Lucent Technologies
Bell Labs Innovations

DEFINITY®
Business Communications System and GuestWorks®
Call Vectoring Guide

555-231-107
Comcode 108436528
Issue 1
June 1999

Notice
Every effort was made to ensure that the information in this book was complete and accurate at the time of printing. However, information is subject to change.

Your Responsibility for Your System's Security
Toll fraud is the unauthorized use of your telecommunications system by an unauthorized party, for example, persons other than your company's employees, agents, subcontractors, or persons working on your company's behalf. Note that there may be a risk of toll fraud associated with your telecommunications system and, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

You and your system manager are responsible for the security of your system, such as programming and configuring your equipment to prevent unauthorized use. The system manager is also responsible for reading all installation, instruction, and system administration documents provided with this product in order to fully understand the features that can introduce risk of toll fraud and the steps that can be taken to reduce that risk. Lucent Technologies does not warrant that this product is immune from or will prevent unauthorized use of common-carrier telecommunication services or facilities accessed through or connected to it. Lucent Technologies will not be responsible for any charges that result from such unauthorized use.

Lucent Technologies Fraud Intervention
If you *suspect that you are being victimized* by toll fraud and you need technical support or assistance, call Technical Service Center Toll Fraud Intervention Hotline at 1 800 643-2353.

Federal Communications Commission Statement
Part 15: Class A Statement. This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Part 68: Network Registration Number. This equipment is registered with the FCC in accordance with Part 68 of the FCC Rules. It is identified by FCC registration number AS593M-13283-MF-E.

Part 68: Answer-Supervision Signaling. Allowing this equipment to be operated in a manner that does not provide proper answer-supervision signaling is in violation of Part 68 Rules. This equipment returns answer-supervision signals to the public switched network when:

- Answered by the called station
- Answered by the attendant
- Routed to a recorded announcement that can be administered by the CPE user.

This equipment returns answer-supervision signals on all DID calls forwarded back to the public switched telephone network. Permissible exceptions are:

- A call is unanswered
- A busy tone is received
- A reorder tone is received.

Canadian Department of Communications (DOC) Interference Information
This digital apparatus does not exceed the Class A limits for radio noise emissions set out in the radio interference regulations of the Canadian Department of Communications.

Le Présent Appareil Nomérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la class A préscrites dans le reglement sur le brouillage radioélectrique édicté par le ministére des Communications du Canada.

Trademarks
Callmaster, CentreVu, AUDIX, DEFINITY, and GuestWorks are registered trademarks of Lucent Technologies. Best Service Routing is a trademark of Lucent Technologies.

Ordering Information
Call: Lucent Technologies BCS Publications Center
Voice 1-800-457-1235 International Voice 317-322-6416
Fax 1-800-457-1764 International Fax 317-322-6699
Write: Lucent Technologies BCS Publications Center
2855 N. Franklin Road
Indianapolis, IN 46219
Order: Document No. 555-231-107
Comcode 108436528
Issue 1, June 1999
For additional documents, refer to the references in "About This Document."

You can be placed on a standing order list for this and other documents you may need. Standing order will enable you to automatically receive updated versions of individual documents or document sets, billed to account information that you provide. For more information on standing orders, or to be put on a list to receive future issues of this document, contact the Lucent Technologies Publications Center.

Product Support
To receive support on your product, call 1-800-242-2121. Outside of the continental United States, contact your local Lucent Technologies authorized representative.

European Union Declaration of Conformity
The "CE" mark affixed to the equipment described in this book indicates that the equipment conforms to the following European Union (EU) Directives:

- Electromagnetic Compatibility (89/336/EEC)
- Low Voltage (73/23/EEC)
- Telecommunications Terminal Equipment (TTE) i-CTR3 BRI and i-CTR4 PRI

For more information on standards compliance, contact your local distributor.

Comments
To comment on this document, return the comment form located at the back of this book.

Lucent Technologies Web Page
The World Wide Web home page for Lucent Technologies is http://www.lucent.com

Acknowledgment
This document was prepared jointly by the Customer Training and Information Products Organization and the Information Development Organization for Global Learning Solutions
Lucent Technologies Bell Laboratories,
Denver, CO 80234-2703.

Contents

Contents **iii**

About This Document **xi**

- Reasons for Reissue xi
- Background xii
- Contents and Organization xii
- Intended Audience xiii
- Conventions xiv
- References xiv

1 Call Vectoring Tutorial **1**

- Entering the Vector On-Line 1
- Constructing a Vector: One Approach 6
  - Phase 1: Queuing a Call to the Main Split 6
  - Phase 2: Providing Feedback and Delay Announcement 7
  - Phase 3: Repeating Delay Announcement and Feedback 9
  - Phase 4: Queuing a Call to a Backup Split 10
  - Phase 5: Checking the Queue Capacity 11
  - Phase 6: Checking for Non-Business Hours 12
- Benefits of Call Vectoring 13

2 Call Vectoring Fundamentals **17**

- Managing Your Calls 17
  - Call Flow 18
  - Caller Control 19
  - Call Queuing to Splits 19
  - Agent Work Mode 20
  - Calling Party Feedback 21
  - Dialed Number Identification Service (DNIS) 22
- Vector Processing 24
  - Vector Directory Number 24
  - Vector Control Flow 28
  - Programming Capabilities 30

- Security 33
  - Limiting Outside Access Using VDN COR Restrictions 33
  - Voice Response Integration 34

3 Basic Call Vectoring 35

- Command Set 35
- Providing Call Treatments 36
  - Announcements 37
  - Delays with Audible Feedback 39
  - Busy Tone 40
  - Disconnect 41
  - Voice Response Scripts 42
- Routing Calls 45
  - Queuing Calls to ACD Splits 45
  - Leaving Recorded Messages 47
  - Sending Calls to a Vector-Programmed Number 49
- Service Observing 50
- Branching/Programming 51
  - Unconditional Branching 51
  - Conditional Branching 52
  - Stopping Vector Processing 53
- Vector Chaining 54

4 Call Prompting 55

- Command Set 56
- Touch-Tone Collection Requirements 57
- Call Prompting Digit Entry 58
  - Removing Incorrect Digit Strings 58
  - Entering Variable-Length Digit Strings 59
  - Entering Dial-Ahead Digits 60
- Functions and Examples 60
  - Treating Digits as a Destination 61
  - Using Digits to Collect Branching Information 62
  - Using Digits to Select Options 63
  - Displaying Digits on the Agent's Set 63
- Dial-Ahead Digits 65

5 Call Vectoring Applications 69

- Customer Service Center 70
- Automated Attendant 71
- Dial-by-Name Feature 72
- Data In/Voice Answer (DIVA) and Data/Message Collection 75
- Vector Exercises 79
  - Exercise 1: Emergency and Routine Service 79
  - Exercise 2: Late Caller Treatment 82
  - Exercise 3: Messaging Option 84

A Call Vectoring Commands 87

- Command Description/Reference 88
- Command/Option Summary 89
- Command Job Aid 90
- Command Directory 94
- Announcement 95
  - Purpose 95
  - Syntax 95
  - Valid Entries 95
  - Requirements 95
  - Example 95
  - Operation 95
  - Answer Supervision Considerations 96
  - Feature Interactions 96
  - BCMS Interactions 96
- Busy 97
  - Purpose 97
  - Syntax 97
  - Requirements 97
  - Operation 97
  - Answer Supervision Considerations 97
  - Feature Interactions 97
  - BCMS Interactions 97

- Check 98
  - Purpose 98
  - Syntax 98
  - Valid Entries 98
  - Requirements 98
  - Examples 98
  - Operation 99
  - Answer Supervision Considerations 100
  - Feature Interactions 100
  - BCMS Interactions 100
- Collect Digits 101
  - Purpose 101
  - Syntax 101
  - Valid Entries 101
  - Requirements 101
  - Example 101
  - Operation 101
  - Answer Supervision Considerations 104
  - Feature Interactions 104
  - BCMS Interactions 104
- Converse-on 105
  - Purpose 105
  - Syntax 105
  - Valid Entries 105
  - Requirements 105
  - Examples 105
  - Operation 106
  - Answer Supervision Considerations 108
  - Feature Interactions 109
  - BCMS Interactions 114

- Disconnect 115
  - Purpose 115
  - Syntax 115
  - Valid Entries 115
  - Requirements 115
  - Example 115
  - Operation 115
  - Answer Supervision Considerations 115
  - Feature Interactions 116
  - BCMS Interactions 116
- Goto Step 117
  - Purpose 117
  - Syntax 117
  - Valid Entries 118
  - Requirements 119
  - Examples 119
  - Operation 119
  - Answer Supervision Considerations 120
  - Feature Interactions 120
  - BCMS Interactions 120
- Goto Vector 121
  - Purpose 121
  - Syntax 121
  - Valid Entries 122
  - Requirements 123
  - Examples 123
  - Operation 123
  - Answer Supervision Considerations 124
  - Feature Interactions 124
  - BCMS Interactions 124

- Messaging 125
  - Purpose 125
  - Syntax 125
  - Valid Entries 125
  - Requirements 125
  - Examples 125
  - Operation 125
  - Answer Supervision Considerations 126
  - Feature Interactions 127
  - BCMS Interactions 127
- Queue-to 128
  - Purpose 128
  - Syntax 128
  - Valid Entries 128
  - Requirements 128
  - Examples 128
  - Operation 128
  - Answer Supervision Considerations 129
  - Feature Interactions 129
  - BCMS Interactions 129
- Route-to 130
  - Purpose 130
  - Syntax 130
  - Valid Entries 130
  - Requirements 130
  - Examples 131
  - Operation 131
  - Answer Supervision Considerations 133
  - Feature Interactions 133
  - BCMS Interactions 135

- Stop 136
  - Purpose 136
  - Syntax 136
  - Requirements 136
  - Operation 136
  - Answer Supervision Considerations 136
  - Feature Interactions 136
  - BCMS Interactions 136
- Wait-time 137
  - Purpose 137
  - Syntax 137
  - Valid Entries 137
  - Requirements 137
  - Examples 138
  - Operation 138
  - Answer Supervision Considerations 138
  - Feature Interactions 139
  - BCMS Interactions 139
- Criteria for Success/Failure of Call Vectoring Commands 140

B Call Vectoring Management 145

- Call Vectoring Feature Requirements 145
- Enabling the Vector Disconnect Timer 147
- Upgrading to a Call Vectoring Environment 147
- Changing and Testing the Vector 148

C Considerations for the Vectoring Features 149

- Basic Call Vectoring Considerations 149
- Call Prompting Considerations 151
- Transferring Calls to VDNs Considerations 152
- Feature Capacities 153
- Call Vectoring Features Not Supported 154

D **Troubleshooting Vectors** **155**

- Unexpected Feature Operations 155
- Unexpected Command Operations 156
- Converse-on Command Debugging 163
- Tracking Unexpected Vector Events 166
  - Display Events Form 166
  - Display Events Report 167
  - Summary of Vector Events 168

E **Interactions Between Call Vectoring and BCMS** **175**

- BCMS Tracking in a Call Vectoring Environment 175
  - Defining and Interpreting Call Flows 176
- Using BCMS Reports to Evaluate Call Vectoring Activity 181

F **Operation Details for the Route-to Command** **183**

G **Detailed Call Flow and Specifications for Converse—VRI Calls** **189**

- Converse Call Placement 189
- Data Passing 191
- VRU Data Collection 195
- Script Execution 195
- Data Return 196
- Script Completion 199
- Switch Data Collection 200

H **Setting Up Agents in an ACD Hunt Group** **201**

- Call Vectoring Setup 202

I **Improving Performance** **211**

- Looping Examples 212
  - Audible Feedback 212
  - Check 214
- Other Examples 216
  - After Business Hours 216
- Relative Processing Cost of Vector Commands 217

GL **Glossary and Abbreviations** **219**

IN **Index** **225**

About This Document

This document provides information about the Call Vectoring feature as provided with the DEFINITY® Business Communications System (BCS) and GuestWorks® Issue 5/Release 7 and Issue 4/Release 6 offers. The Call Vectoring feature provides the following:

- Basic Call Vectoring
- Call Prompting.

The following Call Vectoring features are **not** available with this offer:

- G3V4 Enhanced (for example, specifying a priority level with the oldest-call-wait condition)
- Advanced Vector Routing
- ANI/II Digits Routing
- Call Information Forwarding (CINFO)
- Best Service Routing™
- Adjunct Routing
- Vector Directory Number (VDN) of Origin Announcement
- VDN Return Destination.

Limited Call Vectoring reports are available by purchasing the separate Basic Call Management System (BCMS) feature.

Reasons for Reissue

This is the first issue of this document.

Background

This document is based on the *DEFINITY® Enterprise Communications System (ECS) Release 6 Call Vectoring/Expert Agent Selection (EAS) Guide*, 555-230-521, Issue 2. This version was developed to support the DEFINITY BCS and GuestWorks offers. This document should be used only with those systems.

Contents and Organization

This document is organized as follows:

- About This Document
- Chapter 1, ‘‘Call Vectoring Tutorial’’
- Chapter 2, ‘‘Call Vectoring Fundamentals’’
- Chapter 3, ‘‘Basic Call Vectoring’’
- Chapter 4, ‘‘Call Prompting’’
- Chapter 5, ‘‘Call Vectoring Applications’’
- Appendix A, ‘‘Call Vectoring Commands’’
- Appendix B, ‘‘Call Vectoring Management’’
- Appendix C, ‘‘Considerations for the Vectoring Features’’
- Appendix D, ‘‘Troubleshooting Vectors’’
- Appendix E, ‘‘Interactions Between Call Vectoring and BCMS’’
- Appendix F, ‘‘Operation Details for the Route-to Command’’
- Appendix G, ‘‘Detailed Call Flow and Specifications for Converse—VRI Calls’’
- Appendix H, ‘‘Setting Up Agents in an ACD Hunt Group’’
- Appendix I, ‘‘Improving Performance’’
- ‘‘Glossary and Abbreviations’’
- Index.

Chapter 1 through Chapter 4 concentrates on illustrating Call Vectoring principles. Chapter 5 presents several Call Vectoring applications. Finally, the appendices, glossary, and index provide information and references to various Call Vectoring topics.

Intended Audience

The document is intended primarily for personnel who set up Call Vectoring. You should use this document as an information source for implementing Call Vectoring. A knowledge of Automatic Call Distribution (ACD) is required.

The level of your expertise in Call Vectoring should determine how you use the document. Users who are unfamiliar with Call Vectoring should read the overview, then study the tutorial. Users who want to learn more about Call Vectoring should review Chapter 1 through Chapter 5 to get a good grasp of how the Call Vectoring features function. Finally, advanced users of Call Vectoring may only find it necessary to periodically reference a specific appendix or two (such as Appendix A, which contains a set of Call Vectoring command “manual pages”) to get the information needed.

Users who want to set up a group of agents should read Appendix G.

Conventions

- The terms switch or system refer equally to the DEFINITY BCS and GuestWorks offers.
- Text in **bold font** represents commands.
- Text in `courier font` represents field names from screens.
- Text in *`courier italics font`* represents recorded announcement information in vector examples.
- Keys (such as TAB, RETURN) are in capital letters.

References

The publications listed in this section should be used to supplement the information presented in this document:

- *DEFINITY® Enterprise Communications Server Release 7 System Description*, 555-230-211
- *DEFINITY® Enterprise Communications Server Basic Call Management System (BCMS) Operations*, 555-230-706
- *DEFINITY® Enterprise Communications Server Release 7 Administrator's Guide*, 555-233-502
- *BCS Products Security Handbook*, 555-025-600.

Call Vectoring Tutorial

1

This chapter provides you with a practical start in using Call Vectoring and presents the basics you need to write a representative vector and to enter it on-line. The last section of the chapter summarizes the benefits of Call Vectoring, and it identifies example vectors in the reference section of the document that illustrate these benefits.

Entering the Vector On-Line

A vector is entered on-line via Basic Screen Administration by completing the Call Vector Form. This form appears on three screens as follows in Screen 1, Screen 2 and Screen 3.

```
change vector 20                                    Page 1 of 3
                            CALL VECTOR

 Number: 20                 Name:_______________                  Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n          CINFO? n           BSR? n

 01 _________________
 02 _________________
 03 _________________
 04 _________________
 05 _________________
 06 _________________
 07 _________________
 08 _________________
 09 _________________
 10 _________________
 11 _________________
```

Screen 1. Call Vector Form (Page 1 of 3)

```
change vector 20                                    Page 2 of 3
                              CALL VECTOR

  12 _________________
  13 _________________
  14 _________________
  15 _________________
  16 _________________
  17 _________________
  18 _________________
  19 _________________
  20 _________________
  21 _________________
  22 _________________
```

Screen 2. Call Vector Form (Page 2 of 3)

```
change vector 20                                    Page 3 of 3
                              CALL VECTOR

  23 _________________
  24 _________________
  25 _________________
  26 _________________
  27 _________________
  28 _________________
  29 _________________
  30 _________________
  31 _________________
  32 _________________
```

Screen 3. Call Vector Form (Page 3 of 3)

The following list summarizes how you can enter a vector on-line via Basic Screen Administration. For complete details on this process, consult the *DEFINITY® ECS Administrator's Guide*.

1. Access the Call Vector Form by executing the **change vector *x*** command, where ***x*** is the number of the vector you want to access. Use the **change vector** command either to change an existing vector, or to create a new vector.

   If you are not certain of the number or name of a vector, enter the **list vector** command to view a complete list of all vectors that have been administered for your system.

2. Assign a name to your vector by completing the blank next to the `Name` field. The vector name can contain up to 15 alphanumeric characters.

3. Set the `Lock?` field to **n**, if displayed.

4. Look at the next fields and note where a **y** (yes) appears. These fields indicate the Call Vectoring features and corresponding commands supported for DEFINITY BCS and GuestWorks. Only the following two sets of the Call Vectoring features are supported.

Basic	See Chapter 3, ''Basic Call Vectoring''.
Prompting	See Chapter 4, ''Call Prompting''.

   All other feature sets cannot be enabled.

5. Enter a maximum of 32 vector commands in the blanks next to the step numbers. See Appendix A for a complete description of all Call Vectoring commands.

   NOTE:
   You need not type every letter of each command that you enter. If you type just the first few letters of a command and press RETURN or TAB, the system spells out the entire command.

6. Save the vector in the system by pressing ENTER.

Enhanced Vector Editing

Enhanced Vector Editing allows you to insert and delete vector steps while editing a vector on the switch.

NOTE:
After editing a vector, be certain to verify that the vector will work as you intend it to. This is particularly important if you deleted a step that was the target of a *goto* step.

Inserting a Vector Step

To insert a vector step, complete the following procedure:

1. After entering the **change vector** command, press F6 (edit).
2. At the command line, type **i** followed by a space and then type the number of the step you would like to add. You cannot add a range of vector steps. Enter the command. For example, to insert a new vector step 3, type **i 3** and then enter the command.
3. Type the new vector step.

When a new vector step is inserted, the system automatically renumbers all succeeding steps and renumbers *goto* step references as necessary. Under certain conditions, attempts to renumber *goto* step references will result in an ambiguous renumbering situation. In this case, the step reference is replaced by an asterisk (*). You will receive a warning indicating that you must resolve the ambiguous references, and your cursor will automatically move to the first reference that needs to be resolved. You cannot save a vector with unresolved *goto* references.

You cannot insert a new vector step if 32 steps are already entered in the vector. However, you can extend the vector program to another vector by using the **goto vector unconditionally** command at step 32.

Deleting a Vector Step

To delete a vector step, complete the following procedure:

1. After entering the **change vector** command, press F6 (edit)
2. At the command line, type **d** followed by a space and then type the number of the step you would like to delete. Enter the command. You can delete a range of vector steps. For example, to delete steps 2 through 5, type **d 2-5** and then enter the command.

When a vector step is deleted, the system automatically renumbers all succeeding steps and renumbers *goto* step references as necessary. Under certain conditions, attempts to renumber *goto* step references will result in an ambiguous renumbering situation. In this case, the step reference is replaced by an asterisk (*).

For example, if a vector step that is the target of a *goto* step is deleted, the *goto* references are replaced by *s. For example, if you delete step 7 when you have a vector step *goto step 7 if* ..., the 7 is replaced by an *.

You will receive a warning indicating that you must resolve ambiguous references, and your cursor will automatically move to the first reference that needs to be resolved. You cannot save a vector with unresolved *goto* references.

Constructing a Vector: One Approach

This section provides you with one logical approach to constructing a vector. In so doing, the section presents a starting vector that consists of one step and then builds upon this vector to produce a new vector that provides additional functions. This vector building process continues through several phases until a final complete vector is constructed. As each phase is presented, you are introduced to one or more new vector commands and/or approaches to vector processing. While it is not practical to present all such commands and approaches along the way to constructing a single final vector, those presented in this tutorial should allow you to get a good grasp of how to use Call Vectoring.

Phase 1: Queuing a Call to the Main Split

If a call cannot be immediately answered by an agent (or operator), the call is usually queued until an agent becomes available. A call can be connected to an available agent or queued via the vector in Screen 4.

```
change vector 20                                    Page 1 of 3
                            CALL VECTOR

 Number: 20                 Name:_______________                  Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n          CINFO? n           BSR? n

 01 queue-to split 5 pri l
 02 _________________
 03 _________________
 04 _________________
 05 _________________
 06 _________________
 07 _________________
 08 _________________
 09 _________________
 10 _________________
 11 _________________
```

Screen 4. Queuing Call to Main Split

If an agent is available, the **queue-to split** command automatically sends the call to the agent without queuing the call. However, if no agent is available, the command queues the call to the main split (or group) of agents. Once the call is sent to the main split queue, the call remains there until either it is answered by an agent or some other treatment is provided.

Each call queued to a split occupies one queue slot in that split. Calls are queued sequentially as they arrive according to the assignment of the priority level. In our vector, note the priority level *low* is assigned to the call. The priority level establishes the order of selection for each call that is queued. A call can be assigned one of four priority levels: *top*, *high*, *medium*, or *low*. Within a given split (the main split, in our vector), calls are delivered to the agent sequentially as they arrive to the split queue and according to the priority level assigned. Accordingly, calls assigned a *top* priority (if any) are delivered to an agent first, calls that are assigned a *high* priority are delivered second, and so on.

Finally, note that the call is queued to Split 5.

Phase 2: Providing Feedback and Delay Announcement

In the last section we mentioned that a call remains queued until an agent becomes available to answer the call. In the meantime, the caller would no doubt like to hear some feedback assuring him or her that the call is being processed. The vector in Screen 5 provides one solution.

```
change vector 20                                  Page 1 of 3
                          CALL VECTOR

 Number: 20                Name:_______________                  Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n          CINFO? n           BSR? n

 01 queue-to split 5 pri l
 02 wait-time 10 secs hearing ringback
 03 announcement 2771
 04 _________________
 05 _________________
 06 _________________
 07 _________________
 08 _________________
 09 _________________
 10 _________________
 11 _________________
```

Screen 5. Providing Feedback and Delay Announcement

The **wait-time** command in step 2 provides a maximum 8-hour delay before the next vector step is processed (the 8-hour limit was an enhancement for Issue 5/R7; Issue 4/R6 and earlier systems were limited to a delay of 998 seconds in 2-second intervals). The time parameter may be assigned as follows:

- 0-999 secs
- 0-480 mins
- 0-8 hrs.

In our vector, the time specified is 10 seconds.

In addition to the delay period, the **wait-time** command provides the caller with feedback. In our vector, **ringback** is provided. Other types of feedback that can be provided with the **wait-time** command are: silence; system music; or an alternate audio/music source. For more information, see ‘‘Delays with Audible Feedback’’ on Page 39.

Theoretically, then, the **wait-time** command in our vector provides the caller with 10 seconds of ringback. But what happens if an agent answers the call before the **wait-time** command runs its course? If this happens, the command is terminated (that is, the delay period is ended and the accompanying feedback is stopped). So, returning to our example, let’s presume the call is delivered to an agent after 4 seconds. In such a case, the following is true:

- Caller does not hear the remaining 6 seconds of ringback, because delivering the call to the agent is the primary objective.
- Announcement in step 3 (discussed next) is not played.

