Assign_Resource action 162
assigning a new value to a variable 272

B
Bellcore specifications 147
Branch action 164

C
Calculate action 166
Call_Agent_Rel 167
Call_Extend_Cfg 168
Call_Extend_Init 170
Call_Extend_Ref 173
Call_Extend_Tran 175
Call_Referral 177
Call_Transfer 179
Caller interaction actions
Clear_Tones 183
Get_a_Tone 210
Get_Tone_String 212
Get_Voice_Cont 215
Get_Voice_Resp 217
Play_Module 233
Play_Text 236
Play_Textus.String 237
VR_Get_String 283
VR_Get_Yesus.No 286
Canceling selections on voice application developer 11
case_sensitive variable 47
in Branch action 165
change password function 86
Changing
 application developer control file 46
application variables 44
voice logic module parameters 65
changing information in the directory 241
Check_Password action 182
Checking
 voice applications 36
Clear_Tones action 183
Close_Database action 184
close_database variable 47
in Add_Record action 160
in Delete_Record action 192
in Update_Record action 281
CM_key_reset variable 48
in Send_Keys_to_Scr action 265
CM32_server variable 47
collecting statistics and logging messages 157
Comment action 185
comments for voice logic modules 65
current text segments 135
common voice logic modules 68
Common voice segments 72
comnum.Compare_Numbers action 187
Compare_Chars action 186
comments about use 186
Comparing characters 186
Concatenate action 188
configuring text to speech 137
Connect_Screen action 189
Connecting to the host session 189
continuous voice recognition 114
Copying
  text segments 135
  voice logic modules 66
  voice segments 71
Create_Variable action 191
Creating
  Application Control File 38
  application developer control file 46
  application variables 44
  DirectTalk/2 ADSI applications 148
  new actions for voice programs 158
  simple application 13
  Variables 191
  voice logic modules 26
  voice programs 18
  Voice segments 33
creating a simple voice application 13
Design 13
Creating and using variable actions
  Calculate action 166
  Compare_Chars 186
  Compare_Numbers 187
  Concatenate 188
  Create_Variable 191
  Delete_Variable action 193
  Get_Length 199
  Get_Substring 209
  Search_String 260
  Set_Variable 272
creating and using variables 155
cs_applname variable 48
  in Log_Statistics action 225
cs_calldate variable 48
  in Log_Statistics action 225
cs_calltime variable 48
  in Log_Statistics action 225
cs_calltype variable 48
  in Log_Statistics action 225
cs_duration variable 48
  in Log_Statistics action 225
cs_hangstep variable 48
  in Log_Statistics action 225
cs_lineno variable 48
  in Log_Statistics action 226
cs_number variable 48
  in Log_Statistics action 225
cs_procedure variable 48
  in Log_Statistics action 225
cs_record variable 48
  in Log_Statistics action 226
cs_termtype variable 48
  in Log_Statistics action 226
currency in voice logic modules 69
current password field 86
D
Database actions
  Add_Record 160
  Close_Database 184
  Delete_Record 192
  Get_Next_Record 204
  Get_Record 206
  Update_Record 281
database_server variable 48
databases
  language identifiers 57
db_record_lock variable 48
    in Get_Next_Record action 204
    in Get_Record action 206
db_record_wait variable 48
    in Delete_Record action 192
    in Get_Next_Record action 205
    in Get_Record action 207
    in Update_Record action 281
delete password function 86
Delete_Record action 192
Delete_Variable action 193
Delete_Voice action 194
Deleting
  silence and unwanted background noise from voice
    segment 74
  Variables 193
  voice logic modules 66
  Voice records from a database 194
  voice segments 71
Designing
  Sample voice application 13
Determining the length of a string 199
dialing an outgoing call 230
directory_server variable 48
  in Check_Password action 182
  in Get_Messages action 201
  in Put_User_Info action 241
  in Search_Directory action 257
DirectTalk/2
  ADSI support 147
  ADSI user actions 147
  creating an ADSI application 148
  library xvi
  publications xvi
  version for password protection 86
DirectTalk/2 system error return codes
  DirectTalk/2 voice programs 62
  REXX voice programs 97
DirectTalk/2 windows
  Canceling selections 11
  Getting help 12
  Making selections 11
  Using function keys 11
  Using Voice Application Developer windows 9

Index 393
Disconnect_Scr action 195
Disconnecting from the host application 195
discrete alphanumeric recognition 113
dtlk_edit_x 48
dtmf_escape_char variable 48
   in Get_Tone_String 214
dtmf_term_char variable 48
   in Get_Tone_String 214

E
  e_full_tran_pos 48
e_lines_for_ref 48
e_lines_for_tran 48
e_min_transc 49
e_num_ext_lines 48
e_r_canheara 49
e_r_canhear_c 49
e_r_canplayto_a 49
e_r_canplayto_c 49
e_w_canheara 49
e_w_canhear_c 49
e_w_canplayto_a 49
e_w_canplayto_c 49
Editing
   voice logic modules 64
   voice programs 59
   Voice segments 73
   editing multiple voice segments 75
   emulators supported 123
   Enhanced sample application
   entry_fail_flag variable 48
      in Get_String action 285
ev_bEEP_flag 49
      in VR_Get_Yes_No action 287
ev_no_bEEP 49
      in VR_Get_String action 285
Exiting the voice application developer 12
  ext_canheara 49
  ext_canhear_c 49
  ext_canplayto_a 49
  ext_canplayto_c 49
  ext_connect_type 49
  ext_fullreferral_pos 49
  ext_fulltran_pos 49
  ext_line2_trunk 49
  ext_line2_ts 49
  ext_monagent_hup 49
  ext_moncall_hup 49
  ext_num_of_lines 49
  ext_switched_at 49
Extracting a portion of a string 209

F
  FDM (Feature Download Management) 147
  Feature Download Management 147
  file type field 85
  Flowcharts
     For the TIME application 14
  Forms
     Free_Resource action 196
     functions
        voice logic modules 63

G
  Get_a_Tone action 210
  Get_ADSI action 197
  Get_ADSI user action 147
  Get_Length action 199
  Get_Messages action 200
  Get_Next_Record action 204
  Get_Record action 206
  Get_Screen_Data action 208
  Get_Substring action 209
  Get_Tone_String action 212
  Get_Voice_Cont action 215
     Comments about use 216
     Parameters 215
     system variables 216
  Get_Voice_Resp action 217
     Return codes 217
  Getting a record from an application database 206
  Getting help on voice application developer windows 12
  Getting information from the simulated screen 208
gmsg_msg_db variable 49
     in Get_Messages action 201
gmsg_msg_server variable 50
     in Get_Messages action 201
gmsg_notebook variable 50
     in Get_Messages action 201
gmsg_phone variable 50
     in Get_Messages action 201

H
  handle_acterr variable 50
  Hang_up_Phone action 221
  hanging up the telephone and ending the call 221
  host access, multiple 125
     with communications applications 126
  Host application interface actions
    Connect_Screen 189
    Disconnect_Scr 195
    Get_Screen_Data 208
    Refresh_OIA 250
    Search_Screen 258
    Send_Alert 263
Host application interface actions (continued)
  Send_Keys_to_Scr 264
  Switch_Screen 275
  Wait_Scr_Update 291
host communications feature 123
  actions 126
  debugging applications 126
  emulators supported 123
  multiple session access 125
  operator information area (OIA) 124
programming considerations 125

I
  initial_appl variable 50
  Inserting
  interacting with the caller 154
  interfacing with a host application 156

K
  key_immediate variable 50
    in Send_Keys_to_Scr action 265

L
  language field 85
  languages
    database identifiers 57
    multiple, using 57
    multiple, using 58
  last_cont_resp1 variable 50
    in Get_Voice_Con cont action 216
  last_cont_resp2 variable 50
    in Get_Voice_Con cont action 216
  last_dtmf_data variable 50
    in Get_Tone_String action 214
  last_dtmf_tone variable 50
    in Get_a_Tone action 211
  last_found_column variable 50
    in Search_Screen action 258
    in Wait_Scr_Update action 292
  last_found_row variable 50
    in Search_Screen action 258
    in Wait_Scr_Update action 292
  last_repeat variable 50
    in Play_Module action 234
  last_voice_resp variable 50
    in Get_Voice_Resp action 218
    in Play_Module action 234
    in Play_Voice action 239
    in VR_Get_String action 285
    in VR_Get_Yes_No action 287
  Link_to_Appl action 222
  linked_from_appl variable (continued)
    in Return_from_Appl action 255
  linking to an running a subprogram 222
  list of actions 159
  Literals
    Log_a_Message action 224
    Log_Statistics action 225
    logging a message 224
    logging statistics 225