If the call is not answered by the time the **wait-time** command in step 2 is completed, vector processing continues with the **announcement** command in step 3.

The **announcement** command consists of a recorded message, and is often used to encourage the caller to stay on the phone or to provide information to the caller. If a call is delivered to an agent during the **announcement** command, the announcement is interrupted. Otherwise, the announcement is played from beginning to end. Announcement 2771 could contain this message: “We’re sorry. All of our operators are busy at the moment. Please hold.” Thereafter, the call remains in queue until it is answered by an agent or until the caller hangs up. Multiple callers can be connected to an announcement at any time. See “Managing Announcements” in the *DEFINITY® ECS Administrator’s Guide* for more information about announcements.

Phase 3: Repeating Delay Announcement and Feedback

The vector in the previous section provides feedback to the caller after the call is queued. However, if the announcement in step 3 is played, and if the agent does not answer the call soon after the announcement is complete, the caller may end up holding the line for too long a time without receiving any further feedback or treatment. The vector in Screen 6 provides one solution.

```
change vector 20                                    Page 1 of 3
                          CALL VECTOR

 Number: 20                Name:_______________                    Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n          CINFO? n           BSR? n

 01 queue-to split 5 pri l
 02 wait-time 10 secs hearing ringback
 03 announcement 2771
 04 wait-time 60 secs hearing music
 05 goto step 3 if unconditionally
 06 _______________
 07 _______________
 08 _______________
 09 _______________
 10 _______________
 11 _______________
```

Screen 6. Repeating Delay Announcement and Feedback

The **wait-time** command in step 4 of this vector provides additional feedback (this time, music) to the caller. If the call is not answered by the time step 4 completes, the **goto step** command in step 5 is processed.

Up to this point, we have discussed and illustrated Call Vectoring commands that cause *sequential flow* (that is, the passing of vector processing control from the current vector step to the next sequential vector step). The **goto step** command is an example of a Call Vectoring command that causes *branching* (that is, the passing of vector processing control from the current vector step to either a preceding or succeeding vector step).

The **goto step** command in step 5 allows you to establish an announcement-wait loop that continues until the agent answers the call. Specifically, the command makes an unconditional branch to the **announcement** command in step 3. If the call is not answered by the time the announcement in step 3 is complete, control is passed to the **wait-time** command in step 4. If the call is still not answered by the time this command completes, control is passed to step 5, where the unconditional branch is once again made to step 3. As a result of the established loop, the caller is provided with constant feedback.

Phase 4: Queuing a Call to a Backup Split

Up to this point, we have dealt with a call queued to one split, the main split. However, Call Vectoring allows a call to be queued to a maximum of three splits simultaneously. If a call is queued to multiple splits, the call has a better chance of being answered more quickly. Multiple split queuing is especially useful during periods of heavy call traffic.

The vector in Screen 7 allows a call to queue to two splits.

```
change vector 20                                    Page 1 of 3
                           CALL VECTOR

 Number: 20                Name:_______________                  Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n           CINFO? n           BSR? n

 01 queue-to split 5 pri l
 02 wait-time 10 secs hearing ringback
 03 announcement 2771
 04 wait-time 10 secs hearing music
 05 check split 7 pri m if calls-queued < 5
 06 wait-time 60 secs hearing music
 07 announcement 2881
 08 goto step 5 if unconditionally
 09 _________________
 10 _________________
 11 _________________
```

Screen 7. Queuing Call to Backup Split

We have already discussed how the **queue-to split** command in step 1 queues the call to the main split. If the call is not answered by the time the **wait-time** command in step 4 completes, the **check split** command in step 5 attempts to queue the call to backup Split 7 at a medium priority. The condition expressed in the command (*if calls-queued* < *5*) determines whether the call is to be queued to the backup split. Specifically, if the number of calls currently queued to Split 7 at a medium or higher priority is less than 5, the call is queued to the split. Note that if the call is queued, the call in this case is assigned a *medium* priority instead of a *low* priority, which is assigned if the call is queued by the **queue-to split** command in step 1. It is a good practice to raise the priority level in subsequent queuing steps to accommodate callers who have been holding the line for a period of time. (We could have assigned a *high* or *top* priority instead of a *medium* priority in step 5.)

The *calls-queued* condition is one of several conditions that can be included in the **check split** command. The other conditions are *unconditionally, available agents, staffed agents*, and *oldest call waiting*. As is true for the **queue-to split** command, the **check split** command can queue a call at one of four priorities: *low, medium, high* or *top*.

We are including a queuing step within the loop, thus giving the call repeated opportunities to queue (if necessary). The call queues to split 7 only once. Announcement 2881 provides a different message, for example, “We are very sorry about your wait. One of our operators will assist you as soon as possible. Thank you.”

Phase 5: Checking the Queue Capacity

It is a good practice to check the main split queue for the number of calls already queued before allowing another call to queue to the split because there is a limited number of queue slots assigned to each split. The number of such slots assigned to each split is defined in the queue length field on the hunt group screen. A call that attempts to queue to a split with no available queue slots cannot be queued to that split and, accordingly, the **queue-to split** command fails. Vector processing would then continue with the next vector step. The vector in Screen 8 contains provisions for checking queue capacity.

```
change vector 20                                    Page 1 of 3
                           CALL VECTOR

 Number: 20                Name:_______________                    Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n           CINFO? n           BSR? n

 01 goto step 10 if calls-queued in split 5 pri l > 20
 02 queue-to split 5 pri l
 03 wait-time 10 secs hearing ringback
 04 announcement 2771
 05 wait-time 10 secs hearing music
 06 check split 7 pri m if calls-queued < 5
 07 wait-time 60 secs hearing music
 08 announcement 2881
 09 goto step 6 if unconditionally
 10 busy
 11 _______________
```

Screen 8. Checking Queue Capacity

A check of split 5 is implemented by the **goto step** command in step 1. In Screen 8, 21 slots are assigned to split 5 (that is, the queue length for split 5 is *21*). Accordingly, the **goto step** command tests whether the split contains more than 20 calls of any priority level via the condition *if calls-queued in split 5 > 20 pri l*. If this test is true, control is passed to the **busy** command in step 10. The **busy** command gives the caller a busy signal and eventually causes the call to drop.

On the other hand, if 20 or fewer calls are queued to the main split when step 1 executes, the **queue-to split** command in step 2 queues the call, and vector processing continues at step 3.

NOTE:
Instead of providing the caller with a busy tone if the *queue-to split* step cannot queue the call, we can queue the call to another split that is designed to serve as a backup split. To do this, we can change the step parameter for the *goto step* command from 10 to 6 (so that the command reads *goto step 6.....*). In such a case, control is passed from step 1 to the *check split* step (step 6). Inasmuch as this queuing step is included within a continuous loop of steps (steps 6 through 9), continuous attempts to queue the call are now made (if necessary).

Phase 6: Checking for Non-Business Hours

If a caller calls during non-business hours, you can still provide the caller with some information for calling back during working hours by playing the appropriate recorded message. The following vector, Screen 9 and Screen 10, illustrates one approach in this regard. This vector would be used for a company that was open seven days a week, from 8:00 A.M to 5:00 P.M., including Saturday and Sunday.

```
change vector 20                                    Page 1 of 3
                            CALL VECTOR

 Number: 20                 Name:_______________                  Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n          CINFO? n           BSR? n

 01 goto step 12 if time of day is all 17:00 to all 8:00
 02 goto step 11 if calls queued in split 5 pri l > 10
 03 queue-to split 5 pri l
 04 wait-time 10 secs hearing ringback
 05 announcement 2771
 06 wait-time 10 secs hearing music
 07 check split 7 pri m if calls-queued < 5
 08 wait-time 60 secs hearing music
 09 announcement 2881
 10 goto step 6 if unconditionally
 11 busy
```

Screen 9. Checking for Non-Business Hours (Screen 1 of 2)

```
                                                Page 2 of 3
                            CALL VECTOR

 12 disconnect after announcement 3222
 13 _________________
 14 _________________
 15 _________________
 16 _________________
 17 _________________
 18 _________________
 19 _________________
 20 _________________
 21 _________________
 22 _________________
```

Screen 10. Checking for Non-Business Hours (Screen 2 of 2)

The **goto step** command in step 1 checks if the call arrives during non-business hours. Specifically, if the call arrives between 5:00 P.M. and 8:00 A.M. on any day of the week, the command passes control to step 12. The **disconnect** command in step 12 includes and provides an announcement that first gives the caller the appropriate information and then advises him or her to call back at the appropriate time. Announcement 3222 could contain this message: “We’re sorry. Our office is closed. Please call back any day between 8:00 A.M. and 5:00 P.M.” The command then disconnects the caller.

As an alternative to disconnecting callers who place a call during non-business hours, you can allow callers to leave a message by including the **messaging split** command within the vector. See Chapter 3 for more details.

On the other hand, if the call does not arrive during the hours specified in step 1, control is passed to step 2, and vector processing continues. On step 2, split 5 is checked for calls waiting at priority low and above (that is, for all priorities).

Benefits of Call Vectoring

Coupled with Automatic Call Distribution (ACD), Call Vectoring enables calls to be processed at a faster rate within an intelligent, real-time system. As a result, Call Vectoring provides an appreciable cost saving to the user.

Table 1 summarizes the benefits of Call Vectoring. The last column in Table 1 identifies the vector(s) [via the appropriate screen(s)] in the reference portion of the manual that illustrate(s) these benefits.

Table 1. Benefits of Call Vectoring

Category	Call Vectoring Benefits	Screen/Page
Call Treatment	Implement special treatment based on the time of day and the day of the week (for example, providing night service).	26 on Page 52 33 on Page 70
	Automatically change treatment according to either how long the call has been waiting or to changing traffic or staffing conditions.	24 on Page 50 25 on Page 51 38 on Page 76 39 on Page 79
	Provide appropriate caller feedback during waiting (for example, music or announcements during heavy calling periods).	13 on Page 38 14 on Page 38 15 on Page 38 16 on Page 39 17 on Page 39
	Provide multiple and/or recurring informational or delay announcements that are selected according to the time of day/day of the week, call volume, or staffing conditions.	23 on Page 48 24 on Page 50 33 on Page 70
	Provide 24 hour/day, 7 day/week automated information announcements.	15 on Page 38 16 on Page 39
	Remove selected calls (by providing busy or disconnect).	18 on Page 40 19 on Page 41 20 on Page 43 22 on Page 47 25 on Page 51
	Set up and test, in advance, special call treatments for events such as sales, advertising campaigns, holidays, snow days, and so on.	16 on Page 39 19 on Page 41
	Provide the caller with a menu of choices.	30 on Page 63 31 on Page 65 38 on Page 76
	Execute a VRU script.	20 on Page 43

Continued on next page

Table 1. Benefits of Call Vectoring — *Continued*

Category	Call Vectoring Benefits	Screen/Page
Call Routing	Queue calls to up to three splits simultaneously, consequently improving the average speed of answer and agent productivity.	21 on Page 45 33 on Page 70 39 on Page 79
	Implement routing to local or distant destinations.	24 on Page 50 28 on Page 61 29 on Page 62 34 on Page 71 38 on Page 76 39 on Page 79
	Connect callers to a voice-mail or messaging system either automatically or at their request.	22 on Page 47 23 on Page 48 38 on Page 76
	Reduce call transfers by accurately routing callers to the desired destination.	28 on Page 61 29 on Page 62 38 on Page 76
	Provide up to four ACD queuing priority levels and the ability to change the queuing priority dynamically, as a result, providing faster service for selected callers.	33 on Page 70 38 on Page 76 39 on Page 79
	Reduce agent and/or attendant staffing requirements by: (1) automating some tasks; (2) reducing caller hold time; (3) having agents in one split service multiple call types.	16 on Page 39 17 on Page 39 28 on Page 61 31 on Page 65 32 on Page 66 34 on Page 71 38 on Page 76
Information Collection	Provide customized and/or personalized call treatment via information collection and messaging.	28 on Page 61 30 on Page 63 32 on Page 66 34 on Page 71 38 on Page 76

Call Vectoring Fundamentals

2

The manner in which a call is processed depends upon a number of components within both the switch and the Call Vectoring software. Some of these components include the following:

- Resources available to process a call (for example, agents, splits, software, hardware)
- Vector control flow
- Commands used within the relevant vector(s).

A prudent utilization of these components will produce an effective means of processing telephone calls. This chapter discusses these components, which constitute the fundamentals of Call Vectoring.

Managing Your Calls

When a call is placed to a system with Call Vectoring activated, the call accesses the appropriate vector(s) via a Vector Directory Number (VDN). A VDN is a soft extension number that is not assigned to an equipment location. Each VDN maps to one vector, but several VDNs may map to the same vector. The VDN is discussed later in this chapter.

Once the call goes to a vector, the call's routing and treatment are determined by the commands in the vector. Processing starts at the first step and then proceeds usually sequentially through the vector. Any steps left blank are skipped, and the process automatically stops after the last step in the vector.

Call Vectoring allows the chaining of vector steps and vectors. Accordingly, one vector can direct the call to another vector or VDN, which in turn can direct the call to yet another vector, and so on. Note, however, that a maximum of 1000 vector steps can be executed for any call. When a call enters vector processing, a loop counter keeps track of the number of vector steps executed. If the loop counter exceeds 1000, a *stop* command is executed.

When a call is delivered to an available agent, the agent can see the information associated with the VDN (for example, the VDN name) on his or her display (if present) and, as a result, can respond to the call with knowledge of the service or response required.

In the real world, of course, not every call placed to a site is immediately answered by an agent. The customer usually has fewer agents than the maximum simultaneous call capacity. Therefore, calls will have to be queued.

The following sections discuss how calls are routed and/or queued via Call Vectoring. Subsequent sections discuss agent states, priority levels, caller feedback, and caller control.

Call Flow

Calls enter a vector and execute steps sequentially beginning with step 1, unless there is a *goto* step. Most steps take microseconds to execute. The exception is steps with **announcement**, **wait-time** and **collect digits** commands. A 0.2 second wait occurs after every seven executed steps unless an explicit wait has occurred. Note that **wait-time** with 0 seconds is not an explicit wait.

Call Vectoring uses several call flow methods to redirect and/or queue calls. These methods involve the use of the Call Vectoring commands, which are described later in this chapter. The methods for queuing and redirecting calls follow:

- **Multiple split queuing** allows a call to queue to up to three splits.
- **Intraflow** allows calls unanswered at a split within a predefined time frame to be redirected to one or more other splits on the same switch. If redirection depends upon a condition to be tested, the process is referred to as *conditional intraflow.*
- **Interflow** allows calls directed to a vector to be redirected to an external or non-local split destination. This destination is represented by a number programmed in the relevant vector. Calls can be routed to an attendant (or attendant queue), a local extension, a remote Uniform Dialing Plan (UDP) extension, an external number, or a VDN.

Each of these call control flow methods is discussed in the upcoming chapters.

Caller Control

Call Vectoring allows for the temporary transfer of call management control to the caller via several means, as follows:

- **Caller-Selected Routing**. If Call Prompting is enabled, the vector commands can prompt the caller to input information in the form of dialed digits from a touch-tone telephone or from an internal rotary telephone that is located on the same switch. (A recorded announcement is usually used for prompting purposes.) Once the caller inputs the digits, the call is efficiently and accurately routed to the correct department or destination. This procedure can significantly reduce the number of transferred calls and thus better satisfy the caller's needs.
- **Messaging** is a means of satisfying customer demand during peak calling periods. The caller can leave a voice message in the event that the call cannot be or has not yet been answered. When messaging is enabled, control is eventually passed to the Audio Information Exchange (AUDIX) or message service split. AUDIX is a voice mail adjunct that allows you to record, edit, forward, and retrieve voice messages to and from callers.

Subsequent chapters discuss these procedures in more detail.

Call Queuing to Splits

Basic Call Vectoring is used primarily to control the call activity of ACD splits. Basic Call Vectoring can queue calls to up to three such splits simultaneously at any one of four priority levels. This process is called *multiple split queuing*. The first split to which a call is queued via this process is called the *main split*, while the second split and the third split (if necessary) are called *backup splits*.

Multiple split queuing serves to provide better service to the caller, and it also enables a better utilization of agents. A call remains queued until either vector processing terminates or the call reaches an agent or another destination. (Vector processing termination is discussed later in this chapter.)

When an agent becomes available in any split to which the call is queued, the following events take place:

- The call begins ringing the agent (or connects if the agent is set up for auto-answer).
- The call is removed from any other queues. Announcements, music, ringback, or other audio source are also removed.
- Vector processing terminates.

Note that these actions always happen *immediately*, even if the caller is receiving call treatment (for example, hearing an announcement). (Call treatments are discussed later in this chapter.)

Multiple split queuing is illustrated in Chapter 3, ‘‘Basic Call Vectoring’’.

Split Queue Priority Levels

If a call is queued without Call Vectoring enabled, the call is tracked at one of two priority levels: *Medium* and *High*. On the other hand, if a call is queued via Call Vectoring, the call can be assigned one of four priority levels: *Top*, *High*, *Medium*, and *Low*. Within each priority level, calls are processed sequentially as they arrive. This is equivalent to a FIFO (first-in, first-out) order. A vector can be administered to queue calls at any of the four priority levels.

NOTE:
If a call is already queued to one or more splits that are currently intended to serve as backup splits, the call could be requeued at the new priority level indicated in the command step. For further details on requeuing, see Appendix A.

Agent Work Mode

Call Vectoring can make call management decisions according to real-time agent work modes. These states, *available-agents* and *staffed-agents*, can appear as conditions within the **check split** and **goto** Call Vectoring commands (that is, the commands can check for the number of available agents or staffed agents).

For ACD splits, *staffed-agents* represents the number of agents logged-in. *Available-agents* represents the number of agents logged-in and ready to receive an ACD call.

For non-ACD hunt groups, *staffed-agents* is synonymous with *administered*, since hunt groups do not have any log-in, log-out, or work modes. *Available-agents* is the number of agents ready to receive a hunt group call.

For ACD calls, an agent’s state is further defined by the relevant *work mode*. The following list describes these modes:

- *After-Call-Work Mode* means the is agent unavailable to receive any ACD calls for any split. This mode can be used when the agent is doing ACD call-related work.

- *Auto-In Work Mode* means the agent is available to receive calls. Also, the agent receives a new ACD call immediately after disconnecting from the previous call. When Multiple Call Handling by Request is enabled, an agent in Auto-In Work Mode can receive additional ACD calls while still active on a call.

  NOTE:
  Multiple Call Handling Forced is not supported with the DEFINITY BCS and GuestWorks offers.

- *Auxiliary-Work Mode* means the agent is unavailable to receive any ACD calls for the specified split. This mode can be used when an agent is performing non-ACD activities, such as going on a break.
- *Manual-In Work Mode* means the agent is available to receive calls. This mode automatically puts the agent into the *After Call Work Mode* after disconnecting from an ACD call. When Multiple Call Handling by Request is enabled, an agent in Manual-In Work Mode can receive additional ACD calls while still active on a call.

See the *DEFINITY® ECS Administrator's Guide* for a more complete description of agent work modes and the Multiple Call Handling on Request feature.

Calling Party Feedback

The initial feedback a caller hears as the call is being processed by a vector depends upon the origin classification of the call, which can be one of the following:

- Internal (internal call from another switch user)
- Non-Central Office (CO) [incoming call over a Direct Inward Dialing (DID) or tie trunk over which incoming digits are received]
- CO (incoming call over a CO or automatic-type tie trunk over which no digits are received).

For an internal or a non-CO call, the caller hears silence until one of the following vector steps is reached:

- Wait with system music, ringback, or an alternate audio/music source (caller hears system music, ringing, or the music or audio associated with an administered port)
- Announcement (caller hears the announcement)
- Busy (caller hears a busy tone)
- Call ringing an agent or at a station (caller hears ringing or the agent answering the call).

For a CO call, the caller hears CO ringback until one of the following vector steps is reached:

- Announcement (caller hears the announcement)
- Wait with system music or alternate audio/music source (caller hears system music, or the music or audio associated with an administered port)
- Call answered (caller hears the agent or voice response answering the call).

For a CO call for which answer supervision has already been supplied (via the processing of an announcement or the issuing of a *wait-time* command), the caller may hear any of the following:

- Announcement when any **announcement** command is processed
- Ringback, silence, system music, or an alternate audio/music source when a **wait-time** command is processed
- Busy when a **busy** command is processed
- Ringback when the call is ringing a station.

Regardless of the call's origin, the caller can expect to hear different forms of the feedback described in this section as the relevant vector steps are processed. Examples of how subsequent caller feedback is provided in the vector appear in Chapter 3, ''Basic Call Vectoring'', and in several of the following chapters.

Dialed Number Identification Service (DNIS)

In the traditional ACD arrangement, each agent in a given split is trained to answer calls relevant to one specific purpose in an efficient and professional manner. However, ACD managers have recognized the need to enhance this arrangement in which each split is limited to a single call-answering task.

To this end, there is a split arrangement available in which each group of agents is proficient in dealing with several types of calls. The intent is to service multiple call types with the use of fewer agents overall and with less administrative intervention by the ACD manager. Usual economies of scale come into play here. For example, where five agents might be needed in each of three smaller splits (15 agents total) to handle three types of calls, only 11 or 12 agents might be needed in the combined split.

To aid in providing capabilities such as the one just presented, a network service known as Dialed Number Identification Service (DNIS) is available. DNIS enables a unique multidigit number (of usually four digits) that is based on the dialed number to be associated with the call (sent to a customer's telephone, used to provide different treatments for the call, and so on.). The number that is sent depends upon the telephone number dialed by the caller. Each DNIS number in your telephone system can be programmed to route to an ACD split comprised of agents who are proficient in handling several types of calls.

Call Vectoring takes the DNIS number from the network and interprets this number as a VDN. When the call is delivered to the agent terminal, the unique name assigned to the particular VDN is displayed on the agent's terminal. This allows the agent to know the specific purpose of the call. As a result, the agent can answer with the appropriate greeting and be immediately prepared to service the customer.

Vector Processing

If Call Vectoring is in effect, telephone calls are processed by one or more programmed sequence of command steps called vectors.

The following sections provide a general overview of vector processing. To this end, the following topics are discussed:

- Vector Directory Number (VDN)
- Vector control flow
- Programming capabilities.

Vector Directory Number

Within Call Vectoring, calls access the appropriate vector(s) via a Vector Directory Number (VDN). A VDN is a soft extension number that is not assigned to an equipment location. In effect, the digits dialed by a caller or sent to the switch from an external network are translated within the system as a VDN.

The VDN points to the vector, and it defines the service desired by the caller. The VDN allows for specific call-handling and agent-handling statistical reporting for BCMS.

VDNs are assigned to different vectors for different services that require specific treatments. Any number of VDNs can be assigned to the same vector. As a result, the same sequence of treatments can be given to calls that reach the system via different numbers or different locations.

The VDN has several properties. These properties are administered on the Vector Directory Number administration form, Screen 11.

```
add vdn 2000                                                Page 1 of   1

                          VECTOR DIRECTORY NUMBER

                           Extension: 2000
                                Name:
                  Allow VDN Override?
                               COR:
                                TN:
                        Vector Number:
                           AUDIX Name:
                Messaging Server Name:
                            Measured:
```

Screen 11. Vector Directory Number Administration

- **Extension** — The extension number used to identify the VDN.
- **Name** — The name that is associated with the VDN. This name, which is shown on agents' displays, is optional and can contain up to 15 characters. The name may be truncated on agents' displays depending on the application.
- **Allow VDN Override** — An option that allows the name and other attributes of a subsequently routed to VDN to be used instead of the name and attributes of the current VDN. See ''VDN Override'' on Page 26 for more information.
- **COR (Class of Restriction)** — A 1- or 2-digit number that specifies the COR of the VDN.