M
  mailbox_server variable 50
    in Get_Messages action 201
    in Take_a_Message action 278
  Making selections on voice application developer
    windows 11
  more about voice applications 55
  more about voice logic modules 63
  more about voice programs 59
  more about voice segments 71
  multiple languages, using 58
  multiple voice segments, editing and recording 75

N
  Naming
    a voice application 17
    new password field 86

O
  OIA (operator information area ) 124
  oia_data variable 50
    in Refresh_OIA action 250
  oia_data_ascii variable 50
    in Refresh_OIA action 250
  Opening
    Another voice application 55
    operator information area (OIA) 124
  Optional features
    host communications feature 123
    TDD (Telecommunication Devices for the Deaf) 145
    Telecommunication Devices for the Deaf(TDD) 145
    voice messaging 101
    voice recognition 103

P
  Parameters
    creating or changing application developer control
      file 46
    creating or changing application variables 44
    system variables 47
    Types 43
  User Variables 52
password
  adding 86
  application field 85
  converting databases 86
  current password 86
  deleting 86
  file type field 85
  functions
    add password 86
    change password 86
    delete password 86
  language field 85
  new password 86
  prompt 87
  security 85
  using 85
  verify password 86
  version of DirectTalk/2 86
Pause action 229
pausing the application 229
phonetic input 140
Place_a_Call action 230
Play_Module action 233
Play_Text action 129, 236
Play_Text_String action 129, 237
Play_Voice action 238
  playing text segments 129
Playing voice records 238
previous_rc variable 50
previous_step variable 50
previous_step2 50
Printing
  a voice application 56
  Voice segments 72
publications for DirectTalk/2 xvi
Put_Tone_String action 240
Put_User_Info action 241

Q
quiesce
  Place_a_Call action 231
  Wait_for_Call action 288

R
Receive_TDD action 243
recognizing voice response from a caller 217
Recognizing words spoken by a caller 215
Record_Voice action 248
Recording
  caller’s voice response 248
  telephone messages from callers 277
  Voice segments 33, 76
  recording multiple voice segments 75
recording_time variable 50
  in Record_Voice action 249
Refresh_OIA action 250
Remove_Stat_Item action 254
removing a variable from a statistics record 254
Renumbering
Repeat value
  repeat value for voice logic modules 65
resp_escape_char variable 51
  in Get_Voice_Resp action 219
resp_term_char variable 51
  in Get_String action 285
  in Get_Voice_Resp action 218
  in Play_Module action 234
  in Play_Voice action 239
retrieving a directory entry 256
Retrieving messages from a mailbox 200
return processing from a subprogram 255
Return_from_Appl action 255
REXX
  global variable 95
REXX feature 89
REXX Voice Program
    Linking voice programs 95
Running
  a voice logic module using the Play_Module
    action 233