  NOTE:
  As a security measure, you can deny incoming callers access to outgoing facility paths by configuring the COR of the VDN to prohibit outgoing access. For details, refer to the *BCS Products Security Handbook*.
- **TN** — Leave at the system default; this field is not supported for DEFINITY BCS and GuestWorks.
- **Vector Number** — An identification number that determines which vector is activated when a call comes into a VDN. Several VDNs may send calls to the same vector.
- **AUDIX Name** — Only displayed for G3r systems. The name of the AUDIX that contains the voice mailbox for this VDN [appears on the Node Names (R7) or Adjunct Names (R6) form].
- **Messaging Server Name** — Only displayed for G3r systems. The name of the messaging server that contains the text mailbox for this VDN [appears in the Node Names (R7) or Adjunct Names (R6) form].
- **Measured** — Enter `None` if the VDN is not measured by BCMS; enter `Internal` if the VDN is being measured by BCMS. The options `Both` and `External` do not apply.

VDNs can be preassigned to incoming (automatic) trunk groups, or they can be sent in digit form to the switch by the public or a private network. The digits sent to the system can come from the serving Central Office (CO) or toll office via the Direct Inward Dialing (DID) feature or DNIS. The digits can also come from another location via dial-repeating tie trunks, or they can be dialed by an internal caller. For a non-ISDN call, the last four digits of the number are sent to the system, while for an ISDN call, the entire ten-digit number is sent.

The last four or five digits of the destination address passed to the switch on a DID/DNIS or on a dial tie-trunk call comprise the VDN. Automatic trunks do not pass destination address digits. Instead, each such trunk always routes to a specific incoming destination that is programmed for the corresponding automatic trunk group. The destination can be an attendant queue, an extension, a hunt group number, or a VDN.

VDN Override

VDN Override allows information about a subsequently routed to VDN (if any) to be used instead of the information about the current VDN. This information includes the following:

- The name of the subsequent VDN
- Messaging split command with the “active” entry
- VDN of Origin Announcement
- VDN Return Destination with the condition that once the call leaves vector processing for the first time, the Return Destination never changes. See Appendix C, ‘‘Considerations for the Vectoring Features’’ for more information.

NOTE:
Throughout this document the “active” VDN is the active called VDN as modified by VDN override rules. The “latest” VDN is the most recent VDN to which the call was routed.

VDN Override can be used in conjunction with a vector that prompts the caller for a particular service. Let’s say, for example, a call is placed to an automobile dealership. This dealership consists of several departments, including “Sales” and “Parts.” Let’s presume the caller wants to talk to someone in “Sales.” In such a case, the call comes into the “Main” vector (whose VDN name is “Main”) and is eventually routed to the “Sales” vector (whose VDN name is “Sales”). If VDN Override is assigned to the “Main” VDN, the “Sales” VDN name appears on the agent’s display when the call is finally connected to the agent. This process is illustrated in Figure 1. In this example, the “Sales” VDN is the active VDN as well as the latest VDN. If VDN override had not been assigned to the “Main” VDN, the agent’s display would have shown “Main.” In this case, “Main” would be the active VDN while “Sales” would be the latest VDN.

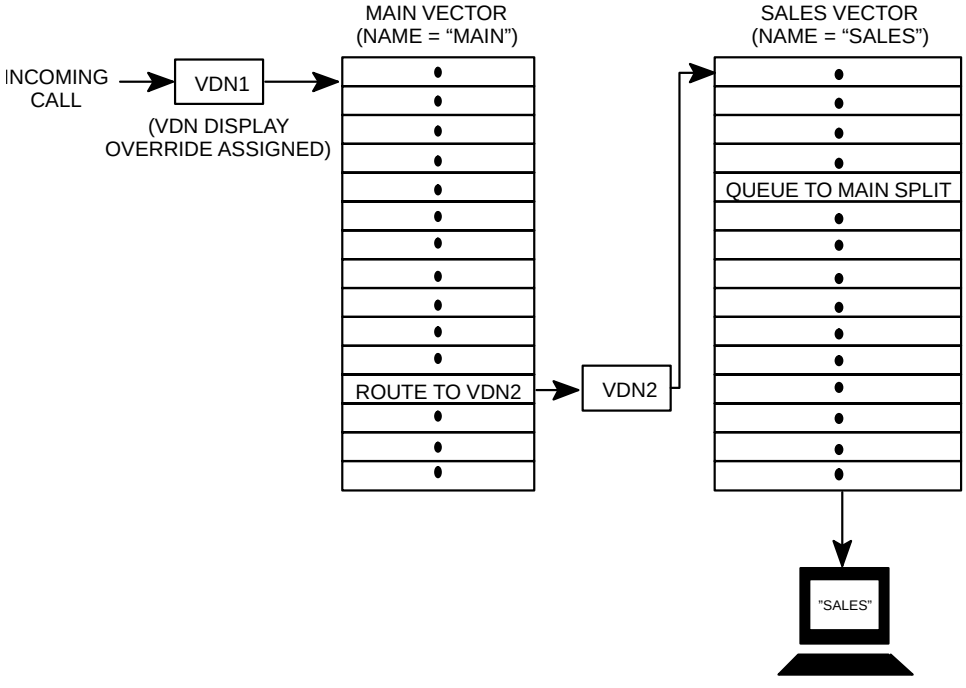

Figure 1. VDN Override Assigned to Originally Called VDN

VDN in a Coverage Path

A VDN can be assigned as the last point in a coverage path. Whenever a VDN is assigned as such, a call goes to coverage and can then be processed by Call Vectoring or Call Prompting (if either is enabled). Accordingly, the Call Coverage treatment for the call is extended (that is, coverage can be sent to an external location, or the type of coverage can be controlled by the caller).

VDN in a coverage path is used for a number of applications, including the following:

- Routing coverage calls off-premises via the **route-to** command
- Serving as a coverage point for specific call operations (for example, sending calls to a secretary during the day and to AUDIX at night).

VDN as a coverage point is illustrated in Chapter 3, ‘‘Basic Call Vectoring’’. For information about VDN in a Coverage Path interactions, see the *DEFINITY® ECS Administrator’s Guide*.

Redirection on No Answer Feature Redirected to a VDN

The Redirection on No Answer feature redirects a ringing ACD call after an administered number of rings. It prevents a call from ringing indefinitely at a terminal when an agent does not answer. When a call is redirected, the system puts the agent into AUX work so that the agent is no longer available to receive ACD calls unless the agent has an active or held ACD call. In the case of Auto-Available Splits, the system logs the agent out when a call is redirected.

A VDN can be administered as the destination of a Redirection on No Answer redirected call. In this way, a call that is not answered can be redirected to a VDN to receive special treatment. Enter the number of the destination VDN for a Redirection on No Answer call in the `Redirect to VDN` field on the Hunt Group form. All calls that are redirected by Redirection on No Answer from that split are sent to the same administered VDN. If no destination VDN is administered, but the number of rings for redirection is entered, the call redirects back to the split.

See the Redirection on No Answer description in the *DEFINITY® ECS Administrator's Guide* for a more details.

Vector Control Flow

Vector Processing starts at the first step in the vector and then proceeds sequentially through the vector unless a **goto** command is encountered. Any steps left blank are skipped, and the process automatically stops after the last step in the vector.

The Call Vectoring ''programming language'' provides three types of ''control flow'' that serve to pass vector-processing control from one vector step to another. Control flow types are described in the following list.

- **Sequential flow** passes vector-processing control from the current vector step to the following step. Most vector commands allow for a sequential flow through the vector.

  NOTE:
  Any vector command that fails automatically passes control to the following step. For success and/or failure criteria for the Call Vectoring commands see ''Criteria for Success/Failure of Call Vectoring Commands''.

- **Unconditional branching** *unconditionally* passes control from the current vector step to either a preceding and/or succeeding vector step or to another vector (for example, *goto step 6 if unconditionally*).
- **Conditional branching** *conditionally* passes control from the current vector step to either a preceding and/or succeeding vector step or to a different vector. This type of branching is based on the testing of threshold conditions (for example, *goto vector 29 if staffed-agents in split 6 < 1*).

Each of these control flow types is fully described in the upcoming chapters.

NOTE:
With one exception, call vectoring has an execution limit of 1000 steps. Once a call enters vector processing, a "loop counter" keeps track of the number of vector steps executed. If the loop counter exceeds 1000, a *stop* command is executed.

NOTE:
An implicit wait of 0.2-seconds is provided after every seven vector steps if vector processing is not suspended during any one of these steps (see the **wait-time** command manual pages in Appendix A, ''Call Vectoring Commands'').

Termination Versus Stopping

For the purposes of this document, the expression *vector processing terminates* means a call has completely left vector processing. This occurs when the call is ringing at an agent's station, is abandoned by the calling party, receives a forced disconnect or a forced busy, or is successfully routed to an extension or to an off-premises number.

It is important to differentiate between vector processing termination and *stopping*, the latter of which is caused by the **stop** command or by the execution of the final step in the vector. Whereas vector processing termination removes the call from the queue if the call is queued, the **stop** command prevents the processing of new vector steps but leaves the call in queue as the calling party receives feedback, such as ringback. If vector processing stops and the call is not queued, the call is dropped.

Vector processing termination and the **stop** command are discussed and illustrated later in this document.

Programming Capabilities

The Call Vectoring commands can perform a number of functions relevant to processing telephone calls. A brief explanation for each of these functions follows.

- **Providing call treatments**. The caller can be provided with a recorded announcement explaining that, at the moment, an agent cannot answer the call for some reason (for example, there are no agents available, the work day is over, and so on). Announcements also provide the caller with instructions and encouragement. Also, audible feedback (silence, ringback, system music, or an alternate audio or music source) or a busy tone can be provided to the caller. Provisions can also be made to delay vector processing for a specific number of seconds before the next vector step is executed. Also, when necessary, the call can be disconnected. Finally, a connection to AUDIX can be initiated.
- **Routing calls**. Calls not immediately answered by an agent can be queued to one or more splits, as explained earlier in this chapter. A caller can also leave a recorded message if he or she chooses to do so. Finally, a call can be routed to a number programmed in the vector or to digits collected from the caller.
- **Branching/programming**. Branches can be made from one vector step to another such step or to another vector. This can be done unconditionally as well as conditionally. Conditional branching is done according to a number of conditions (for example, number of available agents in a split, number of calls in a split queue, the number of the phone the call is made from, and so on). Finally, vector processing can be stopped when necessary.
- **Collecting and acting on information**. Optionally, touch-tone digits can be collected and serve as the basis for further vector processing (for example, a specific agent can be reached via touch-tone digit(s) entered by the caller).
- **Executing VRU scripts**. Voice scripts housed within a Voice Response Unit (VRU) can be executed for the caller. Voice scripts provide the caller with information or instructions, and the caller can often make an appropriate response thereto (by, for example, entering touch-tone digits).

Command Summary

This section lists and describes the commands used by the Call Vectoring features. The list is meant to help familiarize the reader with these commands. The commands are explained further in also in Chapter 3, Chapter 4, and Appendix A.

- **Announcement** provides the caller with a recorded announcement.
- **Busy** gives the caller a busy signal and causes termination of vector processing.
- **Check** conditionally checks the status of a split for possible termination of the call to that resource. The command either connects to an agent in the split or puts the call into its queue (at the specified queuing priority level) if the condition specified as part of the command is met. A call may be queued to up to three different splits simultaneously.
- **Collect Digits** collects up to 16 digits that are either entered by the caller during vector processing or sent by the network. An optional announcement can be played first when the digits are being collected directly from the caller.
- **Converse-on Split** integrates Voice Response Units (VRUs) with the switch. Specifically, the command allows voice response scripts to be executed while the call remains in queue, and it allows the passing of data between the switch and the VRU.
- **Disconnect** ends treatment of a call and removes the call from the switch. The command also allows the optional assignment of an announcement that will play immediately before the disconnect.
- **Goto Step** is a branching step that allows conditional or unconditional movement to a preceding or succeeding step in the vector. Conditional branching is determined by a number of factors (for example, number of calls queued in the split, number of staffed agents in the split, and so on).
- **Goto Vector** is a branching step that allows conditional or unconditional movement to another vector. Conditional branching is determined by a number of factors (for example, number of calls queued in the split, number of staffed agents in the split, and so on).
- **Messaging Split** allows the caller to leave a message for a specified extension or the VDN extension (default).
- **Queue-to** unconditionally queues a call to a split and assigns a queuing priority level to the call in case no agents are available. A call sent with this command either connects to an agent in the split or enters its queue.
- **Route-to Digits** routes the call to the destination specified by a set of digits collected from the caller by the previous *collect digits* step.
- **Route-to Number** routes the call to the destination specified by the administered digit string.

- **Stop** terminates the processing of any subsequent vector steps.
- **Wait-Time** is used to specify whether the caller will hear ringback, system music, silence, or an alternate audio or music source while the call is waiting in queue. The command also delays the processing of the next vector step by the specified delay time that is included in the command's syntax.

NOTE:
Complete operation details for the **route-to** commands are included in Appendix F.

Condition Testing within the Commands

In the previous section, a number of the Call Vectoring commands are implemented according to a tested condition that comprises part of the command. For example, if the condition expressed in the command is true, the command action is executed. If the condition expressed in the command is false, the command action is *not* implemented, and the next vector step is processed.

The following list provides a set of conditions that might comprise the conditional portion of a Call Vectoring command. Refer to Appendix A for the syntax of each condition.

- Number of staffed agents in a split (explained earlier in this chapter)
- Number of available agents in a split (explained earlier in this chapter)
- Number of calls queued at a given priority to a split
- Amount of time that the oldest call has been waiting in a split
- Number of calls active in a VDN
- Digits entered by the caller or received from a VRU adjunct
- Time of day and day of the week that the call is placed.The syntax for this condition can be illustrated as follows: *mon 8:01 to fri 17:00* means "anytime between 8:01 a.m. Monday through 5:00 p.m. Friday," and *all 17:00 to all 8:00* means "between 5:00 p.m. and 8:00 a.m. on any day of the week."

Depending upon the condition, you can use the following comparison operators with the BCS and GuestWorks offers:

- For comparing digits collected, you can only use = (equal to).
- For comparing to agent status (i.e., available agents, login agents, calls in queue), you can use < (less than), > (greater than), and = (equal to).

The chapters on the Call Vectoring features illustrate condition checking in more detail.

Security

There are a number of security issues for Call Vectoring that should be noted.

NOTE:
For more information on security issues, refer to the *BCS Products Security Handbook.*

Limiting Outside Access Using VDN COR Restrictions

A VDN has a Class Of Restriction (COR). Calls processed by the vector carry the permissions and restrictions associated with the COR of the VDN.

For example, if a vector in the switch is written to collect digits, and then to route to the digits dialed, the restrictions on what calls can be placed are determined by the COR of the latest VDN. Also, checks can be made on the digits that are dialed, using *goto _ if digits* vector commands (for example, *goto _ if digits* in table) to disallow routing to undesired destinations. The *collect digits* step can also be limited to collect only the number of digits required (for example, only collecting five digits for internal dialing).

An incoming caller can access Trunk Access Codes, some Feature Access Codes, or most other sets of dialed digits. To deny incoming callers access to outgoing facility paths, the COR of the Vector Directory Number must be configured to disallow outgoing access. This should include: lowering the Facility Restriction Level (FRL) in the COR to the lowest acceptable value (FRL=0 provides the most restricted access to network routing preferences), assigning a Calling Party Restriction of “Toll” or “Outward” denying Facility Test Call capability, and blocking access to specific Cars assigned to outgoing Trunk Groups using the Calling Permissions section of the Class of Restriction Screen.

Review the Classes of Restriction assigned to your VDNs. If they are not restricted, consider assigning restrictions on the VDN and/or using *goto* tests on those digits to prevent callers from exiting the system via the vector.

Voice Response Integration

When a converse step is used to access a VRU application that returns data for a collect-digits step, the opportunity for toll fraud exists when the VRU application fails to return any data. To avoid this type of toll fraud, be certain that one of the following is true:

- If the collected digits are used to route calls internally, be certain that the Class of Restriction (COR) for the Vector Directory Number (VDN) does not allow calls to route externally.
- If it is necessary to use the collected digits to route calls externally, use a password to verify that the collected digits have been passed by the VRU application. For example, in the following vector Screen 12 the VRU application returns a three-digit password followed by the eight-digit external number. The vector routes calls without the correct password to a different vector and routes calls with the correct password to the collected digits.

```
converse-on split 10 pri m passing none and none
collect 3 digits after announcement none
goto vector 23 if digits <> 234
collect 8 digits after announcement none
route-to digits with coverage n
```

Screen 12. Voice Response Integration Security Example

Basic Call Vectoring

3

Basic Call Vectoring allows you to program the type of treatment a telephone call receives. You can program the type of treatment accordingly by using a set of vector commands.

Vector commands can direct calls to various destinations such as adjuncts and splits. The commands can also direct calls to various treatments such as announcements, a forced disconnect, a forced busy, or a delay treatment.

Command Set

Table 2 illustrates the commands used for Basic Call Vectoring.

Table 2. Basic Call Vectoring Command Set

Command Category	Action Taken	Command
TREATMENT	Play an announcement.	**announcement**
	Delay with audible feedback of silence, ringback, system music, or alternate audio/music source.	**wait-time**
	Play a busy tone and stop vector processing.	**busy**
	Disconnect the call.	**disconnect**
	Execute a Voice Response Unit (VRU) script.	**converse-on split**

Continued on next page

Table 2. Basic Call Vectoring Command Set

Command Category	Action Taken	Command
ROUTING	Queue the call to an ACD split.	**queue-to split**
	Queue the call to a backup ACD split.	**check split**
	Leave a message.	**messaging split**
	Route the call to a number programmed in the vector.	**route-to number**
BRANCHING/ PROGRAMMING	Go to a vector step.	**goto step**
	Go to another vector.	**goto vector**
	Stop vector processing.	**stop**

The following sections explain these command categories.

Providing Call Treatments

In this document, the term treatment is used to indicate the type of feedback the caller receives if the caller is not immediately connected to an agent, or if the system is too busy or not in operation. Basic Call Vectoring provides several types of treatment, as follows:

- Announcements
- Delays with audible feedback
- Busy tone
- Disconnect
- Voice response scripts.

The sections that follow explain these treatments.

Announcements

If a caller is not able to connect to an agent immediately, it is logical to provide the caller with a recorded message in order to accomplish one of the following, depending upon the circumstances:

- Encourage the caller to continue to hold the line.
- Provide the caller with information that will satisfy his or her needs, thereby keeping him or her from waiting a long time for service and also allowing him or her to hang up as soon as possible.

Such a recorded message is referred to as an *announcement*, and it is provided via the **announcement** command.

Whenever a call is connected to an announcement, any previous treatment is discontinued, and answer supervision is sent (unless it has already been provided). If, during an announcement, the call is moved from waiting in a split's queue to ringing or connecting to an agent's station, the announcement is disconnected, and the caller hears ringback. When the announcement completes and is disconnected, the caller hears silence until either a vector step with alternate treatment is processed or the call reaches an agent's station.

Announcements can be classified into three groups, as follows:

- Delay announcements
- Forced announcements
- Information announcements.

NOTE:
In these examples, the recorded messages are shown in italics, but are not part of actual vector command.

Delay Announcements

Screen 13 shows an example of a delay announcement.

```
announcement 2556 (''All our agents are busy.
Please hold.'')
```

Screen 13. Delay Announcement

If the caller does as suggested but ends up waiting an appreciable amount of time without receiving further feedback, he or she may tire of waiting and hang up. To keep the caller on the phone at least a little longer, a supplementary delay announcement similar to Screen 14 might be used:

```
announcement 2557 (''Thanks for holding.  All
our agents are still busy.  Please hold.'')
```

Screen 14. Supplementary Delay Announcement

A delay announcement is usually coupled with a delay step, which is provided by the **wait-time** command (discussed later).

The customer should incorporate as many supplementary delay announcements as he or she deems necessary, given the resources available.

Forced Announcements

There are times when the customer may find it advantageous to have the agents not answer calls. Usually, this option is exercised whenever the customer anticipates a barrage of calls concerning an emergency or a service problem of which the customer is already aware. Accordingly, the customer can incorporate an appropriate announcement as the very first step in the vector. Such an announcement is referred to as a *forced announcement* as illustrated in Screen 15.

```
announcement 1050 (''We are aware of the current
situation and are working to rectify the problem.  If your
call is not urgent, please call back later.'')
```

Screen 15. Forced Announcement

Information Announcements

Under certain circumstances, the customer may find it necessary to provide the caller with recorded information that, by its very content, resolves a problem with such finality that the caller feels no need to follow up on his or her call. Such a recorded message is referred to as an *information announcement*, as illustrated in Screen 16.

```
disconnect after announcement 2918 (''Today has
been declared a snow day.  Please report for work tomorrow
at 8 a.m.'')
```

Screen 16. Information Announcement

Note that the **disconnect** command is used with the announcement. After the announcement, the caller is disconnected, since he or she need not stay on the line any longer.

Delays with Audible Feedback

In presenting an example of a delay announcement earlier in this chapter, we mentioned that this type of announcement is usually coupled with a delay step. A delay step is provided by the **wait-time** command, which allows the caller to remain on hold for at least the amount of time indicated in the command.

Let's take another look at our delay announcement. However, this time, let's couple the announcement with a delay step as illustrated in Screen 17.

```
announcement 2556 (''All of our agents are busy.
Please hold.'')
wait-time 20 secs hearing music
```

Screen 17. Delay with Audible Feedback

Here, the caller is allowed to wait at least 20 seconds for the call to be answered by an agent. During this wait period, the caller is provided with system music, which is one type of feedback available via the **wait-time** command.

If the delay step is the final effective step in the vector, the audible feedback continues beyond the specified duration. (A “final effective step” in a vector is either the last vector step or a vector step that is followed by a *stop* step.) Under normal circumstances, the audible feedback continues until the call is either answered or abandoned. However, if the call is not queued when vector processing stops, the call is dropped. Feedback also continues while a call is queued to a converse split, that is, any split routed to by a converse-on split command, and while data is being passed to a Voice Response Unit (VRU). (See the “Voice Response Scripts’’ section later in this chapter.) Finally, feedback also continues during the wait period before the connection of an announcement and/or a tone detector. (Tone detectors are used in conjunction with the Call Prompting feature and are discussed in Chapter 4.)

Busy Tone

A busy tone and subsequent termination of vector processing are produced via the **busy** command. An exception to this occurs on CO trunks where answer supervision has not been sent. Callers on such trunks do not hear the busy tone from the switch. Instead, these callers continue to hear ringback from the CO. The **busy** command eventually times out and drops the call after 45 seconds. With ISDN PRI, busy tone can be provided from the network switch.

You might want to force a busy tone to process a call that arrives at a time when there is a large number of calls queued in the main split, or when the group of agents is out of service or closed.

The vector in Screen 18 illustrates how you can use the **busy** command.

```
1. goto step 6 if calls-queued in split 1 pri h > 30
2. queue-to split 1 pri h
3. announcement 4000
4. wait-time 2 secs hearing music
5. stop
6. busy
```

Screen 18. Providing Busy Tone

In this vector, the **goto step** command in step 1 sends call control to *busy* in step 6 if the conditions in the former command are met. Specifically, if the number of calls queued at a high priority is greater than 30, the **busy** command is accessed.

Disconnect

You can opt to have a call disconnected by incorporating the **disconnect** command. As a courtesy to the caller, an announcement should be given to the caller before he or she is disconnected under any circumstances.

The **disconnect** command itself has a built-in announcement option. We saw an example of the command when we were discussing information announcements earlier in this chapter. Screen 19 shows the example again:

```
disconnect after announcement 2918 (''Today has
been declared a snow day.  Please report for work tomorrow
at 8 p.m.'')
```

Screen 19. Disconnecting a Call

This example presents an ideal use of the **disconnect** command. The caller is given recorded information that, by its very content, resolves a problem so that the caller feels no need to follow up on his or her call.

Voice Response Scripts

Voice Response Integration (VRI) is designed to enhance the integration of a call group, and to integrate Call Vectoring with the capabilities of voice response units (VRUs).

VRI can do the following:

- Execute a VRU script while retaining control of the call in the vector processing.
- Execute a VRU script while the call remains in the split queue and retains its position in the queue.
- Pool VRU ports for multiple applications.
- Use a VRU as a flexible external announcement device.
- Pass data between the switch and a VRU.

The capabilities in the previous list are provided by the **converse-on split** command, which is an enhancement to the Basic Call Vectoring customer option. The **converse-on split** step is specifically designed to integrate a VRU with the switch. VRI allows VRU capabilities to be used while keeping control of the call in the switch. The inclusion of VRUs with vector processing provides the following advantages:

- Access to local and host databases
- Validation of caller information
- Text to speech capabilities
- Speech recognition
- Increased recorded announcement capacity
- Audiotex applications
- Interactive Voice Response (IVR) applications
- Transaction processing applications.

One of the advantages of VRI is that it allows users to make more productive use of queuing time. For example, while the call is waiting in queue, the caller can listen to product information by using an audiotex application or by completing an interactive voice response transaction. In some cases, it may even be possible to resolve the customer’s questions while the call is in queue. This can help reduce the queuing time for all other callers during peak intervals.

During the execution of a VRU script, if the caller previously queued to an ACD split, the caller retains his/her position in queue. If an agent on the switch becomes available to service the call, the line to the VRU is immediately dropped, and the calling party is connected to the available agent.

Screen 20 shows an example of a vector that can access voice response scripts from a VRU.

NOTE:
Recall that one or more VDNs can access the same vector. This capability is appropriate for the following example.

```
VDN (extension=1040   name=‘‘car loans’’     vector=5)
VDN (extension=1041   name=‘‘equity loans’’  vector=5)
Vector 5
     1. goto step 10 if calls-queued in split 1 pri h > 30
     2. queue-to split 1 pri h
     3. announcement 4000
     4. goto step 7 if calls-queued in split 1 pri h < 5
     5. wait-time 0 secs hearing music
     6. converse-on split 11 pri h passing vdn and none
     7. wait-time 20 secs hearing music
     8. announcement 4001
     9. goto step 7 if unconditionally
    10. busy
```

Screen 20. Accessing Voice Response Scripts

For this example, let’s suppose first that a caller would like to hear information concerning car loans. Let’s also assume the call is queued to split 1 (step 2) and that vector processing proceeds to step 6. In such a case, the **converse-on split** command in this step delivers the call to the converse split if there is a queue for the split and the queue is not full, or if a VRU port is available. (Otherwise, vector processing continues at the next vector step.) When the VRU port responds, the step then outpulses VDN 1040 to the VRU via the *passing vdn* parameter included in the command. In turn, the VRU executes the “car loans” voice response script for the caller. Note that it is important to provide a feedback step prior to the converse-on step in case there is a delay in reaching an available converse split port. In this example step 5 provides music for this purpose.

Now, let’s suppose another caller wants information concerning equity loans. In such a case, if everything proceeds according to form, VDN 1041 is outpulsed to the VRU, which in turn executes the “equity loan” voice response script for the caller.

In either case, while interaction with the VRU is taking place, the call remains in the appropriate split's queue (split 1 in this example). If an agent answers the call while the voice response script is being executed, the voice response script is interrupted, the line to the VRU is dropped, and the caller is connected to the available agent. Once a voice response script starts, no further vector steps are executed until the voice response script is completed.

NOTE:
Refer to Appendix G for a detailed explanation of the call flow for converse—VRI calls.

Besides VDN extensions, the **converse-on split** command can outpulse to the VRU calling party extensions, collected (inputted) caller digits (if Call Prompting is enabled), a string of a maximum of six digits or asterisks, a pound sign (#), or nothing. Further details are included in Chapter 4, ''Call Prompting'' and in Appendix A, ''Call Vectoring Commands''.

NOTE:
In Screen 20, the *calls-queued* condition in the second *goto* step (step 4) in effect serves as a checkpoint for determining whether there is enough time for the voice response script (activated by the *converse-on* step) to be executed. Specifically, if five or more calls are queued to split 1, it is considered feasible to execute the voice response script.

Routing Calls

Basic Call Vectoring offers several means of routing telephone calls, as follows:

- Queuing calls to ACD splits
- Leaving recorded messages
- Sending calls to a vector-programmed number (that represents an internal or external destination).

The following sections discuss these routing procedures.

Queuing Calls to ACD Splits

Calls that come into the Call Vectoring system can be queued to a maximum of three ACD splits. Two commands are used to queue calls to splits.

The **queue-to split** command queues a call *unconditionally.* The command sends a call to a split and assigns a queuing priority level to the call in case all agents are busy.

The **check split** command *conditionally* checks the status of a split for possible termination of the call to that split. The command either connects the call to an agent in the split or puts the call into the split's queue (at the specified priority level) if the condition specified as part of the command is met.

Multiple Split Queuing

The term *multiple split queuing* refers to the queuing of a call to more than one split at the same time. The following vector, Screen 21, helps to illustrate this process.

```
1. goto step 4 if calls-queued in split 1 pri l >= 10
2. queue-to split 1 pri t
3. wait-time 12 secs hearing ringback
4. check split 2 pri m if calls-queued < 5
5. check split 3 pri m if calls-queued < 5
6. announcement 3001
7. wait-time 50 secs hearing music
8. goto step 4 if unconditionally
```

Screen 21. Multiple Split Queuing

To avoid completing vector processing without queuing the call to a split, it is always good practice to check a split's queue before queuing to that split. If the queue is full, alternate treatment (such as queuing to an alternate split) should be provided. In this vector, if the main split's queue (which has ten queue slots) is full, the **goto step** command in step 1 skips the main split and goes directly to step 4 to check the backup splits. Although calls are queued in step 2 at a top priority, a low priority is specified in step 1 so that calls in queue at all priority levels are counted. If there are ten or fewer calls in the main split, control is passed to step 2, where the **queue-to split** command queues the call to split 1. Once the call is queued, vector processing continues at the next step.

Step 4 contains a **check split** command. (Recall that in the last paragraph we mention that this step is branched to if the main split queue is holding ten or more calls.) If the call is not answered by the time step 4 is reached, the **check split** in the step attempts to queue the call to a second split. Specifically, the command first determines whether there are fewer than five calls queued to split 2. If so, the command then attempts to connect the call to an agent in the split. If such a connection cannot be made, the command puts the call into the split's queue (at the specified priority level). Vector processing then continues at the next step. On the other hand, if there are five or more calls queued to split 2, the command fails, and vector processing continues at step 5.

Step 5 contains another **check split** command and, accordingly, the process described in the previous paragraph is repeated, with one difference: the queuing attempt is made to split 3 instead of to split 2.

Except for the condition check, the circumstances under which the **check split** command cannot queue a call are identical to those for the **queue-to split** command.

Finally, note that whenever a call is queued to a backup split, the call remains queued to the main split and/or to another backup split (if already queued to either or both of these splits). Once the call is answered in a split to which it is queued, the call is automatically removed from all the other split(s) to which it is also queued.

NOTE:
The **check split**, **queue-to split**, and **converse-on** commands can access *only* those splits that are "vector-controlled." A split is considered "vector-controlled" if *yes* is entered in the `Vector` field of the Hunt Group form.

Leaving Recorded Messages

Basic Call Vectoring allows the caller to leave a message for the customer if the agents at the customer site are not available to take telephone calls. This is done with the help of the **messaging split** command. Let's take a look at the example in Screen 22.

```
 1. goto step 8 if time-of-day is all 16:30 to all 7:30
 2. goto step 10 if calls-queued in split 47 pri l >= 20
 3. queue-to split 47 pri m
 4. wait-time 12 secs hearing ringback
 5. announcement 4001
 6. wait-time 60 secs hearing music
 7. goto step 5 if unconditionally
 8. announcement 4111 (''We're sorry, our office
    is closed.  If you'd like to leave a message, please
    do so after the tone.  Otherwise, please call back
    weekdays between 7:30 a.m. and 4:30 p.m.  Thank you.'')
 9. goto step 11 if unconditionally
10. announcement 4222 ("We're sorry, all of our agents are busy,
    please leave a message after the tone and we will return your
    call.")
11. messaging split 18 for extension 2000
12. disconnect after announcement 4333 (''We're sorry, we are
    unable to take your message at this time.  Please
    call back at your convenience weekdays between
    7:30 a.m. and 4:30 p.m. Thank you.'')
13. busy
```

Screen 22. Leaving Recorded Message

In this vector, the **goto step** command in step 1 checks to see if the office is open, and branches to step 8 if the office is closed. This is done to accommodate calls that are made during non-working hours, when there are no agents available to take telephone calls. Accordingly, step 8 provides the caller with an appropriate announcement and an opportunity to leave a recorded message.

Step 2 checks to see if split 47's queue (which has 20 queue slots) is full, and branches to step 10 if it is. Steps 3 to 7 queue the call to split 47 and then give audible feedback to the caller.

If the caller chooses to leave a message, the **messaging split** command in step 11 is executed. Split 18 in the command is the Audio Information Exchange (AUDIX) split. AUDIX is a voice mail adjunct that allows a customer to record, edit, store, forward, and retrieve voice messages to and/or from callers. Extension 2000 is the mailbox for split 47 (from step 2).

Upon execution of the **messaging split** command, an attempt is made to connect the caller to AUDIX so he or she can leave a recorded message. If the split queue is full, or if the AUDIX link is down, termination to AUDIX is unsuccessful, and vector processing continues at the next vector step, which (as is the case here) usually contains an announcement that provides the caller with the appropriate apology and subsequent directives. If the caller is successfully connected to AUDIX, vector processing terminates, and a message may be left for the specified mailbox (2000, in this case).

Finally, if the supervisor or a group of agents has an Automatic Message Waiting (AMW) lamp for the mailbox used, and if the lamp lights, the relevant party, upon returning, knows a caller has left an AUDIX message.

Option with the VDN as the Coverage Point

Recall from Chapter 2 that the Vector Directory Number (VDN) can be used as the last point in a coverage path. This capability allows the call to first go to coverage and to then be processed by Call Vectoring and/ or Call Prompting. The capability also allows you to assign AUDIX or the Message Server to a vector-controlled hunt group and to therefore enable access to these servers via a **queue-to split** or **check split** command. The result of all this is that call handling flexibility is enhanced.

Screen 23 shows a vector, for which the VDN serves as a final coverage point, that allows the caller to leave a recorded message.

```
VDN 1234 (used in a coverage path)
Vector 1
    1. goto step 7 if time-of-day is mon 8:01 to fri 17:00
    2. goto step 13 if staffed-agents in split 10 < 1
    3. queue-to split 10 pri 1 (AUDIX split)
    4. wait-time 20 secs hearing ringback
    5. announcement 1000 (''Please wait for voice
       mail to take your message.'')
    6. goto step 4 if unconditionally
    7. goto step 2 if staffed-agents in split 20 < 1
    8. queue-to split 20 pri 1 (message server split)
    9. wait-time 12 secs hearing ringback
   10. announcement 1005 (''Please wait for an attendant
       to take your message.'')
   11. wait-time 50 secs hearing music
   12. goto step 10 if unconditionally
   13. disconnect after announcement 1008 (''We cannot
       take a message at this time.  Please call back tomorrow.'')
```

Screen 23. Leaving Recorded Messages (VDN as the coverage point option)

In steps 3 and 8 of the vector, the caller is given the option of leaving a recorded message. However, in accord with our discussion at the beginning of this section, the **queue-to split** command instead of the **messaging split** command is used in each case. The advantage here is that the call is actually *queued* to the AUDIX split or to the message server split. On the other hand, a **messaging split** command does not queue the call to the split; instead (if successful), it simply connects the caller to the split so the caller may leave a message for the specified extension. However, termination to the split may turn out to be unsuccessful due to a factor that cannot be checked by vector processing. (For example, the AUDIX link might be down, or all AUDIX ports might be out of service.)

As a result of the queuing process, a wait-announcement loop can be included after each `queue-to split` step, and the appropriate loop can be executed until the call is actually terminated to either an AUDIX voice port or to an available message service agent. In this vector, steps 4 through 6 comprise the first wait-announcement loop, and steps 10 through 12 comprise the second such loop.

Sending Calls to a Vector-Programmed Number

Earlier in this chapter, we mentioned calls can be queued to a maximum of three splits. Calls can also be routed to a programmed number in the vector via a process known as *interflow.*

Interflow

Interflow is a process that allows calls that are directed or redirected to one split to be redirected to an internal or an external destination. For Basic Call Vectoring, this destination is represented by a number programmed in the vector. The number is always included in the **route-to number** command, and it may represent any of the following destinations:

- Attendant (or attendant queue)
- Local extension
- Remote UDP extension
- External number
- VDN.

The following vectors in Screen 24 illustrate how interflow is used.

```
VDN (extension=1000   name=''Billing Service''  vector=5)
Vector 5:
    1. announcement 3001
    2. goto step 8 if oldest call-wait in split 1 pri l > 120
    3. goto step 8 if calls-queued in split 1 pri l > 10
    4. queue-to split 1 pri t
    5. wait-time 50 secs hearing music
    6. announcement 3002
    7. goto step 5 if unconditionally
    8. route-to number 2020 with cov n if unconditionally

VDN (extension=2020 name=''Message Service''  vector=10)
Vector 10:
    1. messaging split 18 for extension 3000(''We're sorry, all our
       agents are busy. Please leave a message. Thank you.'')
    2. disconnect after announcement 2505 (''We cannot
     take a message at this time. Please call back tomorrow.'')
```

Screen 24. Call Interflow

In the first vector, a branch is made to step 8 from step 2 if the condition in the latter step (*oldest call-wait in split 1 > 120 seconds*) is true. If the condition is false, a branch is made to step 8 from step 3 if the condition in the latter step (*calls-queued in split 1 > 10*) is true. If that condition is also false, the call is queued (step 4), and a wait-announcement loop becomes effective (steps 5 through 7).

If a successful branch to step 8 is made from step 2, the **route-to number** command is executed. The destination number (2020) in this particular command is a VDN. Accordingly, vector processing terminates in the first vector and begins at the first step of the second vector to which the VDN points.

Once processing control is passed to the second vector, the caller is provided with the appropriate announcement (step 1). Thereafter, upon execution of the **messaging split** command in step 2, the system attempts to either queue the call to the message service split or else terminate the call to a message service agent or to an AUDIX voice port. If one of these attempts succeeds, the caller may leave a message. If none of the attempts succeed, the command fails, and vector processing continues at the next vector command (usually an announcement explaining that the necessary connection could not be made).

Service Observing

For a complete description of Service Observing, see the *DEFINITY® ECS R7 Administrator's Guide.*

Branching/Programming

Basic Call Vectoring provides several programming methods that affect the processing flow within the vector. These methods, which are implemented via Call Vectoring commands, include the following:

- Unconditional branching
- Conditional branching
- Stopping vector processing.

The following sections explain these programming methods.

Unconditional Branching

Unconditional branching is a method that always passes control from the current vector step to either a preceding or subsequent vector step or to another vector. This type of branching is enabled via the **goto step** and **goto vector** commands, each with a condition of *unconditionally* assigned.

Unconditional branching is illustrated in the following vector, Screen 25.

```
1. goto step 8 if calls-queued in split 3 pri m > 10
2. queue-to split 3 pri m
3. wait-time 12 secs hearing ringback
4. announcement 3001
5. wait-time 30 secs hearing music
6. announcement 3002
7. goto step 5 if unconditionally
8. busy
```

Screen 25. Unconditional Branching

The unconditional branch statement in step 7 establishes an apparent “endless loop” involving steps 5 through 7. The loop, however, is not endless, since vector processing terminates if an agent answers the call. Vector processing also terminates when the system recognizes the caller has abandoned the call.

Conditional Branching

Conditional branching is a method that *conditionally* passes control from the current vector step to either a preceding or subsequent vector step or to a different vector. This type of branching is enabled via the *goto step* and *goto vector* commands, each with one of the following conditions assigned and tested: *available-agents*, *staffed-agents*, *calls-queued*, *oldest call-waiting*, or *time-of-day.* If the command's condition is not met, control is passed to the step that follows.

Conditional branching is illustrated in the following vector, Screen 26.

```
1. goto vector 10 if time-of-day is all 17:00 to all 8:00
2. goto vector 20 if time-of-day is fri 17:00 to mon 8:00
3. goto step 8 if calls-queued in split 1 pri l > 5
4. queue-to split 1 pri l
5. announcement 4000
6. wait-time 60 secs hearing ringback
7. goto step 5 if unconditionally
8. busy
```

Screen 26. Conditional Branching

In this vector, a conditional branch test statement appears in steps 1, 2 and 3. If the call is placed during non-business hours (between 5:00 p.m. and 8:00 a.m.) on any day of the week, the **goto vector** command in step 1 routes the call to vector 10. However, if the call is placed during business hours, control is passed to step 2, where the **goto vector** command there checks whether the call is placed during the weekend. If this is the case, the call is routed to vector 20. If not, control is passed to step 3, where the **goto step** command checks for the number of calls queued to the main split. If the number of calls is greater than 5, control is passed to **busy** in step 8. If the number of calls is 5 or less, the call is queued (step 4). Thereafter, an announcement-wait cycle (steps 5 through 7) is implemented until an agent answers the call or the call is abandoned.

Stopping Vector Processing

Basic Call Vectoring provides a specific command that stops vector processing. The **stop** command halts the processing of any subsequent vector steps. If a call is not queued when vector processing stops, the call is dropped and tracked as an "abandon" by BCMS. After the **stop** command is processed, any calls that are already queued remain queued, and any wait treatment (silence, ringback, system music, or alternate audio/music source) is continued.

The following vector, Screen 27, illustrates how vector processing is stopped via the **stop** command.

```
1. goto step 6 if calls-queued in split 21 pri m > 10
2. queue-to split 21 pri m
3. announcement 4000
4. wait-time 30 secs hearing ringback
5. stop
6. busy
```

Screen 27. Stopping Vector Processing

If the **stop** command is reached, the queued caller will continue to hear ringback. Also, if the **stop** command in step 5 is executed, step 6 is not executed immediately thereafter. The latter step can be executed only if the **goto** command in step 1 succeeds.

Note that an *implied stop* follows the last step within a vector. In addition, a vector will stop processing whenever 1000 vector steps have been processed.

Vector Chaining

Multiple vectors can be chained together to enhance processing capabilities. In this regard, the following points involving two Basic Call Vectoring commands should be noted:

- **Route-to number**. If this command is used to point to a VDN, the following happens:
  1. Vector processing continues at the first step in the vector assigned to the routed-to VDN.
  2. Call (if queued) is dequeued.
  3. Wait treatment (if any) is disabled.
  4. Processing then continues in the receiving vector at step 1.
- **Goto vector**. If this command is used, the following happens:
  1. Vector processing continues at the first step in the branched-to vector.
  2. Call (if queued) remains in queue.
  3. Wait treatment (if any) is continued.
  4. Processing then continues in the receiving vector at step 1.

Call Prompting

4

Call Prompting provides flexible call handling based on information collected from a calling party. This information comes in the form of dialed digits originating from an internal or external touch-tone telephone, or from an internal rotary telephone that is on the same switch as the vector. In effect, Call Prompting allows for the temporary transfer of call management control to the caller.

With Voice Response Integration (VRI), digits may be returned to the switch by a Voice Response Unit (VRU) script accessed via a **converse-on split** command. Such digits can also be used for call management.

Call Prompting may be used in various applications to achieve a better and more flexible handling of telephone calls.

Command Set

Table 3 illustrates the commands used for Call Prompting.

Table 3. Call Prompting Command Set

Command Category	Action Taken	Command
INFORMATION COLLECTION	Collect information from the calling party, from the public network in an ISDN SETUP message or from a Voice Response Unit (VRU).	**collect digits**
TREATMENT	Play an announcement.	**announcement**
	Delay with audible feedback of silence, ringback, system music, or an alternate audio/music source.	**wait-time**
ROUTING	Leave a message.	**messaging split**
	Route the call to a number programmed in the vector.	**route-to number**
	Route the call to digits supplied by the calling party.	**route-to digits**
BRANCHING/ PROGRAMMING	Go to a vector step.	**goto step**
	Go to another vector.	**goto vector**
	Stop vector processing.	**stop**

Continued on next page

Touch-Tone Collection Requirements

Before the switch can accept the touch-tone digits entered by a caller, the switch must be equipped with a "collection resource." The resource used for collecting and interpreting touch-tone digits can be either the TN744 Touch-Tone Classifier/Tone Detector or the TN2182 Tone-Clock/Tone Detector/Call Classifier. See *DEFINITY® ECS System Description* for more information.

Whether configuring a new or existing system, the required resources should be traffic-engineered based on the busy-hour call volumes and the number of digits being collected.

Outside callers must have a touch-tone phone to enter the digits requested via the **collect digits** command. For callers using rotary dialing, the Call Prompting timeout takes effect, the **collect digits** command times out, and vector processing continues at the next step. As a precaution, always provide a default treatment (for example, **route-to** attendant command, **queue-to split** command) in the vector script unless the script is created exclusively for users of touch-tone telephones.

NOTE:
The Call Prompting inter-digit timeout can be administered for any number of seconds from 4 to 10. This value is administered on the Feature-Related System Parameters form. See *DEFINITY® ECS Administrator's Guide*.

Provisions for users of rotary phones are illustrated in the vector scripts in this chapter.

Call Prompting Digit Entry

The touch-tone digits entered by a Call Prompting user are collected via the **collect digits** command. This command allows the system to collect up to 24 digits from a touch-tone phone. Sixteen of these digits may be collected immediately, while any remaining digits are stored as dial-ahead digits. (Dial-ahead digits provide the caller with a means of bypassing unwanted announcement prompts on the way to acquiring the information or servicing he or she desires.)

Call Prompting allows some flexibility in entering digits. Specifically, the caller can do the following:

- Remove incorrect digits strings
- Enter variable-length digit strings
- Enter dial-ahead digits.

The following sections explain these processes.

Removing Incorrect Digit Strings

You can (and probably should) include an announcement that requests the caller to enter digits. As an option, the announcement can instruct the caller to enter an asterisk (∗) if he or she enters incorrect data. When the caller enters an "∗", the following happens:

1. Digits collected for the current **collect digits** command are deleted.

   NOTE:
   Also deleted are any dial-ahead digits that are entered and that do not exceed the maximum digit count of 24. (Dial-ahead digits are explained later in this chapter.)

2. Digit collection is restarted.
3. Announcement is not replayed.

Once the caller enters "∗", the caller can re-enter digits for processing.

Entering Variable-Length Digit Strings

The maximum number of digits requested from the caller must be specified in the administration of the **collect digits** command. In some cases, the caller might be permitted to enter fewer digits than the maximum specified. In fact, the number of digits entered by the caller can vary for several variations of one **collect digits** command. Each such grouping of digits is called a *variable-length digit string*.

Call Prompting allows for variable-length digit strings by providing an end-of-dialing indicator in the form of the pound sign (#). "#" is used to end any digit string entered by the caller, and it does the following:

- Tells the system that the caller has finished entering digits
- Causes the next vector step to be processed immediately

Whenever the caller is permitted to enter a variable-length digit string, the announcement portion of the **collect digits** command should specify the largest possible number of digits that can be entered. Accordingly, you should administer each **collect digits** command to collect no more than the intended maximum number of digits. You can have the caller enter "#" as part of a variable digit string entry either at the end of each variable digit string entered or at the end of each such string that, not counting "#," contains *fewer* characters than the maximum number of allowable digits. In the first case, "#" should be included in the count of the number of maximum digits that can be entered; in the second case, "#" should *not* be included in this count.

If the caller enters more digits than the maximum number specified, the additional digits are saved as "dial-ahead" digits for subsequent **collect digits** commands. (Dial-ahead digits provide the caller with a means of bypassing unwanted announcement prompts on the way to acquiring the information or servicing he or she desires.) If the vector, or vectors chained to it, do not contain another **collect digits** command, the extra digits are discarded.

If the caller enters fewer digits than the maximum number specified *and* does not complete the entry with "#," a Call Prompting timeout occurs. The timeout terminates the command, and any digits collected prior to the timeout are available for subsequent vector processing.