S
Sample application
  creating 13
  Designing 13
  Flowchart for 14
Sample Application for Employee Scheduling 307
sample host application 309
sample voice application form 297
SDC (Server Display Control) 147
Search_Directory action 256
Search_Screen action 258
Search_String action 260
searching a string 260
Send_ADSI action 261
Send_ADSI user action 147
Send_Alert action 263
Send_Keys_to_Scr action 264
Send_TDD action 269
Send_TDD String action 270
sending alerts 263
Server Display Control 147
Set_Timeout action 271
Set_Variable action 58, 272
Setting Break Points 61
simulating keystrokes 264
speech synthesis 129
Start_Clock action 273
stat_server variable 51
state_server variable 51
Statistics and Logging actions
  Add_Stat_Item 161
  Log_a_Message 224
  Log_Statistics 225
  Remove_Stat_Item 254
Start_Clock 273
Stop_Clock 274
Step details
Steps
Stop_Clock action 274
structuring a voice application 156
Switch_Screen action 275
synthesize voice from text 129, 236, 237
sys_logic_srvr variable 51
sys_seg_srvr 51
System variables
  <prefix>_idno 241
  3270_server 47, 189
  5250_server 47, 189
  ADSI_cmn_appl 47, 261
  ADSI_cmn_srvr 47, 261
  ADSI_script_srvr 47
  ANI_DNIS data 47
  application_name 47, 223, 255
  ASCII_server 47
  case_sensitive 47, 165, 260
  close_database 47, 160, 192, 281
  CM_key_reset 48, 265
  CM32_server 47
  cs_applname 48, 225
  cs_calldate 48, 225
  cs_calltime 48, 225
  cs_caltype 48, 225
  cs_duration 48, 225
  cs_hangstep 48, 225
  cs_lineno 48, 226
  cs_number 48, 225
  cs_procedure 48, 226
  cs_record 48
  cs_termtype 48, 226
  database_server 48
  db_record_lock 48, 204, 206
  db_record_wait 48, 192, 205, 207, 281
  directory_server 48, 182, 201, 241, 257
dtlk_edit_x 48
dtmf_escape_char 48, 214
dtmf_term_char 48, 214
e_full_tran_pos 48
e_lines_for_tran 48
e_lines_for_ref 48
e_min_tran_scrn 49
e_num_ext_scrn 49
e_r_canhear_a 49
System variables (continued)
e_r_canhear_c 49
e_r_canplayto_a 49
e_r_canplayto_c 49
e_w_canhear_a 49
e_w_canhear_c 49
e_w_canplayto_a 49
e_w_canplayto_c 49
entry_fail_flag 48, 285
evr_beep_flag 49, 287
evr_no_beep 49, 285
ext_canhear_a 49
ext_canhear_c 49
ext_canplayto_a 49
ext_canplayto_c 49
ext_connect_type 49
ext_full_referral_pos 49
ext_fulltran_pos 49
ext_line2_trunk 49
ext_line2_ts 49
ext_monagent_hup 49
ext_moncall_hup 49
ext_num_of_lines 49
ext_switched_at 49
gmsg_msg_db 49, 201
gmsg_msg_server 50, 201
gmsg_notebook 50, 201
gmsg_phone 50, 201
handle_acterr 50
initial_appl 50
key_immediate 50, 265
last_cont_resp1 50, 216
last_cont_resp2 50, 216
last_dtmf_data 50, 214
last_dtmf_tone 50, 211
last_found_col 258, 292
last_found_column 50
last_found_row 50, 258, 292
last_repeat 50, 234
last_voice_resp 50, 218, 234, 239, 285, 287
linked_from_appl 50, 223, 255
mailbox_server 50, 201, 278
oia_data 50, 250
oia_data_ascii 50, 250
previous_rc 50
previous_step 50
previous_step2 50
recording_time 50, 249
resp_escape_char 51
resp_term_char 51, 218, 219, 234, 239, 285
stat_server 51
state_server 51
sys_logic_srvr 51
sys_seg_srvr 51
system_date 51
system_dow 51
System variables (continued)
  system_time 51
  table 47
  telephone_number 51
  term_application 51
  text_segmt_srvr 51
  timeout_flag 51, 211, 214, 219, 235, 287
  timeout_value 51, 234
  tmsg_g/n_db 51, 278
  tmsg_g/n_greet 51, 279
  tmsg_g/n_name 51, 277, 279
  tmsg_g/n_server 51, 278
  tmsg_grtg_exit 52, 279, 280
  tmsg_msg_db 52, 278
  tmsg_msg_server 52, 278
  tmsg_page_data 52, 280
  tmsg_page_phone 52, 279
  tmsg_page_type 52, 279
  tmsg_phone 52, 277, 279
  tmsg_rtime 52, 277
  voice_language 52
  voice_line 52
  voice_logic_srvr 52
  voice_segmt_srvr 52
  voice_server 52
  system-supplied voice segments 72
  system_date variable 51
  system_dow variable 51
  system_time variable 51