A common application involving the entering of variable-length digit strings allows the user to dial either the number for the attendant or an extension (to reach the desired destination.) Let's say the maximum number of digits that can be entered is three. In such a case, if the user wishes to reach the attendant, the user should dial "0#." However, if the user chooses to dial a three-digit extension, the user should dial, for example, "748" and not "748#." Since the maximum number of digits that can be dialed in this case is three, dialing "748#" would cause "#" to be saved as a dial-ahead digit (explained later in this chapter). On the other hand, if the caller dials "748#," and if the maximum number of digits that can be entered is four, "#" is not saved as a dial-ahead digit since it is the fourth of four digits that can be entered in this case.

Entering Dial-Ahead Digits

When digit collection for the current **collect digits** command completes, vector processing continues at the next vector step. However, the switch continues to collect any digits that the caller subsequently dials until the tone detector disconnects. See ''Collect Digits'' on Page 101 for more information. These "dialed-ahead" digits are saved for processing by subsequent **collect digits** commands. Dial-Ahead Digits are explained on Page 65.

Functions and Examples

Call Prompting uses some of the functions found in Basic Call Vectoring. This becomes evident when you compare the command set table for Basic Call Vectoring in Chapter 3, Table 2 with Table 3, Call Prompting, found at the beginning of this chapter.

Call Prompting also provides some additional functions that involve digit processing. These functions include the following:

- Treating digits as a destination
- Using digits to collect branching information
- Using digits to select options
- Displaying digits on the agent's set
- Passing digits to an adjunct.

These functions are illustrated in the following sections.

Treating Digits as a Destination

Call Prompting allows you to route calls according to the digits collected from the caller. Once the digits are collected via the **collect digits** command, the **route-to digits** command attempts to route the call to the destination that the digits represent. The command always routes the call to the destination that is indicated by the digits processed by the most recent collect digits command.

The digits can represent any of the following destinations:

- Internal (local) extension (for example, split/hunt group, station, announcement, and so on)
- VDN extension
- Attendant
- Remote access extension
- External number, such as a trunk access code (TAC) or an Automatic Alternate Route/Automatic Route Selection (AAR/ARS) feature access code (FAC) followed by a public network number (for example, 7-digit ETN, 10-digit DDD, and so on).

Let's take a look at the vector Screen 28 that illustrates how a call is routed via digits that are collected from a caller.

```
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 300
   ("You have reached Redux Electric in Glenrock.
   Please dial a 5-digit extension or wait for the
   attendant.'')
3. route-to digits with coverage y
4. route-to number 0 with cov n if unconditionally
5. stop
```

Screen 28. Treating Digits as a Destination

In this vector, the caller is prompted to enter the destination extension of the party he or she would like to reach (step 2). (The extension in this vector may contain up to five digits.) The vector collects the digits, then routes to the destination via the **route to digits** command in step 3.

If the **route-to digits** command fails (because the caller fails to enter any digits, or because the extension number entered is invalid), the **route-to number** command in step 4 routes the call to the attendant (default). However, as long as the destination is a valid extension, the **route-to digits** command succeeds, coverage applies, and vector processing terminates. (Even if the destination is busy, vector processing terminates because coverage call processing takes effect.)

NOTE:
From time to time, all of the system's tone detectors might be in use. As a result, when you are collecting digits from a caller, you should avoid starting your main vector with a **collect digits** command, since the caller in this case receives no audible feedback if he or she has to wait for a tone detector to become available. Accordingly, it is a good practice to include some treatment (for example, *wait-time 0 secs hearing ringback*) before the initial *collect digits* step.

In addition, if calls are likely to be transferred to this vector, a wait-time step of sufficient length is recommended before the collect step to allow the transferring party enough time to complete the transfer.

Using Digits to Collect Branching Information

Call Prompting allows you to direct a call to another step or vector based on the digits entered by the caller. This branching is accomplished with a *goto* step. For example, in vector Screen 29 digits are used to route calls to different vectors based on an assigned customer number.

```
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 200
   (“Please enter your customer number)
3. goto vector 8 if digits = 10111
4. goto vector 9 if digits = 11111
5. goto vector 10 if digits = 12111
6. route-to number 0 with cov n if unconditionally
7. stop
```

Screen 29. Using Digits to Collect Branching Information

With BCS and GuestWorks, if you are comparing digits collected, you can only use “=” (equal). So, callers entering the digits 10111 are routed to vector 8, callers entering the digits 11111 are routed to vector 9, and callers entering the digits 12111 are routed to vector 10.

Using Digits to Select Options

Call Prompting allows you to provide a menu of options that the caller can use to satisfy his or her information needs. The caller selects the desired option by entering the appropriate requested digit. Once the digit is entered, a conditional branch to the appropriate treatment is made. The treatment is usually provided via the **route-to number** command.

The following vector Screen 30 illustrates how digits are used to select options.

```
1.  wait-time 0 secs hearing ringback
2.  collect 1 digits after announcement 3531
    (''Thank you for calling Bug Out Exterminators.  If you
    wish to learn about the services we provide, please
    dial 1.  If you'd like to set up an appointment for
    one of our representatives to visit your home or
    place of business, please dial 2.'')
3.  route-to number 4101 with cov y if digit = 1
4.  route-to number 4102 with cov y if digit = 2
5.  route-to number 0 with cov n if unconditionally
6.  disconnect after announcement none
```

Screen 30. Using Digits to Select Options

In step 2 of this vector, the user is asked to enter either 1 or 2, depending upon the service he or she desires. If one of these digits is entered, the appropriate one of the next two steps (3 through 4) routes the call to the relevant extension (that is, either 4101 or 4102). If one of the digits is not entered, the call is routed to the attendant (step 5).

Displaying Digits on the Agent's Set

You may include the CALLR-INFO button at the agents' display stations to help process calls that are serviced by the Call Prompting feature. However, if the agent has a 2-line display set, such as an 6424 or Callmaster® set, and the display is in normal or inspect mode, the collected digits are automatically displayed on the second line. These digits remain on this line until they are overwritten, even after the call is released by the agent. On the other hand, for other display sets, the agent must press the CALLR-INFO button to display the collected digits.

You might find it beneficial to install the CALLR-INFO button if you want to expedite calls by reducing the amount of time agents spend on the telephone. For example, the button could be set up to collect specific information (such as a customer account number) before the call is answered by the agent, thus eliminating the need for the agent to ask for this information.

The CALLR-INFO button displays information in the following format:

x=Info: 1234567890

where

- *x* is a call appearance letter (for example, *a*, *b*, *c*, and so on)
- *1234567890* represents the digits collected from the caller.

The digits entered by the caller are collected by the most recent collect digits command. Any digits that were “dialed ahead” and not explicitly requested by the most recently executed **collect digits** command are not displayed.

Let’s assume that digits have been collected via Call Prompting. If the agent presses the CALLR-INFO button when the call is ringing at the agent’s station or when the station is active on a call appearance, the following events occur:

- Ten second timer for display interval is set.
- Status lamp (if available) associated with the button is lit.
- Display is updated. Specifically, the incoming call identification (calling party ICI) is replaced with the collected digits in the format presented earlier in this section. Only those digits collected for the last **collect digits** command are displayed.

If all the conditions to use the button (except for the collection of digits) are set, and the agent presses the button, the status lamp (if available) associated with the button flashes denial.

One or more events may occur during a successful execution after the button is pushed. These events include the following:

- Ten second timer times out.
- Incoming call arrives (at any call appearance).
- Active call changes status (for example, another caller is added to the conference).

If any of these events occur, the following takes place:

- Status lamp (if available) associated with the button is turned off.
- Display is updated (as previously described).

NOTE:
If the agent needs to display the collected digits again, the CALLR-INFO button can be depressed again to repeat the operation described in this section (provided the agent is active on the call or the call is still ringing). Also, the agent can flip between the collected digits and the ICI by alternately pressing the CALLR-INFO and NORMAL buttons.

Dial-Ahead Digits

Dial-ahead digits provide the caller with a means of bypassing unwanted announcement prompts on the way to acquiring the information or servicing he or she desires. These digits are available for use only by subsequent *collect digits* commands. The digits are never used by other vector commands that operate on digits (for example, *route-to digits*, *goto...if digits*, and so on) until they are collected. These digits are not forwarded with interflowed calls. In addition, these digits are not displayed as part of the CALLR-INFO button operation until they are collected by a **collect digits** command.

The vectors shown in Screen 31 and Screen 32 illustrate a situation where a caller can enter dial-ahead digits. Note that, in this case, we are requiring the caller to have a touch-tone telephone. Typically an alternative handling sequence should be programmed in case the caller does not dial a touch-tone digit before the timeout period.

```
VDN (extension=1030   name=''Coastal''   vector=10)
Vector 10:
      1. wait-time 0 secs hearing ringback
      2. collect 1 digits after announcement 3000
         (''Thank you for calling Coastal League Baseball Hotline.
         You must have a touch-tone telephone to use this service.
         If you wish to hear the scores of yesterday's games,
         please press 1.  If you wish to hear today's schedule
         of games, please press 2.'')
      3. route-to number 1031 with cov y if digit = 1
      4. route to number 1032 with cov y if digit = 2
      5. announcement 301 (''Entry not understood.  Please
         try again.'')
      6. goto step 2 if unconditionally
VDN (extension=1031   name=''Scores''   vector=11)
Vector 11:
      1. collect 1 digits after announcement 4000
         (''If you wish to hear scores of games in both divisions,
         please press 3.  If you wish to hear scores for Northern
         Division games only, please press 4.  If you wish to hear
         scores for Southern Division games only, please press 5.'')
      2. goto step 7 if digits = 3
      3. goto step 7 if digits = 4
      4. goto step 9 if digits = 5
      5. announcement 301 (''Entry not understood.  Please
         try again.'')
      6. goto step 1 if unconditionally
      7. announcement 4002 (Northern Division scores)
      8. goto step 10 if digits = 4
      9. announcement 4003 (Southern Division scores)
     10. collect 1 digits after announcement 4004
         (''If you wish to return to the main menu,
         please press 9.  Otherwise, press 0.)
     11. route-to number 1030 with cov n if digit = 9
     12. goto step 15 if digit = 0
     13. announcement 301 (''Entry not understood.  Please
         try again.'')
     14. goto step 10 if unconditionally
     15. disconnect after announcement none
```

Screen 31. Dial-Ahead Digits

```
VDN (extension=1032   name=Schedule   vector=12)
Vector 12
      1. collect 1 digits after announcement 5000
         (''If you wish to hear today's schedule of games in
         both divisions, please press 6.  If you wish to hear
         today's schedule of games in the Northern Division
         only, please press 7.  If you wish to hear today's
         schedule of games in the Southern Division only,
         please press 8.'')
      2. goto step 7 if digits = 6
      3. goto step 7 if digits = 7
      4. goto step 9 if digits = 8
      5. announcement 301 (''Entry not understood.  Please
         try again.'')
      6. goto step 1 if unconditionally
      7. announcement 5002 (Northern Division schedule)
      8. goto step 10 if digits = 7
      9. announcement 5003 (Southern Division schedule)
     10. collect 1 digits after announcement 4004
         (''If you wish to return to the main menu,
         please press 9.  Otherwise, press 0.)
     11. route-to number 1030 with cov n if digit = 9
     12. goto step 15 if digits = 0
     13. announcement 301 (''Entry not understood.  Please
         try again.'')
     14. goto step 10 if unconditionally
     15. disconnect after announcement none
```

Screen 32. Dial-Ahead Digits

Step 2 in the first vector Screen 31 gives the caller two options, each of which provides different information. The caller is prompted to enter either 1 or 2, depending on what information he or she wishes to hear. Once the caller enters a digit, the digit is collected by the **collect digits** command. Thereafter, an attempt is made by the **route-to number** command to route the call to the appropriate vector (step 3 or 4). If the caller enters a digit other than *1* or *2*, the appropriate announcement is provided (step 5), and the digit entry cycle is repeated (step 6).

Let's suppose that the caller, when prompted, enters *1*. In such a case, the second vector is accessed.

In step 1 of this vector, the caller is given three options that supplement the original option provided in the first vector. The caller is prompted to enter either 3, 4, or 5, depending on what information he or she wishes to hear. If the caller enters an incorrect digit, the customary digit correction routine is implemented (steps 5 and 6). Once an appropriate digit is entered, the call is routed—this time via use of a **goto step** command (step 2, 3, or 4)—to the appropriate announcement (step 7 or step 9).

In step 10 of the second vector, the caller is once again prompted. Specifically, the caller is given the choice of returning to the main menu provided in the first vector or of terminating the phone call. If the caller selects the former option (by entering 9), the call is routed to the first vector, and the entire process is repeated.

Note the third vector Screen 32 is similar in design to the second vector. The major difference is the information provided and the requested digit entries.

In our example, we have just seen that the caller has to go through at least two sets of options to get the information he or she wants. Each option set is introduced by an announcement. However, because of the “dial-ahead” digit capability, the caller can bypass the announcements if he or she so chooses. Thus, in our example, the caller could enter 1 and 5 within a matter of seconds to hear yesterday’s Southern Division scores.

The caller may enter digits while he or she is being queued for an announcement or while the announcement is playing. If digits are entered during an announcement, the announcement is disconnected. If digits are entered while a call is queued for an announcement, the call is removed from the announcement queue.

Collection of dial-ahead digits continues until one of the following occurs:

- Vector processing stops or is terminated.
- Sum of the digits collected for the current **collect digits** command plus the dial-ahead digits exceeds the switch storage limit of 24. Any additional digits are discarded until storage is freed up by a subsequent **collect digits** command.

  NOTE:
  Any pound sign (#) digits dialed ahead or entered after the pound sign digit count toward the 24-digit limit.

- The tone detector required by the user to collect digits has been disconnected. This happens whenever one of the following conditions is true:
  - — Successful or unsuccessful *route-to number* step is encountered during vector processing, except where the number routed to is a VDN extension.
  - — Successful or unsuccessful *route-to digits* step is encountered during vector processing, except where the number routed to is a VDN extension.
  - — Successful or unsuccessful *converse-on* step is encountered during vector processing.

— Call Prompting timeout occurs, during which time the caller has not dialed any additional digits [pound signs (#)].

— Vector processing stops or is terminated.

— A successful or unsuccessful *collect ced/cdpd* step is encountered.

NOTE:
When the tone detector is disconnected due to a *route-to number, route-to digits,* or *converse-on* step, all dial-ahead digits will be discarded. This means that following a failed *route-to* or *converse* step, a subsequent *collect digits* step always requires the user to enter digits.

The caller who enters dial-ahead digits no doubt knows which digits to enter ahead of time due to his or her familiarity with the service provided. Once the caller masters the digit sequence relevant to a particular service, the dial-ahead digit capability saves time and also eliminates much of the redundancy associated with automatic telephone servicing.

Call Vectoring Applications

5

This chapter is intended to present a few generic Call Vectoring applications a customer might use. Each application is based on one or more of the Call Vectoring features discussed in this document. Vector exercises are provided at the end of the chapter.

Table 4 identifies the feature(s) used within each example in this chapter. The examples are numbered according to the order in which they appear within the chapter. The name of the section in which each example appears is listed first.

Table 4. Applications and Corresponding Feature(s)

Section Title	Example No.	Feature(s) Used
Customer Service Center	1	Basic Call Vectoring
Automated Attendant	2	Call Prompting
Dial by Name	3	Basic Call Vectoring, Call Prompting
DIVA and Data/Message Collection	4	Call Prompting, Basic Call Vectoring

Continued on next page

Customer Service Center

Example 1 presents a scenario where a customer service center is open weekdays from 8 a.m. until 5 p.m. The center provides two separate telephone numbers. One number is for ordinary customers using low priority, while the other number is for high-priority customers. The following three vectors in Screen 33 illustrate how calls to the customer service center are handled.

```
VDN (extension=1021  name=''Customer Serv''  vector=1)
Vector 1:
      1. goto vector 9 if time-of-day is all 17:00 to all 08:00
      2. goto vector 9 if time-of-day is fri 17:00 to mon 08:00
      3. goto step 10 if calls-queued in split 1 pri l > 10
      4. queue-to split 1 pri m
      5. wait-time 10 secs hearing ringback
      6. announcement 3521
      7. wait-time 50 secs hearing music
      8. announcement 3522
      9. goto step 7 if unconditionally
     10. busy
VDN (extension=1022  name=''Priority Cust''  vector=2)
Vector 2:
      1. goto vector 9 if time-of-day is all 17:00 to all 08:00
      2. goto vector 9 if time-of-day is fri 17:00 to mon 08:00
      3. goto step 12 if calls-queued in split 1 pri h > 10
      4. queue-to split 1 pri h
      5. announcement 3521
      6. wait-time 10 secs hearing music
      7. check split 2 pri h if oldest-call-wait < 20
      8. check split 3 pri h if oldest-call-wait < 20
      9. announcement 3522
     10. wait-time 60 secs hearing music
     11. goto step 7 if unconditionally
     12. route-to number 0 with cov n if unconditionally
No VDN
Vector 9:
      1. announcement extension 3529
      2. wait-time 10 secs hearing silence
      3. disconnect after announcement 3529
```

Screen 33. Example 1: Customer Service Center

First, let's assume that a priority customer places a call. In such a case, if the correct number is dialed, vector 2 is accessed. The first two steps of this vector determine if the call arrives during nonbusiness hours. If the call arrives between 5:00 p.m. and 8:00 a.m. on any given day, step 1 routes the call to Vector 9. Step 2 does the same if the call arrives during the weekend (that is, between 5:00 p.m. Friday and 8:00 a.m. Monday). If vector 9 is accessed, the caller is given the appropriate announcement twice (steps 1 and 3) and is then disconnected (step 3).

If the call is placed during business hours, step 3 of vector 2 determines if the number of calls queued in the main split exceeds ten. If so, control is sent to step 12, which routes the call to the attendant. If not, the call is queued to the main split (step 4). Thereafter, if necessary, the appropriate announcement is provided (step 5), followed by a wait period (step 6).

If the call is not answered by this time, steps 7 and 8 attempt to queue the call to a backup split (*2* and *3*, respectively). The call is queued to either split if the oldest call therein has been waiting fewer than 20 seconds. Whether or not the call is queued, steps 9 through 11 implement an announcement-wait cycle that continues until an agent answers the call, or until the caller abandons the call.

A call placed by a nonpriority customer is processed by vector 1. Vector 1 provides a treatment similar to that provided by vector 2. The three differences are as follows:

- The nonpriority customer's call is not given the chance to be queued to more than one split.
- The priority customer's call is given a higher priority in the queue.
- The priority customer's call routes to an operator when there are too many calls queued whereas the nonpriority customer routes to a busy signal.

Automated Attendant

Example 2, Screen 34, illustrates Automated Attendant, which is one of the applications that can be supported by the Call Prompting feature. Automated Attendant allows the caller to enter the extension of the party the caller would like to reach. Depending upon the parameters established, the user can enter up to 16 digits from a touch-tone phone.

Automated Attendant is usually used for customers without DID trunks whose callers know the extension of the people they are calling. Because it reduces the need for "live attendants," Automated Attendant allows the customer to reduce costs.

Screen 34 shows an example of a vector that implements Automated Attendant:

```
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 30001
   (''You have reached Ridel Publications in Greenbrook.
   Please dial a 5-digit extension or wait for the
   attendant.'')
3. route-to digits with coverage y
4. route-to number 0 with cov n if unconditionally
5. stop
```

Screen 34. Example 2: Automated Attendant

Step 1 of this vector contains the **wait-time** command, which is placed before the **collect digits** command in step 2 to provide the caller with ringback in the event that a tone detector is not immediately available. (Recall that a tone detector must be connected for the **collect digits** command to take effect.) Once a tone detector is connected, the caller is prompted to enter the destination extension of the party he or she would like to reach (step 2). The **collect digits** command in step 2 collects the digits. Thereafter, the **route-to digits** command in step 3 attempts to route the call to the destination.

If the **route-to digits** command fails (because the caller fails to enter any digits, or because the digits entered do not comprise a valid extension), the **route-to number** command in step 4 routes the call to the attendant. However, as long as the destination is a valid extension, the **route-to digits** command succeeds, coverage applies, and vector processing terminates. (Even if the destination is busy, vector processing terminates because coverage call processing takes effect.)

Dial-by-Name Feature

The Dial-by-Name feature allows you to "dial" someone by entering their name from your touch-tone keypad. This feature is accessible by using the Call Vectoring feature and the integrated announcement circuit pack (TN750C) to create an "auto-attendant" procedure where one of the options allows callers to enter a person's name instead of their extension number. The system processes the name characters received, and, when a match is found, the number is dialed automatically.

A typical scenario might be as follows:

- When a call comes in to the system (usually to a Listed Directory Number), a vector routes the call to an announcement that says, "Hello. You have reached A1 Hotel. Please press 1 for the operator, press 2 to reach a guest or employee, or press 3 for the reservation desk."
- When the caller selects 2, the caller then hears, "If you know the person's extension, press 1. If you know their name, press 2."
- If the caller does not know the person's extension number, the caller can opt to enter the person's name instead. The caller would press 2.
- The caller is then instructed to enter the person's name.
- As soon as a single match is found, the call is placed to that person.

You can assign several vectors that define how calls will be handled as users select the different prompts. Screen 35, Screen 36, and Screen 37 show an "auto-attendant" procedure that can be used to access the Dial-by-Name feature. Step numbers 1-20 contain the basic auto-attendant steps, and steps 21-32 contain the Dial-by-Name steps.

```
change vector 2                                          Page 1 of 3
                          CALL VECTOR

 Number: 2                Name:Dial by Name                      Lock? n

    Basic? y  EAS? n   G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
Prompting? y  LAI? n  G3V4 Adv Route: n          CINFO? n           BSR? n

01 wait-time    2   secs hearing ringback
02 collect      1  digits after announcement 381
03
04 route-to     number 0               with cov n if digit             =  0
05 route-to     number 105             with cov n if digit             =  1
06 goto         step   12  if digits           =      2
07 goto         step   21  if digits           =      3
08 goto         step   19  if digits           =      4
09 goto         step   16  if digits           =      5
10 route-to     number 0               with cov n if unconditionally
11
```

Screen 35. Example 3: Dial by Name Screen 1

```
change vector 2                                          Page 2 of 3
                          CALL VECTOR

12 collect      3  digits after announcement 382
13 route-to     digits with coverage y
14 route-to     number 0               with cov n if unconditionally
15
16 goto         step   2   if unconditionally
17
18
19 collect      3  digits after announcement 383
20 goto         step   13  if unconditionally
21 collect      4 digits after announcement 661
22 route-to     name1 with coverage y
```

Screen 36. Example 3: Dial by Name Screen 2

```
change vector 2                                          Page 3 of 3
                          CALL VECTOR

23 goto         step 30 if nomatch
24 collect      11 digits after announcement 662
25 route-to     name2 with coverage y
26 goto         step 30 if nomatch
27 collect      2 digits after announcement 663
28 route-to     name3 with coverage y
29 goto         step 30 if nomatch
30 collect      1 digits after announcement 660
31 goto         step 21 if digits = 1
32 route-to     number 0               with cov n if unconditionally
```

Screen 37. Example 3: Dial by Name Screen 3

The procedure above does the following:

1. When someone calls the system, the person receives ringback for 2 seconds.
2. Announcement 381 plays. This announcement asks them to do one of the following:
   - Press 0 or wait if they want the operator; if they press 0 or wait for the timeout, they are routed to the operator.
   - Press 1 if they want the help desk; if they press 1, they are routed to extension 105, which is the help desk.
   - Press 2 if they know the person's extension; if they press 2, they are routed to announcement 382, which tells them to dial the person's extension.
   - Press 3 if they know the person's name; if they press 3, the following sub-procedure occurs:
     a. Announcement 661 plays requesting they enter the first four characters of the person's last name.
        — If there is a match, the call is redirected.
        — If there are multiple matches, continue with Step b.
        — If there is no match, go to Step d.
     b. Announcement 662 plays requesting they enter the rest of the person's last name, followed by the # key.
        — If there is a match, the call is redirected.
        — If there are multiple matches, continue with Step c.
        — If there is no match, go to Step d.
     c. Announcement 663 plays requesting they enter the first two characters of the person's first name.
        — If there is a match, the call is redirected.
        — If there is no match, continue with Step d.
     d. Since there are still no matches, announcement 660 plays telling them they can press 1 to try again, or press 0 to get an operator.
   - Press 4 if they know the department they wish to access (such as engineering); if they press 4, they are routed to announcement 383, which gives them a listing of several departments that they can dial directly.
   - Press 5 to start over again; if they press 5, the caller hears announcement 381, which repeats all of the options.
   - If the caller dials anything else, the call is routed to the operator.

Data In/Voice Answer (DIVA) and Data/Message Collection

Example 4 involves a mutual fund company that is open 24 hours a day, 7 days a week. All incoming calls are directed to a single VDN extension that maps to a main vector. The main vector presents a menu of options to the calling party, and it uses Call Prompting to determine the desired service. Three services are offered, and they are identified and described as follows:

- *New accounts* enables the customer to open a new account.
- *Account inquiries* enables the customer to make inquiries concerning his or her account.
- *Net asset values* enables the customer to hear information concerning the net asset values of the company's funds.

If the caller selects "account inquiries," he or she is prompted to input his or her account number before being answered by an agent. The agent can display this number via use of the CALLR-INFO button, if the button is available and needed.

NOTE:
If the agent has a two-line display telephone supported by the system, the account number is automatically displayed on the second line. Some supported display telephones include 6416, 6424, 8410, 8434 and *Callmaster* set.

This example uses three other applications that can be supported by the Call Prompting feature. These applications are described as follows:

- **Data In/Voice Answer (DIVA)** allows a caller to receive information on a topic selected at the prompt. The caller selects the desired topic by entering the appropriate digit(s).
- **Data Collection** provides a method of collecting digits from a caller. The digits requested comprise an official number of some sort (for example, Social Security Number), and they help the system process the call more efficiently.
- **Message Collection** allows the caller to leave a recorded message in lieu of waiting for the call to be answered.

The following four vectors in Screen 38 illustrate how the mutual fund company handles telephone calls.

NOTE:
Typically your vector would check to see if queue slots are available.

```
VDN (extension=1030  name=''ABC Inv''  vector=10  display override=''y'')
Vector 10
      1. wait-time 0 secs hearing ringback
      2. collect 1 digits after announcement 3531
         (''Thank you for calling ABC Investments. If
         you wish to open a new account, please dial 1.  If
         you wish to make an account inquiry, please dial 2.
         If you wish to know the current net asset values of
         our funds, please dial 3.'')
      3. route-to number 1031 with cov y if digit = 1
      4. route-to number 1032 with cov y if digit = 2
      5. route-to number 1033 with cov y if digit = 3
      6. route-to number 0 with cov n if unconditionally
      7. disconnect after announcement none
VDN (extension=1031  name=''New Account''  vector=11)
Vector 11
      1. goto step 5 if calls-queued in split 1 > 19
      2. queue-to split 1 pri t
      3. announcement 3535
      4. wait-time 10 secs hearing music
      5. collect 1 digits after announcement 4020
         (''We're sorry.  All of our operators are busy at
         the moment.  If you'd like to leave your name and
         telephone number so that we can get back to you,
         dial 1.'')
      6. goto step 9 if digit = 1
      7. announcement 3537
      8. wait time 50 secs hearing music
      9. goto step 6 if unconditionally
     10. messaging split 5 for extension 4000
     11. announcement 3538 (''We're sorry, we cannot take
         your message at this time.  You may continue to hold, or
         you can call back later.'')
     12. goto step 6 if unconditionally
VDN (extension=1032  name=''Account Inq''  vector=12)
Vector 12:
      1. wait-time 0 secs hearing ringback
      2. collect 6 digits after announcement 3533
         (''Please enter your 6-digit account number.'')
      3. goto step 8 if calls-queued in split 1 > 19
      4. queue-to split 1 pri m
      5. announcement 3535
      6. wait-time 60 secs hearing music
      7. goto step 8 if unconditionally
      8. collect 1 digits after announcement 4020
         (''We're sorry.  All of our operators are busy at
         the moment.  If you'd like to leave your name and
         telephone number so that we can get back to you,
         dial 1.'')
      9. goto step 8 if digit = 1
VDN (extension=1033  Name=''Net Asset Val''  Vector=13)
Vector 13:
      1. disconnect after announcement 3534
         (''The net asset values of our funds at the close
         of the market on Wednesday, May 15 were as follows:
         ABC Growth.....33.21.....up 33 cents; ABC
         High Yield.....11.48.....down 3 cents.'')
```

Screen 38. Example 4: DIVA and Data/Message Collection

When the call is placed, vector processing begins in vector 10, which is the main vector. Step 1 of the vector contains the **wait-time** command, which is placed before the **collect digits** command in step 2 to provide the caller with feedback in the event a tone detector is not immediately available. Once a tone detector is connected, the **collect digits** command provides an announcement requesting the caller to enter 1, 2, or 3, depending upon the service desired. If the caller enters a digit other than one of the three mentioned, or if the caller fails to enter any digits within 10 seconds, the command fails, and the call is routed to the attendant (step 6). On the other hand, if the caller enters 1, 2, or 3 within 10 seconds, the call is routed to the vector specified in the appropriate **route-to number** command, which appears in steps 3, 4, and 5.

Let's say that, when prompted, the caller enters 3 because he or she wants to learn about the net asset values of the company's funds. In such a case, the **route-to number** commands in step 3 and in step 4 fail because, in each case, the digit tested for in the condition portion of the command is not 3. However, the **route-to number** command in step 5 succeeds because the digit tested for matches the one entered by the caller. Accordingly, the call is routed to VDN extension 1033, and vector processing continues in vector 13.

The **announcement** command in step 1 of vector 13 provides the caller with the information on net asset values and then disconnects the call.

The process just described, whereby the caller receives information as a result of making a request at the prompt, is an example of the Data In/Voice Answer (DIVA) application.

Returning to the main vector, suppose another caller wants to make an inquiry into his or her account and the caller enters 2 when prompted. In such a case, step 3 fails, but step 4 succeeds. Accordingly, the call is routed to VDN extension 1032, and vector processing continues in vector 12.

The **collect digits** command in step 2 of vector 12 first requests the caller to enter his or her 6-digit account number. The command then collects the digits entered by the caller. Whether or not the caller correctly enters the digits, the **queue-to split** command in step 4 queues the call. If an agent does not immediately answer the call, the standard announcement is provided in step 5 and, if necessary, a delay is provided in step 6 . The **goto step** command in step 7 sends the call control back to step 8, thus ensuring that the announcement-wait cycle will continue until the agent answers the call, or until the caller abandons the call.

The process just described, whereby the caller, when prompted, enters digits that comprise an official number (an account number, in this case), is an example of the Data Collection application. If the agent has a CALLR-INFO button or a two-line display, the agent can see the digits entered by the caller. As a result, the agent need not request the account number from the caller.

Finally, suppose a third caller wants to open an account and that he or she enters 1 when prompted in the main vector. In such a case, step 3 of the main vector is successful. Accordingly, the call is routed to VDN extension 1031, and vector processing continues in vector 11.

In step 2 of vector 11, the call is queued to the main split. Thereafter, if necessary, step 3 provides the appropriate announcement, and step 4 provides a delay period. The announcement in step 5 provides the caller with the option of leaving a recorded message for the mutual fund company instead of having his or her call wait in queue. (This is an example of the Message Collection application.) The caller is instructed to enter 1 if he or she wishes to leave a recorded message. If the caller does not enter 1, the **goto step** command in step 6 fails, and an announcement-wait cycle is implemented by steps 7 , 8 , and 9 until the call is answered or abandoned. If the caller does enter 1 within 10 seconds, step 6 passes control to step 10 . The messaging split command in step 10 attempts to connect the caller to an AUDIX or Message Center split so that the caller can leave a message. If the connection is made, the caller first hears ringback and can then leave a message. If the connection is not made, the step is unsuccessful, and step 11 provides an announcement indicating that a connection could not be made. Thereafter, the **goto step** command in step 12 sends call control back to step 7, which is the first step of the aforementioned announcement-wait cycle.

Vector Exercises

This section presents several typical business world scenarios involving telephone usage, and it shows how to write one or more vectors to handle each of these scenarios.

Note that the vectors presented here are intended to be “suggested solutions.” The customer should take into account his or her requirements and budget in selecting and/or writing vectors.

NOTE:
Exercise 1 in this section presents two solutions, one of which involves Call Prompting, which is discussed in Chapter 4.

Exercise 1: Emergency and Routine Service

Write a vector that does the following:

- Delivers the following message to handle emergency calls: “We are aware of the power outage in the northeastern part of the city. Crews have been dispatched. If you are calling for other reasons, please hold to see if an operator is available.”
- Enables the caller to speak with an agent (if available) concerning a nonemergency matter.

Suggested Solution 1

```
 1. wait-time 0 secs hearing ringback
 2. announcement 4100 ("We are aware of the
    power outage in the northeastern part of the city.
    Crews have been dispatched.  If you are calling for
    other reasons, please hold to see if an operator
    is available.")
 3. wait-time 2 secs hearing ringback
 4. goto step 10 if calls-queued in split 1 pri l > 20
 5. queue-to split 1 pri l
 6. wait-time 6 secs hearing music
 7. announcement 4200 ("We're sorry.  All of
    our operators are busy.  Please hold.")
 8. wait-time 10 secs hearing music
 9. goto step 7 if unconditionally
10. disconnect after announcement 4200 ("We're
    sorry.  All of our operators are busy at the moment.
    Please call back at your convenience.")
```

Screen 39. Emergency and Routine Service (Call Vectoring Option)

In step 2 of this vector, Screen 39, the **announcement** command provides the caller with the appropriate emergency information, and it invites the caller to hold if he or she wishes to speak with an operator on another matter. If the caller holds, the caller hears several seconds of ringback provided by the **wait-time** command in step 3. Thereafter, the **goto step** command in step 4 checks whether there are more than 20 calls queued in split 1. If so, a branch is made to step 10, where the **disconnect after announcement** command first informs the caller that the call cannot be serviced at this time and then drops the call.

On the other hand, if 20 or fewer calls are queued to split 1, the call is queued to the split by the **queue-to split** command in step 5. Thereafter, unless the call is answered, feedback in the form of music is provided by step 6 and an announcement urging the caller to hold is provided by step 7. After another wait with music period (if necessary) provided by step 8, the **goto step** command in step 9 branches back to the aforementioned “please hold” announcement in step 7. The resulting “announcement-wait” loop (steps 7 through 9) is then repeated until either an agent answers the call or the caller hangs up.

Suggested Solution 2

```
VDN (extension=1030   name="Hub"   vector=10)
Vector 10:
      1. wait-time 0 secs hearing ringback
      2. collect 1 digits after announcement 3000
         ("We are aware of the power outage in the northeastern
         part of the city.  Crews have been dispatched.  If
         you are calling for other reasons, please press 1.
         Otherwise, please hang up now.")
      3. route-to number 1031 with cov y if digit = 1
      4. announcement 3100 ("Entry not understood.  Please
         try again.")
      5. goto step 2 if unconditionally
VDN (extension=1031   name="Service"   vector=11)
Vector 11:
      1. announcement 4000 ("Please hold.  We will
         try to connect you to an operator.")
      2. wait-time 2 secs hearing ringback
      3. goto step 9 if calls-queued in split 1 pri l > 20
      4. queue-to split 1 pri l
      5. wait-time 6 secs hearing music
      6. announcement 4200 ("We're sorry.  All of
         our operators are busy.  Please hold.")
      7. wait-time 10 secs hearing music
      8. goto step 6 if unconditionally
      9. disconnect after announcement 4200 ("We're
         sorry.  All of our operators are busy at the moment.
         Please call back at your convenience.")
```

Screen 40. Emergency and Routine Service (Call Vectoring and Call Prompting Option)

Suggested Solution 2 involves both Call Vectoring and Call Prompting. Also, it involves two vectors instead of just one vector, and it assumes that the caller has a touch-tone telephone. Modifications to this solution are required to process calls from rotary-dial callers.

The announcement portion of the **collect digits after announcement** command in step 2 of Vector 10 first provides the caller with the appropriate emergency information. It then invites the caller to press “1” if the caller is calling for some other reason. If this is not the case, it finally suggests that the caller hang up.

First, let’s assume that the caller wants to hold the line but enters the incorrect touch-tone digit (“2,” for example). In such a case, the **route-to number** command in step 3 attempts to route the call to VDN extension 1031 according to the entered digit. However, because a number other than “1” has been entered, the call is not routed to the VDN extension. Instead, control is passed to step 4, where the **announcement** command first informs the caller of the input error and then invites the caller to try again. Thereafter, the **goto step** command in step 5 unconditionally sends control back to step 2, where the **collect digits** command ultimately collects the digit entered by the caller. The digit-input loop (steps 2 through 5) continues for as long as the caller enters an incorrect digit.

On the other hand, let’s assume that the caller correctly enters the digit “1” as requested by the **collect digits** command in step 2. In such a case, the **route-to number** command in step 3 sends control to the vector whose VDN extension is “1031” (that is, to Vector 11). Thereafter, the call is processed almost identically to the procedure explained in Suggested Solution 1.

Exercise 2: Late Caller Treatment

Your ACD is staffed by union agents. The latest union agreement stipulates that these agents are free to leave promptly at 5:00 p.m. However, you are concerned about the callers who will call shortly before 5:00 p.m. on any given day and find themselves waiting in queue (and, in effect, ignored) after the top of the hour.

Write a vector that warns late callers that their call may not be serviced. (Business hours are from 8:00 a.m. to 5:00 p.m., Monday through Friday.)

Suggested Solution

```
 1. goto step 15 if time-of-day is all 1700 to all 0800
 2. goto step 15 if time-of-day is fri 1700 to mon 0800
 3. goto step 16 if calls-queued in split 1 pri l > 20
 4. queue-to split 1 pri l
 5. goto step 10 if time-of-day is all 1645 to all 1700
 6. wait-time 20 secs hearing ringback
 7. announcement 100 ("We're sorry, all of our
    agents are busy...Please hold...")
 8. wait-time 998 secs hearing music
 9. stop
10. announcement 200 ("It is almost closing time.
    We will try to service you before we close for the day.
    However, if we are unable to do so, please call back
    at your convenience between 8:00 a.m. and 5:00 p.m.,
    Monday through Friday.")
11. wait-time 30 secs hearing music
12. goto step 14 if time-of-day all 1700 to all 1701
13. goto step 11 if unconditionally
14. disconnect after announcement 300 ("We're
    sorry, our office is now closed.  Please call back
    at your convenience between 8:00 a.m. and 5:00 p.m.,
    Monday through Friday.")
15. disconnect after announcement 400 ("We're
    sorry, our office is closed.  Please call back at
    your convenience between 8:00 a.m. and 5:00 p.m.,
    Monday through Friday.")
16. disconnect after announcement 500 ("We're
    sorry, we cannot service your call at this time.
    Please call back at your convenience between
    8:00 a.m. and 5:00 p.m., Monday through Friday.")
```

Screen 41. Late Caller Treatment

This vector, Screen 41, provides specific treatment for calls coming into the switch after working hours, during the weekend, or as the working day comes to a close.

The **goto step** command in step 1 checks whether the call is being placed during nonworking hours during the week (that is, between 5:00 p.m. and 8:00 a.m. on any day of the week). If the call is being placed at this time, a branch is made to step 15, where the **disconnect after announcement** command first informs the caller that the office is closed and then drops the call. On the other hand, if the call is not being placed at this time, control is passed to step 2, where another *goto step* command checks whether the call is being placed during "weekend" hours (that is, between 5:00 p.m. Friday and 8:00 a.m. Monday). If so, a branch is made to step 15, as is the case for a failure of the *goto step* command in step 1. On the other hand, if the call is not being placed at this time, control is passed to step 3.

The **goto step** command in step 3 checks for the number of calls in split 1. If more than 20 calls are queued to split 1, control is passed to step 16, where the **disconnect after announcement** command first informs the caller that the call cannot be serviced at this time and then disconnects the call. On the other hand, if 20 or fewer calls are queued to split 1, control is passed to step 4, where the **queue-to split** command queues the call to split 1.

Control is then passed to step 5, where the **goto step** command checks whether the current time is any time between 4:45 p.m. and 5:00 p.m. inclusive (or, in other words, very close to [if not] closing time). If the current time does not fall within this clock range, the **wait-time** command in step 6 provides the caller with 20 seconds of ringback. Thereafter, the **announcement** command in step 7 plays the appropriate "hold" message, and the **wait** command in step 8 provides the caller with 998 seconds of music. Finally, the **stop** command in step 9 halts vector processing, and the call remains in queue until either the agent answers the call or the caller hangs up.

On the other hand, if the current time is any time between 4:45 p.m. and 5:00 p.m., inclusive when step 5 is executed, a branch is made to step 10, where the appropriate "late caller" announcement is provided to the caller. Thereafter, the **wait-time** command in step 11 provides the caller with 30 seconds of music. Control is then passed to step 12, where the **goto step** command checks whether the time has now advanced to any time between 5:00 p.m. and 5:01 p.m., inclusive. If so, control is passed to step 14, where the **disconnect after announcement** command first informs the caller that the office is now closed and then invites the caller to call back at the appropriate time before finally disconnecting the call.

On the other hand, if the time is still between 4:45 p.m. and 5:00 p.m. inclusive, control is passed to step 13, where the **goto step** command branches back to the **wait-time** command in step 11. The resulting loop consisting of steps 11 through 13 is repeated for as long as the time is between 4:45 p.m. and 5:00 p.m. inclusive, or until the caller hangs up. Once step 12 is executed at least a second after 5:00 p.m., control is passed to step 14 as described previously.

Exercise 3: Messaging Option

Write a vector that:

- Does the following if the oldest call waiting has been in queue longer than 75 seconds:
  - — Sends the call to AUDIX (if possible)
  - — Delivers to the caller the following personalized AUDIX message: “All of our MegaSports agents are busy...Please leave your name and telephone number.”
- Plays for the caller 30 seconds of ringback
- Plays for the caller (after the ringback) an announcement followed by music.

Suggested Solution

```
 1. goto step 8 if oldest-call-wait in split 5 pri l > 74
 2. goto step 8 if calls-queued in split 5 pri l > 20
 3. queue-to split 5 pri l
 4. wait-time 30 secs hearing ringback
 5. announcement 1000 ("All of our MegaSports
    agents are busy...Please wait...")
 6. wait-time 998 secs hearing music
 7. stop
 8. announcement 2000 ("We're sorry, all of our
    MegaSports agents are busy.  If you'd like to leave a
    message, please do so after the tone.  Otherwise, please
    call back between 8:00 a.m. and 5:00 P.M, Monday through
    Friday. Thank you.")
 9. messaging split 20 for extension 4000
10. disconnect after announcement 2050 ("We're sorry, we are unable
    to take your message at this time.  Please call back
    between 8:00 a.m. and 5:00 p.m., Monday through Friday.
    Thank you.")
```

Screen 42. Messaging Option

The **goto step** command in step 1 of this vector, Screen 42, checks whether the oldest call waiting in split 5 has been waiting for 75 seconds or more. If so, control is passed to step 8, where the **announcement** command first informs the caller that all the agents are busy and then invites the caller to either call back at the appropriate time or leave a recorded message for the agent. If the caller elects to leave a message, the **messaging split** command in step 9 is executed. Upon execution of the **messaging split** command, an attempt is made to connect the caller to AUDIX so that he or she can leave a recorded message. If the split queue is full, or if the AUDIX link is down, termination to AUDIX is unsuccessful, and vector processing continues at the next vector step, which (as is the case here) usually contains an announcement that provides the caller with the appropriate apology and subsequent instructions. If the caller is successfully connected to AUDIX, vector processing terminates, and a message may be left for the specified mailbox (4000, in this case).

Returning to step 1, if on the other hand the oldest call waiting in split 5 has been waiting fewer that 75 seconds, control is passed to step 2, where another **goto step** command checks for the number of calls in split 5. If more than 20 calls are queued to split 5, control is passed to step 8. Thereafter, the procedure for the messaging option provided in the previous paragraph is implemented. On the other hand, if there are 20 or fewer calls waiting in split 5, control is passed to step 3, where the **queue-to split** command queues the call to the split. Thereafter, the obligatory `wait-time` and `announcement` steps (steps 4 through 6) are executed, followed by the `stop` step (step 7).

Call Vectoring Commands

A

This appendix provides various information about the commands used within Call Vectoring. Specifically, the following information is presented:

- Table that contains a brief description of each command's function and also the page where the command can be referenced
- Table that identifies the commands available in Basic Call Vectoring and/or Call Prompting
- Job aid tables that graphically illustrate how to use the Call Vectoring commands
- Manual page directory that details the purpose and function of the Call Vectoring commands and also any relevant interactions involving the commands
- Table that summarizes the criteria for the success/failure of the Call Vectoring commands.

Command Description/Reference

Table 5 provides a brief description of the function of each of the Call Vectoring command. See the listed page number for a complete description of the command.

Table 5. Command Description/Reference Table

Command	Function	Page
announcement	To connect caller to delay recording.	95
busy	To connect caller to busy tone.	97
check split	To connect/queue call on a conditional basis.	98
collect digits	To prompt caller for digits.	101
converse-on split	To deliver a call to a converse split and to activate a Voice Response Unit (VRU).	105
disconnect	To force disconnect of call with optional announcement.	115
goto step	To cause unconditional/conditional branch to another step in the vector.	117
goto vector	To cause unconditional/conditional branch to another vector.	121
messaging split	To allow caller to leave message for callback.	125
queue-to split	To connect/queue call to the primary split	128
route-to	To connect call to destination entered via **collect digits** command, or to connect call to internal/external destination.	130
stop	To stop further vector processing.	136
wait-time	To initiate feedback to caller and delay processing of the next step.	137

Continued on next page

Command/Option Summary

Table 6 indicates which Call Vectoring commands can be used within Basic Call Vectoring and/or Call Prompting.

Table 6. Command/Option Summary Table

Command	Basic	Prompting	Other Options Required
announcement	x	x	
busy	x		
check split if <condition>	x		ACD
collect digits		x	
converse-on split	x		
disconnect	x		
goto step/vector if unconditionally	x	x	
goto step/vector if <condition> in split	x		ACD
goto step/vector if digits		x	
goto step/vector if time-of-day	x		
messaging split	x	x	
messaging split active/latest	x	x	
queue-to split	x		ACD
route-to digits with cov y (n)		x	
route-to number if digit		x	
route-to number if unconditionally with cov y (n)	x	x	
route-to number if digit with cov y (n)		x	
route-to number if unconditionally	x	x	
stop	x	x	
wait-time <time>	x	x	
wait-time <time> hearing i-silent	x	x	

Command Job Aid

Table 7. Vectoring Commands

announcement[1]_____ (1- to 5-digit extension)
busy
collect[1]____digits after announcement_____ (1-16)(Optional: 1- to 5-digit extension or none)
disconnect after announcement_____ (1- to 5-digit extension or none [default])
messaging split[1]_____for extension _____ (1-99 [csi/si])(1- to 5-digit extension, active [default], latest)[2] (1-600 [r])

Continued on next page

1. This command is also available with Call Prompting.
2. Active means the called VDN as changed by VDN override. Latest means the VDN assigned to the vector in which the call is currently being processed.