T
  T-REXX 95
  Table 52
  Take_a_Message action 277

TDD
  Assign_Resource 162
  Free_Resource 196
  Receive_TDD action 243
  Send_TDD action 269
  Send_TDD String action 270

Telecommunication Devices for the Deaf actions 158
  Receive_TDD 243
  Send_TDD 269
  Send_TDD String 270

Telecommunication Devices for the Deaf feature 145
  telephone_number variable 51

Telephony actions 153
  Assign_Resource 162
  Call_Agent_Rel 167
  Call_Extend_Cfg 168
  Call_Extend_Init 170
  Call_Extend_Ref 173
  Call_Extend_Tran 175
  Call_Referral 177
  Call_Transfer 179

Telephony actions (continued)
  Free_Resource 196
  Hang_up_Phone 221
  Place_a_Call 230
  Put_Tone_String 240
  Set_Timeout 271
  Wait_for_Call 288
  Wait_for_Hang 290
  term_application variable 51

Testing
  a voice application 37
  Voice logic modules 67

Text Segment Editor
  text segments
    common, using 135
    copying 135
    exporting 135
    importing 135
    playing 129

Text-to-speech 129
  BeST phonetic table 142
  configuring 137
  exception dictionary 142
  L&H phonetic table 141
  Play_Text action 236
  Play_Text_String action 237
  using phonetic input 140

Text segments
  common, using 135
  copying 135
  exporting 135
  importing 135
  playing 129

Text-to-speech 129
  BeST phonetic table 142
  configuring 137
  exception dictionary 142
  L&H phonetic table 141
  Play_Text action 236
  Play_Text_String action 237
  using phonetic input 140

Text segments
  common, using 135
  copying 135
  exporting 135
  importing 135
  playing 129

Time-out value
  time-out value for voice logic modules 65

timeout_flag variable 51
  in Get_a_Tone action 211
  in Get_a_Tone_String action 214
  in Get_Voice_Resp action 219
  in Play_Module action 235
  in VR_Get,Yes_No action 287

timeout_value variable 51
  in Play_Module action 234
  tmsg_g/n_db variable 51
  in Take_a_Message action 278
  tmsg_g/n_greet variable 51
  in Take_a_Message action 279
  tmsg_g/n_name variable 51
  in Take_a_Message action 279
  tmsg_g/n_server variable 51
  in Take_a_Message action 278
  tmsg_grtg_exit variable 52
  in Take_a_Message action 279, 280
  tmsg_msg_db variable 52
  tmsg_msg_db variable 278
  tmsg_msg_server variable 52
  in Take_a_Message action 278
  tmsg_page_data variable 52
  in Take_a_Message action 280
  tmsg_page_phone variable 52
  in Take_a_Message action 279
tmsg_page_type 52
  in Take_a_Message action 279
msg_phone variable 52
  in Take_a_Message action 279
tmsg_rtime 52
transaction messaging 310
TTS 129

U
understanding the voice application developer 7
Update_Record action 281
user actions for ADSI 147
User variables
  <prefix>_dept 53, 256
  <prefix>_fnm 53, 256
  <prefix>_grtgkey 53, 256, 257, 279
  <prefix>_idno 53, 256
  <prefix>_lnm 53, 256
  <prefix>_loc 53, 256
  <prefix>_mailbox 53, 256
  <prefix>_namekey 53, 257, 279
  <prefix>_node 53, 257
  <prefix>_pagedata 53, 241, 257
  <prefix>_pagephon 53, 241, 257
  <prefix>_pagetype 53, 241, 257
  <prefix>_phone 53, 257
  <prefix>_pswd 53, 241, 257
  <prefix>_rtime 53, 241, 257
  <prefix>_userid 53, 257
  assigning a new value to 272
  Creating 191
  using
    common text segments 135
    common voice logic modules 68
    common voice segments 72
    currency in voice logic modules 69
    system supplied voice segments 72
  using actions 153
  Using function keys on voice application developer windows 11
  Using Voice Application Developer windows 9

V
VAD (Voice Application Developer) 148
Variables
  See also application, system, and user variables
  Deleting 193
  Types 43
  verify password field 86
  voice actions
  Voice Application Developer
    add password function 86
    application field 85
    change password function 86
Voice Application Developer (continued)
  components of 7
  current password 86
  delete password function 86
  Exiting 12
  file type field 85
  language field 85
  new password 86
  password security 85
  using 85
  Starting 9
  verify password 86
voice application developer (VAD) 148
Voice application structuring actions
  Branch 164
  Comment 185
  Link_to_Appl 222
  Pause 229
  Return_from_Appl 255
voice application variables 5
Voice applications
  checking 36
  Creating the Application Control File 38
  creating the voice logic modules for 26
  creating the voice segments for 33
  Flowchart for TIME application 14
  more about 55
  naming 17
  Opening another 55
  printing 56
  setting languages 57
  testing 37
  with multiple languages 58
voice cutthrough 111
Voice Logic Module Editor
Voice logic modules
  common, using 68
  copying 66
  creating 26
  deleting 66, 71
  functions 63
  more about 63
  printing 67
  Testing 67
  using currency 69
voice messaging 157
Voice messaging actions
  Check_Password 182
  Get_Messages 200
  Put_User_Info 241
  Search_Directory 256
  Take_a_Message 277
voice messaging feature 101
Voice pattern
  adding silence 75
  deleting silence and unwanted background noise 74
Voice programs
creating 18
creating new actions 158
editing 59
voice recognition
Get_String action 283
Get_Yes_No action 286
voice recognition feature 103
continuous voice recognition 114
discrete alphanumeric recognition 113
voice cutthrough 111
voice stop 103
Voice segments
adding silence 75
common, using 72
copying 71
creating 33
deleting silence and unwanted background
noise 74
Editing 73
more about 71
playing 71
Printing 72
recording 33, 76
system supplied 72
voice stop 103
Voice-related actions 155
Delete_Voice 194
Play_Voice 238
Record_Voice 248
voice_language variable 52
using 57
voice_line variable 52
voice_logic_srvr variable 52
voice_segmt_srvr variable 52
voice_server variable 52
VR_Get_String action 283
VR_Get_Yes_No action 286

W
wait for and answer incoming calls 288
Wait_for_Call action 288
Wait_for_Hang 290
Wait_Scr_Update action 291
where to find more information xvi
working with a database 157
Sending your comments to IBM

Application Development User's Guide

SB35-4408-03

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

• By mail, use the Readers’ Comment Form.

• By fax:
  – From outside the U.K., after your international access code use 44 1962 870229
  – From within the U.K., use 01962 870229

• Electronically, use the appropriate network ID:
  – IBM Mail Exchange: GBIIBM2Q9 at IBMMAIL
  – IBMLink: WINVMD(IDRCF)
  – Internet: idrcf@winvmd.vnet.ibm.com

Whichever you use, ensure that you include:

• The publication number and title
• The page number or topic to which your comment applies
• Your name and address/telephone number/fax number/network ID.
Readers’ Comments
Application Development User’s Guide

SB35-4408-03

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Telephone

Email
You can send your comments POST FREE on this form from any one of these countries:

<table>
<thead>
<tr>
<th>Australia</th>
<th>Finland</th>
<th>Iceland</th>
<th>Netherlands</th>
<th>Singapore</th>
<th>United States</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belgium</td>
<td>France</td>
<td>Israel</td>
<td>New Zealand</td>
<td>Spain</td>
<td>Sweden</td>
</tr>
<tr>
<td>Bermuda</td>
<td>Germany</td>
<td>Italy</td>
<td>Norway</td>
<td>Switzerland</td>
<td>United Arab Emirates</td>
</tr>
<tr>
<td>Cyprus</td>
<td>Greece</td>
<td>Luxembourg</td>
<td>Portugal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Denmark</td>
<td>Hong Kong</td>
<td>Monaco</td>
<td>Republic of Ireland</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

By air mail  
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

REPONSE PAYEE  
GRANDE-BRETAGNE

IBM United Kingdom Laboratories 
Information Development Department (MP095) 
Hursley Park, 
WINCHESTER, Hants 
SO21 2ZZ United Kingdom

From: Name _______________________________________
      Company or Organization __________________________
      Address ________________________________________
      EMAIL _________________________________________
      Telephone ______________________________________