Table 8. Vectoring Commands

queue-to split_____pri__ (**l**ow, **m**ed, **h**igh, **t**op) (1-99 [csi/si]) (1-600 [r])
route-todigits[1]with coverage__ y or n [default] route-tonumber_____with cov__ ifdigit[1]_____ (1- to 16-digit number,(y or(=)(0-9) which may include ~p,n [default](0-9, #) ~s, ~w, ~W, ~m, ∗, or #)
stop[2]
wait-time[2]_____secs hearing____ (0-999)(i-silent) (0-998 in 2-second intervals — R6 limit) _____ mins hearing ____ (0-480)(i-silent) _____ hrs hearing ____ (0-8) (i-silent)

1. The Call Prompting feature must be enabled.
2. This command is also available with Call Prompting.

Table 9. Vectoring Commands

check split_____pri__ifavailable-agents>_____
(1-99 [csi/si])(**l**ow, **m**ed, (1-150)
(1-600 [r])**h**igh, **t**op)

calls-queued<_____
(1-200[csi/si])
(1-999 [r])

staffed-agents>_____
(1-150)

unconditionally

Continued on next page

Table 10. Vectoring Commands

converse-on split_____pri__passingDATA 1[1]andDATA 2
(1-99 [csi/si])(**l**ow, **m**ed,(string of up to(string of up to
(1-600 [r])**h**igh, **t**op)six digits orsix digits or
asterisks, “vdn,”asterisks, “vdn,”
“digits,”[2]“digits,”[2]
“#,” or “none”)“#,” or “none”)

Continued on next page

1. If DATA 1 is administered as “none,” DATA 2 must also be administered as “none.”
2. The Call Prompting feature must be enabled.

Table 11. Vectoring Commands

goto step[1]____ ifavailable- _____ _____ _____
(1-32) agents in split (1-99 [csi/si]) (<) (1-150)
(1-600 [r]) (>, =) (0-149)

staffed-_____
agents in split(1-99 [csi/si])(<)(1-150)
(1-600 [r])(>, =)(0-149)

calls-queued_____pri _____
in split(1-99 [csi/si])(**l**ow, **m**ed,(<)(1-150)
(1-600 [r])**h**igh, **t**op)(>, =)(0-149)

digits[2]_____
(=)(1-16 characters: 0-9, #, ?, none)[3]

1. The “goto vector” command is identical to the “goto step” command, except the word “step” is replaced by the word “vector”. The valid values for vector are: 1-4 [R6], 1-10 [R7csi/si], 1-20 [R7r].
2. The Call Prompting feature must be enabled.
3. A ? can be entered in any character position and matches any character in that single character position.

Command Directory

The manual page directory in this section lists and discusses all of the commands used within Call Vectoring. For each command presented, the following is provided: purpose, syntax, valid entries, requirements, an example, description of the command's operation, answer supervision considerations, feature interactions, and BCMS interactions. The following points concerning the appearance of the command line are in effect:

- Data that must be entered as part of the command line is shown in **bold**.
- Variable fields that (in most cases) must be completed are enclosed in < >.
- Optional fields are enclosed in [ ].

NOTE:
If a variable field appears within an optional field, an entry for the variable field appears only if the optional field is included during command execution.

Announcement

Purpose

Provides the caller with a recorded announcement.

Syntax

announcement <extension>

Valid Entries

Valid announcement extension number

Requirements

Basic Call Vectoring or Call Prompting software must be installed. Also, integrated announcement board, auxiliary trunk or analog (Tip and Ring [T&R] or Lineside DS1) announcement equipment must be installed. Finally, the announcements themselves need to be administered and recorded. See “Managing Announcements” in the *DEFINITY® ECS Administrator’s Guide* for more information.

Example

announcement *2982*

Operation

The announcement is played from beginning to end unless an agent becomes available. In such a case, the announcement is interrupted and (if manual answering operation is assigned to the agent, or if calls are delivered to the agent on a manual answering basis) ringback is provided. If the call is queued, the call remains as such while the announcement is played. Any feedback provided before an announcement is continued until the announcement is played.

If the announcement’s queue is full, the call retries the announcement step every 5 seconds and for an indefinite period of time before any new vector steps are processed.

The **announcement** command step is skipped, and vector processing continues at the next vector step, whenever any of the following conditions exist:

- Requested announcement is busied out, not available, or not administered.
- Integrated board is not installed.
- External auxiliary trunk or analog equipment is not attached.

For a complete description of the types and operation of announcements see "Managing Announcements" in the *DEFINITY®ECS Administrator's Guide*.

Answer Supervision Considerations

Unless answer supervision has already been sent, it is sent as soon as the command starts to process the call (even before the announcement starts).

Feature Interactions

The command is considered a call acceptance vector command whenever one of the following is true:

- Announcement is available.
- Call is queued for an announcement.
- Announcement is retried.

The command is considered a neutral vector command whenever the announcement is unavailable.

BCMS Interactions

The command is not tracked on BCMS.

Busy

Purpose

Gives the caller a busy signal and causes termination of vector processing.

Syntax

busy

Requirements

Basic Call Vectoring software must be installed.

Operation

The command takes effect on non-CO trunk calls whether or not answer supervision has been sent. However, if the call is on a CO trunk and answer supervision has not been sent, the busy is not passed back by the CO, and the caller continues to hear ringback from the CO. Calls are dropped approximately 45 seconds after the busy tone is applied.

If ISDN-PRI is involved, the application of the busy tone is enabled via D-channel messaging. The network switching office returns the busy tone to the caller. The facility to the switch is dropped, thus making it immediately available for another call.

Answer Supervision Considerations

After the 45-second timeout, an unanswered CO trunk call is answered and then dropped. All other unanswered calls after this timeout are dropped without being answered. For an ISDN call that has not yet queued or been answered, no timeout occurs, and answer supervision is not sent. Instead, a message requesting a busy tone is sent to the network and, subsequently, the trunk is released.

Feature Interactions

None.

BCMS Interactions

A call that is forced busy due to the command is tracked as “OTHER” in the VDN Report.

Check

Purpose

Checks the status of a split for possible termination of the call to that split.

Syntax

check split <split #> pri <priority level> if <condition> [<comparator> <threshold>]

Valid Entries

split #: *1 through 99* (csi/si), *1 through 600* (r)

priority level: *l* (low), *m* (medium), *h* (high), *t* (top)

condition:	comparator:	threshold:
unconditionally	N/A	N/A
available-agents	>	*0-149*
calls-queued	<	*1-200* (csi/si), *1-999* (r)
staffed-agents	>	*0-149*

Requirements

Basic Call Vectoring software must be installed, and the split involved must be vector-controlled.

Examples

check split *22* pri *h* if **unconditionally**

check split *11* pri *l* if **available-agents** > *5*

check split *11* pri *t* if **calls-queued** < *5*

check split *8* pri *m* if **staffed-agents** > *5*

Operation

The **check** command checks the status of a split against conditions specified in the command. If the conditions specified in the command are met, the call is terminated to the split. If the conditions are met but no agents are available, the call is queued to the split and waits for an agent to become available.

Each **check** command may be used with the keyword *split*. The **check split** command requires you to specify the split to be checked.

The command is customized to check for and/or respond to specific conditions. For example, the command can queue/terminate unconditionally. The command can also queue/terminate if any of the following is true:

- Number of available agents is *greater than* the threshold value.
- Number of staffed agents is *greater than* the threshold value.
- Number of calls queued for a specified priority level or higher is *less than* the threshold value.
- Oldest call waiting in queue at the specified priority level or higher has been waiting *less than* the threshold value, which is expressed in seconds.

A call may be queued to up to three splits simultaneously. A call remains queued either until vector processing terminates (via a successful **disconnect**, **busy**, or **route-to** command, or via an abandoned call), the call is routed to another VDN (by a **route-to number** or **route-to digits** command), or the call reaches an agent. When an agent becomes available in any split to which the call is queued, the following actions take place:

- Call begins ringing the agent.
- Call is removed from any other queues.
- Vector processing terminates.

If the desired backup split is one of the splits to which the call is already queued, the call is requeued at the new priority level, provided that the command conditions are met. The step is skipped, and vector processing continues at the next step if any of the following conditions are true:

- Command conditions are not met.
- Desired split's queue is full.
- Desired split has no queue and also no available agents.
- Desired split is not vector-controlled.
- Call is already queued to this split at the specified priority level.
- Call has been previously queued to three different splits.

NOTE:
A *route-to* step to another VDN can be used to remove the call from the splits it is queued to if necessary. The steps in the routed-to vector then can be used to queue to other splits.

Answer Supervision Considerations

No answer supervision is returned.

Feature Interactions

The **check** command can access an AUDIX®/Message Center/Server split in cases where a VDN is assigned as a coverage point. To enable this function, the split must be assigned as a vector-controlled hunt group.

The command is considered a call acceptance vector command whenever one of the following is true:

- Call terminates to an agent.
- Call queues to a split.

The command is considered a neutral vector command when the call neither terminates nor queues.

No COR checking is carried out when a *check* step places a call to a split.

BCMS Interactions

The total number of calls to the VDN that are queued via the command and then answered by an agent within a specified time period is tracked as “ACD CALLS” in the VDN Report. The average time that calls spend in a vector before being connected via the command as an ACD call to an agent is tracked as “AVG SPEED ANS” in the same report.

Collect Digits

Purpose

Allows the user to enter up to 16 digits from a touch-tone phone.

Syntax

collect <# of digits> digits after announcement <extension>

Valid Entries

of digits: 1 through 16

extension: *none* or valid announcement extension

Requirements

Call Prompting software must be activated. Also, at least one TN744 or TN2182 Call Classifier circuit pack must be in the system.

Example

collect *12* digits after announcement *2982*

Operation

The **collect digits** command allows a caller to enter digits from a touch-tone or an internal rotary phone. An optional announcement may be used to request the caller to enter these digits. The announcement can instruct the user to enter an asterisk (∗) if incorrect data is entered. When the caller enters an asterisk, the digits collected for the current **collect digits** command are deleted, digit collection is restarted, and the announcement is not replayed.

In using this command, the maximum number of digits requested of the caller must be specified in the administration of the command. If the caller can enter fewer digits than the maximum specified, the announcement should instruct the caller to terminate the entry with a pound sign (#) digit as an end-of-dialing indicator. If all the digits strings for all the variations of a specific **collect digits** command are terminated with "#," the "#" must be counted as one of the digits. Therefore, the number of digits collected should include any "#" that needs to be collected. Otherwise, the terminating "#" is kept as a dial-ahead digit and is processed by a subsequent **collect digits** command. If fewer digits than the maximum specified are entered, and if the caller does not complete the entry with a pound sign, an interdigit timeout occurs. The timeout terminates the command, and any digits collected prior to the timeout are available for subsequent vector processing.

Generally, processing of the command requires that a tone detector be connected. (If the call originates from an internal rotary phone, no tone detector is needed.) Tone detectors accept the touch-tone digits that are entered by Call Prompting users. Tone detectors are automatically connected as needed by the system.

The connection of the announcement prompt is skipped and the digit collection phase begins whenever one of the following conditions is true:

- Dial-ahead digits exist.
- No announcement is administered for the *collect digits* step.
- Announcement administered for the *collect digits* step does not exist.

Otherwise, an attempt is made to connect the administered announcement. If the announcement to be connected is busy, and if the queue for the announcement is full, or if there is no queue, the calling party continues to hear the current feedback. The system waits 5 seconds and then tries again to connect the call to the announcement. This process continues until the call is successfully queued or connected to the announcement, or until the calling party disconnects from the call. If the queue for the announcement is not full, the call is queued for the announcement.

If the announcement to be connected is available (either initially or after queuing, or after system retry), any previous feedback is disconnected, and the calling party is connected to the announcement.

While the announcement is playing, or while the call is being queued for an announcement, the caller may enter digits at any time. This causes the announcement to be disconnected or removed from the queue, as appropriate, and the digit collection phase to begin. If the caller does not enter any digits during the announcement phases, the digit collection phase begins when the announcement completes.

As soon as the digit collection phase begins, interdigit timing is started, unless the tone detector is already in timing mode (that is, the dial-ahead capability is active and the tone detector is not disconnected).

Digits are *collected* either as digits dialed during the **collect digits** command or as dial-ahead digits dialed since a previous **collect digits** command but prior to the current appearance of the command. Digit collection continues for the current command until one of the following conditions exists:

- Number of digits specified is collected.
- Pound sign (#) digit is collected (signifying end of dialing).
- Interdigit timer expires.

If, during the digit collection phase, a "∗" is encountered within a stream of dialed or dial-ahead digits, all digits that are collected for the current *collect digits* step are discarded. If additional dial-ahead digits occur after the asterisk, these digits continue to be processed. If there are no such digits, and if no tone detector is connected, vectoring continues at the next vector step. If a tone detector is connected, the caller can start entering digits again. In such a case, the announcement is not replayed, and the interdigit timer is restarted.

NOTE:
If an asterisk is entered after the requested number of digits are entered, the asterisk has no effect on the previously entered digits. However, in such a case, the asterisk is treated as a dial-ahead digit for the next **collect digits** command.

When digit collection is completed, and if a tone detector is connected (for a touch-tone phone), the interdigit timer is restarted to detect a timeout for releasing the tone detector. Vector processing then continues at the next vector step. However, the switch continues to collect any subsequent dialed digits (including the pound sign (#) and asterisk (∗) digit) to allow for the dial-ahead capability. These additional "dialed ahead" digits are saved for use by subsequent **collect digits** commands, and they provide the caller with a means to bypass subsequent unwanted announcement prompts. A single "#" digit can be collected and tested by subsequent **route-to**...**if digits** or **goto**...**if digits** commands. Collection of dial-ahead digits continues until one of the following occurs:

- Vector processing stops or is terminated.
- The sum of the digits collected for the current **collect digits** command and the dial-ahead digits exceeds the switch storage limit of 24. Any additional dialed digits are discarded until storage is freed up by a subsequent **collect digits** command.

NOTE:
Any asterisk (*) or pound sign (#) digits count towards the 24-digit limit, as do any dial-ahead digits entered after the asterisk or pound sign digit.

- The tone detector required by the touch-tone phone user to collect digits is disconnected. This occurs under the following conditions:
  - — Successful or unsuccessful *route-to number* step is encountered during vector processing except where the number routed to is a VDN extension.
  - — Successful or unsuccessful *route-to digits* step is encountered during vector processing except where the number routed to is a VDN extension.
  - — Successful or unsuccessful *converse-on* step is encountered during vector processing.
  - — 10 second timeout occurs, during which time the caller does not dial any digits, asterisks (*) or pound signs (#).

    NOTE:
    When the tone detector is disconnected due to a *route-to number*, *route-to digits*, or a *converse-on* step, all dial-ahead digits are discarded. This means that, following a failed *route-to* or *converse-on* step, a subsequent *collect digits* step always requires the caller to enter digits.

Answer Supervision Considerations

Answer supervision is provided as soon as a tone detector is connected and processing of the command starts. The command always provides answer supervision to an incoming trunk if supervision has not been previously provided except that a collect ced/cdpd digits step does not return answer supervision.

Feature Interactions

None.

BCMS Interactions

Digits are not passed to BCMS.

Converse-on

Purpose

Delivers a call to a converse split and activates a voice response script that is housed within a VRU.

Syntax

converse-on split <split #> pri <priority level> passing <data_1> and <data_2>

Valid Entries

split #: *1 through 99* (csi/si), *1 through 600* (r)

priority level: *l* (low), *m* (medium), *h* (high), *t* (top)

data_1, data_2: String consisting of any digits *0-9* and/or one or more asterisks *(*)* and not exceeding a total of six such digits; the pound sign *(#)*; any of the following keywords: *vdn, digits, qpos, wait, none* (with the exception that *none* cannot be included as an entry for data_1 if data_2 has an entry other than *none*).

Requirements

Basic Call Vectoring software must be activated, and the split must be vector controlled. If the keyword *digits* is included within the command syntax, Call Prompting software must be installed. Moreover, Call Prompting software is necessary to allow for the full functionality of Voice Response Integration (VRI).

Examples

converse-on split *1* pri *h* passing *none* and *none*

converse-on split *20* pri *m* passing *123456* and *none*

Operation

NOTE:
Refer to Appendix G, ‘‘Detailed Call Flow and Specifications for Converse—VRI Calls’’, for details regarding call flows, data passing, collection, and return specifications involving the **converse-on** command.

The **converse-on** command is designed primarily to integrate VRUs with the switch. The command affects data passing between the switch and the VRU, and it enables the caller to hear the appropriate voice response script housed in the VRU.

If the command is successful, it delivers the call to a predetermined split, which is referred to as the converse split. Once the call is answered by the VRU, the command may or may not pass data to the VRU (depending upon the parameters of the command). Regardless of whether or not data is passed, the caller is then connected to the VRU, which in turn executes the voice response script. If by this time the call has already queued to a nonconverse split, the call retains its position in the nonconverse split queue. If an agent from the nonconverse split becomes available to service the call while the voice response script is being executed, the switch drops the line to the voice information system and connects the caller to the available agent. The voice information system, in turn, detects the disconnect and terminates the voice response script. Whenever a voice response script is executed, any audible feedback provided by the vector is disconnected, and no further vector steps are executed until the voice response script is executed.

The VRU may or may not eventually return data to the switch. If, once the voice response script is completed, there is no data to be returned from the voice information system to the switch, the VRU drops the line to the switch, and vector processing is reactivated on the switch.

If there is data to be returned to the switch, the “Converse data return code” is outpulsed before the data to be passed is outpulsed. Once all VRU data is received, it is stored in the Call Prompting digits buffer as dial-ahead digits, and vector processing is reactivated. Digits returned by the voice information system are not heard by the caller.

Digits returned from the VRU can be as follows:

- Displayed on the answering agent’s display set (automatically for 2-line displays, or by using the “CALLR-INFO” button for 1-line displays)
- Treated as an extension in a *route-to digits* step
- Used for vector conditional branching in a step containing a command with the *if digits* parameter.

The switch can be set up to pass information in-band to the voice information system. In such a case, the **converse-on** command can outpulse up to two groups of digits to the voice information system. The digits may serve two major purposes: the digits may notify the voice information system of the application to be executed, and they may share call related data, such as caller digits collected by the switch. (In many applications, both application selection and data sharing are required.) The touch tone outpulsing rate is adjustable. See Appendix G, ‘‘Detailed Call Flow and Specifications for Converse—VRI Calls’’, for details.

Since in many cases the digit strings are of variable length, the switch always appends a pound sign (#) character to the end of each digit string. The *Prompt and collect* steps in the voice response script must therefore always be administered to expect “#” as the end-of-string symbol and to include “#” in the digit count.

The sending of “#” prevents excessive delays caused by digit timeouts, and it prevents other problems caused by timeouts. It also ensures that each data field is used to satisfy a single *prompt and collect* step.

Any data passed from the switch to a VRU is outpulsed in-band. The user can administer two time delays on the System Parameter Features form: “converse first data delay” and “converse second data delay.” These delays may range from 0 to 9 seconds with a default of 0 seconds for the converse first data delay and a default of 2 seconds for the converse second data delay. The delays are needed to give the VRU time to invoke an application and to allocate a touch-tone receiver to receive the passed digits.

NOTE:
No time delays are invoked when the keyword “none” is administered.

If <data_1> is not “none,” the converse first data delay timer starts when the call is answered by the VRU. When the timer expires, the <data_1> digits are outpulsed in-band to the VRU. The end-of-string character (#) is then outpulsed.

If <data_2> is not “none,” the converse second data delay timer starts when the end-of-string character (#) from the first digit string is outpulsed. When the timer expires, the <data_2> digits are outpulsed in-band to the VRU. The end-of-string character (#) for the second digit string is then outpulsed. The following values may be administered for <data_1> and <data_2> within the **converse-on** command:

- **Administered digit string**: This string can contain up to six characters consisting of one or more digits (0 through 9) or asterisks (∗). The pound sign (#) may not be included in a digit string because it is reserved as the end-of-string character. However, a single “#” may be administered.

- **digits**: This data type can be used only if Call Prompting is optioned. The digits data type causes the most recent set of digits collected in vector processing, either from the caller or from the network, to be outpulsed. If no digits are available, the end-of-string pound sign (#) is the only character outpulsed.
- **none**: This data type causes no characters to be outpulsed. Also, no end-of-string pound character (#) is outpulsed, and no time delays are invoked.
- **qpos**: This data type causes the value of the queue position of a call in a nonconverse split to be outpulsed. This value is a variable length data item from which between one and three digits can be outpulsed. If the call is not queued, the end-of-string pound sign (#) is the only character that is outpulsed.

  NOTE:
  The use of this keyword is not recommended with multiple split queuing because any queue position value that is sent may not be meaningful. If the call is queued to multiple nonconverse splits, the value of the caller's queue position in the first nonconverse split is sent.

  This data may be used by the voice information system to inform callers of their position in queue or to decide whether to execute a long or short version of a voice response script.
- **vdn**: This data type causes the VDN extension to be outpulsed. In cases where multiple VDNs are accessed, normal VDN override rules determine which VDN extension is outpulsed.
- **#**: This is the only character outpulsed. Outpulsing this character causes the corresponding **prompt and collect** command in the voice response script to be skipped.

The switch always outpulses a pound character (#) at the end of each digit string. Where "#" is administered, or where the "digits" keyword is administered and the last digit collected from the caller is "#," only one "#" is outpulsed. No "#" is outpulsed when the keyword "none" is administered.

If *data_1* is administered as "none," *data_2* must also be "none."

Answer Supervision Considerations

Answer supervision is returned only once during the life of a call. If a call is answered as a result of a *converse-on* step, answer supervision is sent only if it has not been sent previously. If digits are passed to the VRU, answer supervision is not sent until after the digits are outpulsed.

Feature Interactions

- Abandon Call Search

  If the *converse-on* step places a call to a hunt group, and if the incoming call was placed via a trunk group with Abandon Call Search activated, the system checks that the calling party has not abandoned the call (that is, hung up) before terminating to an agent.

- Audio Information Exchange (AUDIX)

  If a *converse-on* step calls AUDIX, the call is treated as a direct call to AUDIX. The caller hears the “welcome to AUDIX” message and may retrieve his or her messages in the usual manner.

  If a call is forwarded to or covers to a VDN and is then delivered to an AUDIX hunt group by a *converse-on* step, the call to AUDIX is treated as a redirected call, and the caller may leave a message for the principal.

- Auto-Available Splits

  A *converse-on* step may place a call to an auto-available split. Auto-available converse splits are recommended for Voice Response Integration (VRI).

- Basic Call Management System (BCMS)

  BCMS tracks calls placed by a *converse-on* step to a BCMS-measured hunt group. Since with the *converse-on* step it is now possible for a call to be “answered” in more than one split, trunk totals may no longer match split totals. However, VDN totals and trunk totals will match.

- BCMS VDN Reports

  For call tracking in BCMS VDN reports, a *converse-on* step is treated like an *announcement* step. A call is considered “answered” when it is answered by a nonconverse split but never when it is answered by a converse split.

- Call Coverage

  Call Coverage does not apply because the *converse-on* step may deliver calls only to vector-controlled splits, which do not have coverage paths.

- Call Detail Recording

  For incoming calls to a VDN, the duration of the call is recorded from the time answer supervision is returned. Answer supervision is returned for a successful *converse-on* step. No ineffective call attempt records are generated for *converse-on* steps that fail. Also, no outgoing calls can be placed by a *converse-on* step.

- Call Park

  Calls placed by a *converse-on* step may not be parked.

- Call Pickup

  Calls placed by a *converse-on* step ringing at an agent station may be picked up if that agent is part of a pickup group. Subsequent transfers are denied.

- Call Prompting

  The Call Prompting customer option must also be enabled to gain full VRI functionality. Without Call Prompting, any data returned by the voice information system cannot be collected and processed by the switch.

  If the *converse-on* step places a call to a split of live agents, any digits collected previously may be displayed by agents using the callr-info button.

- Class of Restriction (COR)

  As is the case for the *queue-to split* and *check split* vector steps, no COR checking is carried out when a *converse-on* step places a call to a split.

- Conference

  Any attempt to conference a call placed by a *converse-on* step is denied.

- Coverage Callback

  A call placed by a *converse-on* step does not follow any coverage paths. Therefore, Coverage Callback is not available. Also, if a call reaches a *converse-on* step via a VDN in a coverage path, coverage callback cannot be used.

- Direct Department Calling (DDC)

  A converse split may be administered as a direct department calling split.

- Hunt Groups

  The *converse-on* step may deliver a call to a vector-controlled hunt group, ACD split, Message Center or AUDIX hunt group.

- Intercept Treatment

  A caller is never given intercept treatment upon execution of a *converse-on* step. Failing to place a converse call successfully results in the failure of the *converse-on* step. Vector processing continues at the next vector step.

- Interflow

  Since a *converse-on* step can place calls only to hunt groups that are vector-controlled, and since the activation of Call Forwarding for a vector-controlled hunt group is blocked, calls placed by a *converse-on* step to a hunt group cannot interflow.

- Intraflow

  Since a *converse-on* step can place calls only to hunt groups that are vector-controlled (that is, without coverage paths), intraflow is not possible.

- Live Agents

  Although not recommended, the switch does not prevent a *converse-on* step from delivering a call to a group of live agents. To the agent, the call looks like any other ACD call. However, certain features, such as call transfer, conference, and supervisor assist are denied.

  The answering agent can display any digits collected prior to executing the *converse-on* step by using the “callr-info” button.

- Message Center

  The *converse-on* step may deliver calls to message hunt groups. Such calls are treated as direct calls to the message center agent.

  If a call is forwarded to a VDN and then delivered to a message split by a *converse-on* step, the call is treated as a redirected call.

- Multiple Split Queuing

  A call can be queued to three different splits and then to a converse split as a result of a *converse-on* step.

- Music on Hold

  During the data return phase of a *converse-on* step, the caller is temporarily placed on hold. Music on hold, if administered, is suppressed.

- Non-Vector Controlled Splits

  A *converse-on* step may not place a call to a nonvector-controlled split.

- Priority Levels

  A call placed by a *converse-on* step may be queued at one of four priority levels: low, medium, high or top.

- Priority Queuing

  The queue priority of a call placed by a *converse-on* step is administrable on the vector step.

- Queue Status

  All queue status display, queue status indication and queue warning wall lamp feature capabilities also apply to calls queued by the *converse-on* command.

- Queuing

  Calls handled by the *converse-on* step queue when they are delivered to busy hunt groups. Call Vectoring audible feedback is not disconnected while a converse call is in queue.

  If a *converse-on* step is executed while a call is queued to a nonconverse split, the call remains in queue for the nonconverse split.

  The queue priority of the call is administrable on the vector step.

- Recorded Announcement

  VRI may be used to increase the system's recorded announcement capacity by off-loading some recorded announcements to the VRU. Callers can be redirected by the *converse-on* step to a group of VRU ports and use data passing to specify the correct announcement to play.

- Redirection on No Answer

  If a *converse-on* step places a call to a hunt group with a "no answer timeout" administered, and if the call rings at an agent terminal/port for longer than the administered timeout, the call is redirected, and the agent/port is put into the AUX work state (or logged out if the agent is a member of an auto-available split).

  Thereafter, under Redirection on No Answer, the call is requeued to the split unless there is no room in the queue or unless this is an auto-available split whose agents are all logged out. If the call cannot be requeued, the *converse-on* step fails, a vector event is logged, and vector processing is restarted at the next vector step.

- Service Observing

  Calls placed by a *converse-on* step may be service-observed. To prevent the observer from hearing tones being outpulsed to the VRU, the observer is not connected to the call until the data passing phase is complete. If data is returned by the VRU, the observer is put in service observing pending mode, and the calling party is temporarily put on hold while the VRU digits are outpulsed. Upon completion of the converse session, and once the VRU hangs up the line, the observer remains in service observing pending mode.

  It is recommended that a service observing warning tone not be administered since the warning tone may interfere with the interaction between the voice information system and the calling party.

- System Measurements

  System measurements track converse calls to hunt groups and attendant groups.

- Touch-Tone Dialing

  Any touch-tone dialing by the calling party during the digit passing phases of a session involving a *converse-on* step does not result in corruption of data or in the collection of this data in the form of dial-ahead digits by the switch.

  Only after the digit-passing phase from the switch to the voice information system is completed can the calling party enter touch-tone digits in response to a voice information system prompt. Only after the voice information system to the switch data return phase is completed and an additional *collect digits* vector step is executed can the calling party enter a touch-tone response to a switch prompt.

- Transfer

  A call placed by a *converse-on* step may not be transferred. The only form of transfer allowed is the data-passing operation during the data return phase at the end of a voice response script.

  If an illegal attempt to transfer a converse call is made, a vector event is logged, the line to the voice information system is dropped, and vector processing is reactivated at the next vector step.

  If an illegal transfer is attempted by a live agent with a multifunction set, the transfer is denied and the agent may reconnect to the call.

- Transfer out of AUDIX

  If a *converse-on* step delivers a call to an AUDIX hunt group, and if the calling party then attempts to transfer out of AUDIX, the transfer fails, and vector processing is reactivated at the next vector step.

- Uniform Call Distribution (UCD)

  A converse split may be administered as a Uniform Call Distribution split.

- VDN as a Coverage Point

  If a call covering to a VDN is processed by the *converse-on* command and subsequently reaches a station user (that is, a member of a “converse split”), and if the “converse split” agent attempts to activate Consult (coverage), or Coverage Leave Word Calling, any of these coverage attempts is denied because the call is still in vector processing. If the “converse split” is an AUDIX/Message Center split, the call covered to the VDN is treated like a redirected call to AUDIX/MCS; the original principal and reason for redirection is used in the same manner as a Call Forwarded call to a VDN.

- VDN Override

  If a call that accesses multiple VDNs encounters a *converse-on* step passing “vdn,” normal override rules determine which VDN number is outpulsed to the VRU.

- VDN Reports

  For call tracking in BCMS VDN reports, a *converse-on* step is treated like an *announcement* step. A call is considered “answered” when it is answered by a nonconverse split but not when it is answered by a converse split.

- Vector-controlled Splits

  A *converse-on* step may place a call to a split only if that split is administered as a vector-controlled split.

BCMS Interactions

BCMS tracks calls placed by a *converse-on* step to a BCMS-measured split. Since with the *converse-on* step it is now possible for a call to be “answered” in more than one split, trunk totals may no longer match split totals. However, VDN totals and trunk totals will match.

For call tracking in BCMS VDN reports, a *converse-on* step is treated like an *announcement* step. A call is considered “answered” when it is answered by a nonconverse split but not when it is answered by a converse split.

Disconnect

Purpose

Ends treatment of a call and removes the call from the switch. Also allows the optional assignment of an announcement that will play immediately before the disconnect.

Syntax

disconnect after announcement <extension>

Valid Entries

extension: *none* or valid announcement extension

Requirements

Basic Call Vectoring software must be installed. Also, the relevant announcements must be administered and recorded.

Example

disconnect after announcement *2556*

Operation

While the command's optional announcement is playing, the call remains in queue and can be connected to an agent. When the announcement completes (or is not specified), the command forces a disconnect, ends the treatment of the call, and removes the call from the switch.

Answer Supervision Considerations

If the switch has not yet sent answer supervision, the switch does so immediately before disconnecting the call, whether an announcement is specified or not. If an announcement is specified, answer supervision is given before an attempt is made to connect the announcement. The exception is for ISDN calls, where the disconnect can occur without answer supervision being sent when an announcement is not played.

Feature Interactions

The command is considered a call acceptance vector command whenever an announcement is included within the command and one of the following is true:

- Announcement is available.
- Call is queued for an announcement.
- Announcement is retried.

The command is considered a call denial vector command whenever one of the following is true:

- No announcement is included within the command.
- Announcement is included within the command, but the announcement is unavailable.

BCMS Interactions

A call that is disconnected via the command is tracked as “OTHER” in the VDN Report.

Goto Step

Purpose

Allows conditional or unconditional movement (branching) to a preceding or subsequent step in the vector.

Syntax

goto step <step #> if **unconditionally**

goto step <step #> if **digits** <comparator> <digits>

goto step <step #> if **digits** <option> table <table>

goto step <step #> if **time-of-day** is <day> <hour>: <minute> to <day> <hour>: <minute>

goto step <step #> if **no match**

Conditions = available-agent, staffed-agents:

goto step <step #> if <condition> in split <split #> <comparator> <threshold>

Conditions = calls-queued, oldest call-wait:

goto step <step #> if <condition> in split <split #> pri <priority level> <comparator> <threshold>

Condition = counted-calls:

goto step <step #> if <condition> to vdn <vdn> <comparator> <threshold>

Valid Entries

step #: *1-32*

split #: *1 through 99* (csi/si), *1 through 600* (r)

condition:	comparator:	threshold:
unconditionally	N/A	N/A
available-agents	>, =, <	*0-149* *1-150*
calls-queued	>, =, <	*0-199* (csi/si), *0-998* (r) *1-200* (csi/si), *1-999* (r)
counted-calls	>, =, <	*0-999* *1-999*
oldest call-wait	>, =, <	*0-999* seconds (1-second increments) *1-999* seconds (1-second increments)
staffed-agents	>, =, <	*0-149* *1-150*

digits: the following values are accepted:

command	comparator	value
goto step <step#> if digits	=	String of 0-9, #, none, ?

priority level: *l* (low), *m* (medium), *h* (high), *t* (top)

day: *mon, tue, wed, thu, fri, sat, sun, all (that is, "on any day of the week")*

hour: *00* to *23* (24-hour format)

minute: *00* to *59*

vdn: assigned vdn extension, *active*, *latest*. Active is the active called VDN as modified by VDN override rules. Latest is the VDN assigned to the vector in which the call is currently being processed.

Requirements

- Basic Call Vectoring software must be installed for all the options.
- Call Prompting software is required for the digits option.

Examples

goto step *8* if **available-agents** in split *67 < 5*

goto step *12* if **calls-queued** in split *51* pri *t < 17*

goto step *7* if **time-of-day** is *mon 16:30* to *tue 7:30*

goto step *11* if **oldest-call-wait** in split *26* pri *t >= 20*

goto step *10* if **counted-calls** to vdn *5372 >= 50*

Operation

If the command syntax includes **unconditionally**, the command always branches. The unconditional form of the command is commonly used for skipping vector commands as well as for looping through vector commands.

Otherwise, branching takes place according to one of the conditions that follow:

- The number of available agents in the indicated split meets the constraints defined by the comparator and the threshold value.
- The number of queued calls in the indicated split and at the specified priority level (or higher) meets the constraints defined by the comparator and the threshold value.
- The number of active calls in the indicated VDN meets the constraints defined by the comparator and the threshold value.
- The oldest call-waiting in the indicated split at the specified priority level (or higher) has been waiting for a period of time within the constraints defined by the comparator and the threshold value, which is expressed in seconds.
- The number of staffed agents in the indicated split meets the constraints defined by the comparator and the threshold value.

- The digits collected via the **collect digits** command match the criteria defined by the comparator for the specified digit string. The “#” digit can be tested against as a single digit.
- The time-of-day criteria are met.

NOTE:
The syntax for this condition can be illustrated by a couple of examples, as follows: *mon 8:01 to fri 17:00* means “anytime between 8:01 a.m. Monday through 5:00 p.m. Friday,” and *all 17:00 to all 8:00* means “between 5:00 p.m. and 8:00 a.m. on any day of the week.”

Answer Supervision Considerations

The call answer is not affected by the command.

Feature Interactions

None.

BCMS Interactions

The command is not tracked on BCMS.

Goto Vector

Purpose

Allows conditional or unconditional movement (branching) to another vector. The *goto vector* step does not remove a call from queues in which it is already placed.

Syntax

goto vector <vector #> if **unconditionally**

goto vector <vector #> if **digits** <comparator> <digits>

goto vector <vector #> if **digits** <option> table <table>

goto vector <vector #> if **time-of-day** is <day> <hour> : <minute> to <day> <hour> : <minute>

Conditions = available-agent, staffed-agents:

goto vector <vector #> if <condition> in split <split #> <comparator> <threshold>

Conditions = calls-queued, oldest call-wait:

goto vector <vector #> if <condition> in split <split #> pri <priority level> <comparator> <threshold>

Condition = counted-calls:

goto vector <vector #> if <condition> to vdn <vdn> <comparator> <threshold>

Valid Entries

vector #: *1 through 4 (R6), 1 through 10* (R7csi/si), *1 through 20* (R7r)

split #: *1 through 99* (csi/si), *1 through 600* (r)

condition:	comparator:	threshold:
unconditionally	N/A	N/A
available-agents	>, =, <	*0-149* *1-150*
calls-queued	>, =, <	*0-199* (csi/si), *0-998* (r) *1-200* (csi/si), *1-999* (r)
counted-calls	>, =, <	*0-999* *1-999*
oldest call-wait	>, =, <	*0-998* seconds (1-second increments) *1-999* seconds (1-second increments)
staffed-agents	>, =, <	*0-149* *1-150*

digits: the following values are accepted:

command	comparator	value
goto vector <vector #> if digits	=	String of 0-9, #, none, ?

priority level: *l* (low), *m* (medium), *h* (high), *t* (top)

day: *mon, tue, wed, thu, fri, sat, sun, all*

hour: *00* to *23* (24-hour format)

minute: *00* to *59*

vdn: assigned vdn extension, active, latest. Active is the active called VDN as modified by VDN override rules. Latest is the VDN assigned to the vector in which the call is currently being processed.

Requirements

- Basic Call Vectoring software must be installed for all the options.
- Call Prompting software is required for the digits option.

Examples

goto vector *7* if **unconditionally**

goto vector *8* if **available-agents** in split *67 < 5*

goto vector *1* if **digits** *>=14*

goto vector *2* if **digits** *in* table *12*

goto vector *9* if **calls-queued** in split *22 > 5*

Operation

If the command syntax includes **unconditionally**, the command always branches. The unconditional form of the command is useful for applications that require the processing of more than 32 commands. Otherwise, branching takes place according to one of the following conditions:

- The number of available agents in the indicated split meets the constraints defined by the comparator and the threshold value.
- The number of queued calls in the indicated split and at the specified priority level (or higher) meets the constraints defined by the comparator and the threshold value.
- The number of active calls in the indicated VDN meets the constraints defined by the comparator and the threshold value.
- The oldest call-waiting in the indicated split at the specified priority level has been waiting for a period of time within the boundaries defined by the comparator and the threshold value, which is expressed in seconds.
- The number of staffed agents in the indicated split meets the constraints defined by the comparator and the threshold value.

- The digits collected via the collect digits command match the criteria defined by the comparator for the specified digit string.
- The time-of-day criteria are met.

NOTE:
The syntax for this condition can be illustrated by a couple of examples, as follows: *mon 8:01 to fri 17:00* means “anytime between 8:01 a.m. Monday through 5:00 p.m. Friday,” and *all 17:00 to all 8:00* means “between 5:00 p.m. and 8:00 a.m. on any day of the week.”

Answer Supervision Considerations

Call answer is not affected by the command.

Feature Interactions

None.

BCMS Interactions

None.

Messaging

Purpose

Allows the caller to leave a message for the specified extension or the active or latest VDN extension (default).

Syntax

messaging split <split #> for extension <extension>

Valid Entries

split #: *1 through 99* (csi/si), *1 through 600* (r)

extension: extension number, “active,” “latest.” Active is the active called VDN as modified by VDN override rules. Latest is the VDN assigned to the vector in which the call is currently being processed. Active is the default for this field.

Requirements

Basic Call Vectoring software must be installed. Also, the split involved must be an AUDIX split, or a Message Server Adjunct (MSA) split.

Examples

messaging split *18* for extension *2000*

messaging split *45* for extension *active*

Operation

This command causes the caller to be connected to the AUDIX or Message Center split so that the caller may leave a message for the specified extension (call answering service or “mail”).

If the split number specified in the command is a valid message service split (such as an AUDIX or a Message Server Adjunct), and if the extension is either a valid assigned extension or is administered as active or latest, the system attempts to terminate the call to the message service split for call answering service.

If the call is queued to the message service split, or if the call terminates to an available message service agent or AUDIX voice port, the caller is connected to ringback (signifying successful termination), and vector processing terminates. Termination is unsuccessful, and vector processing continues at the next vector step if any one of the following is true:

- Split queue is full.
- AUDIX link is down.
- All AUDIX voice ports are out of service.

If call termination is successful, and if the administered extension (or default VDN) is a message service subscriber, the caller can leave a message for the specified extension.

NOTE:
Agent and/or supervisor stations may be equipped with Automatic Message Wait (AMW) lamps to accommodate the “mail” specified in the *messaging split* command. The lamps can be assigned for VDNs or extensions used to access the messaging split and for which messages are to be left. When messages are left for these VDNs or extensions, the assigned AMW lamps light.

If the extension or VDN is not a subscriber of the message service, one of the following may occur:

- If the message service split is AUDIX, the caller receives ringback until he or she disconnects or is transferred to a default destination.
- If the message service split is a MSA, the caller may be answered by a message service agent, but no message is taken since the specified extension (default VDN) is not a MSA subscriber.

Answer Supervision Considerations

If answer supervision has not already been returned, it is returned when the messaging service port or station is connected to the call (that is, when the call is answered by the port or station).

Feature Interactions

The command can use an AUDIX or MSA hunt group in its operation.

If the command specifies a specific “mailbox” extension, the original principal for a call covered by a VDN is not passed to the adjunct, and it does not appear in the display to the answering agent. The specified extension appears in the display.

If the command is accessed via a direct call to the VDN, and if the mailbox is administered as “active” or “latest,” the corresponding active or latest VDN extension mailbox is sent to the messaging adjunct. Additionally, if the call is sent to a Message Service split, the associated VDN name is sent to the messaging adjunct.

If the command specifies “active” or “latest” as the mailbox extension, the original principal for a call covered to or forwarded to a VDN is used as the default mailbox for the call instead of the “active” or “latest” VDN. Accordingly, the original principal extension and the reason for redirection are passed to the messaging adjunct, and they subsequently appear in the display to the answering agent.

AUDIX does not support mixed length numbering plans.

If the command leaves a message for a VDN or for another messaging service extension, the Automatic Message Waiting Lamp (AMWL) associated with the VDN or extension lights steady.

BCMS Interactions

A call advanced to another position via the command is tracked as an “outflow” in the VDN Report.

Queue-to

Purpose

Unconditionally queues a call to a split and assigns a queuing priority level to the call in case all agents are busy.

Syntax

queue-to split <split #> pri <priority level>

Valid Entries

split #: *1 through 99* (csi/si), *1 through 600* (r)

Requirements

Basic Call Vectoring software must be installed. The split involved must be vector-controlled.

Examples

queue-to split *53* pri *t*

Operation

A call sent with this command either connects to an available agent in the resource specified or enters its queue.

A call may be queued to up to three local splits simultaneously. A call remains queued either until vector processing terminates (via a **disconnect**, **busy**, or **route-to** command, or via a dropped or abandoned call) or until the call reaches an agent. When an agent becomes available in any split to which the call is queued, the following actions take place:

- Call begins ringing the agent.
- Call is removed from any other queues.
- Vector processing terminates.

If the entered split is one of the splits to which the call is already queued, the call is requeued at the new priority level. If the priority level specified is the same as the priority level at which the call is queued, the call remains in the same position in queue. The step is skipped, and vector processing continues at the next step if any of the following conditions are true:

- Desired split's queue is full.
- Call has been previously queued to three different splits.

NOTE:
A *route-to* step to another VDN can be used to remove the call from the splits it is queued to if necessary. The steps in the routed-to vector then can be used to queue to other splits.

Answer Supervision Considerations

Answer supervision is returned (if not already returned) when the call is connected to an answering agent.

Feature Interactions

The **queue-to** command can access an AUDIX/Message Server split in cases where a VDN is assigned as a coverage point. To enable this function, the split must be assigned as a vector-controlled hunt group.

The command is considered a call acceptance vector command whenever one of the following is true:

- Call terminates to an agent.
- Call queues to a split.

The command is considered a neutral vector command when the call neither terminates nor queues.

COR checking is not carried out when a queue-to step places a call to a split.

BCMS Interactions

The total number of calls to the VDN that are queued via the command and then answered by an agent within a specified time period is tracked as "ACD CALLS" in the VDN Report. The average time that calls spend in a vector before being connected via the command as an ACD call to an agent is tracked as "AVG SPEED ANS" in the same report.

Route-to

Purpose

Routes calls either to a destination that is specified by digits collected from the caller or an adjunct (*route-to digits*), or routes calls to the destination specified by the administered digit string (*route-to number*).

Syntax

route-to digits with coverage <option>

route-to number <number> with cov <option> if **unconditionally**

route-to number <number> with cov <option> if **digit** <comparator> <digit>

route-to name1 <number> with cov <option>

route-to name2 <number> with cov <option>

route-to name3 <number> with cov <option>

Valid Entries

number: 1 to 16 digits (includes the Abbreviated Dialing (AD) special characters (*~p, ~w, ~m, ~s, ~W),* ∗, *#.*)

option: *n* (no), *y* (yes)

comparator: =

digit: *0 through 9 or a single "#"*

Requirements

Route-to digits requires Call Prompting software.

Route-to number requires Basic Call Vectoring software.

Route-to name1/2/3 requires the Dial-by-Name feature activated.

Examples

route-to digits with coverage *y*

route-to number *3300* with cov *n* if **unconditionally**

route-to number *473957* with cov *y* if **digit** = *8*

Operation

The **route-to** command attempts to route a call to a set of digits collected from the caller, the network, or from an adjunct, or to route to the destination specified by the administered digit string.

For the **route-to number... if digit** command, the call is conditionally routed to a specified destination according to a single digit entered by the caller. If the digit collected in the last **collect digits** command matches the specified comparison in relation to the administered digit, the command attempts to route the call to the specified destination.

The destination for a route-to command can be any of the following:

- Internal extension (for example, split/hunt group, station, and so on)
- VDN extension
- Attendant or Attendant Queue
- Remote extension (UDP)
- External number, such as a TAC or AAR/ARS FAC followed by a public or private network number (for example, 7-digit ETN, 10-digit DDD, and so on)
- Remote Access Extension.

NOTE:
The VDN's Class of Restriction (COR) is used for calling permissions.

The **route-to digits** command fails if no digits are collected, and vector processing continues at the next vector step.

The **route-to number... if digit** command fails if more than one digit is collected or if the digit comparison fails. Vector processing continues at the next command.

The **route-to number... if interflow-qpos** command fails if the call is not in the eligible queue established by the *interflow-qpos* condition. Vector processing continues at the next command.

If the **route-to** command is successful, vector processing terminates. Otherwise, vector processing continues at the next vector command.

A *route-to* step in a vector is treated as cov=n for a covered call regardless of the cov setting on the **route-to** command.

If the number expressed in the command is a system extension or an attendant group (and not a VDN), the system considers the step successful if one of the following conditions occurs:

- The endpoint is rung.
- The endpoint has Call Forwarding or night service (hunt group) enabled, and the (night service) destination forwarded to is rung; or, if off-premises Call Forwarding (UDP hunt night service), a trunk is seized.

The system then provides ringback to the caller, and vector processing terminates. However, if the call cannot complete successfully (for example, no idle appearance is available), vector processing continues at the next vector command.

If the number is a VDN extension, the following events occur:

- Vector processing terminates within the current vector.
- If the current VDN is administered with override, the new VDN overrides current VDN information.
- Processing of the vector associated with the VDN extension begins.

If the number is an AAR/ARS FAC plus digits, or if it is a remote UDP extension, standard AAR/ARS processing is performed to select the trunk group and outpulse the digits. If a trunk is seized, vector processing terminates, and the calling party hears feedback provided by the far end. Otherwise, the call cannot complete successfully (because no trunks are available, the FRL/COR is restricted, and so on), and vector processing continues at the next vector command.

If the number is a TAC plus digits, and a trunk is seized, vector processing terminates, and the calling party hears feedback provided by the far end. Otherwise, the call cannot complete successfully (because no trunks are available, the COR is restricted, and so on), and vector processing continues at the next vector command.

If the number is any other number (such as an FAC other than AAR/ARS), the command is unsuccessful, and vector processing continues at the next vector command.

Abbreviated Dialing special characters can also be used in the number field. Each of these characters instructs the system to take a different action when dialing reaches the point where the character is stored. The characters are as follows:

- ~*p* (pause)
- ~*w* (wait)
- ~*m* (mark)
- ~*s* (suppress)
- ~*W* (indefinite wait).

Each special character counts as two digits towards the maximum. The maximum number of digits for the command is 16.

The **route-to digits** command can be used to implement an automated attendant function.

Coverage

The optional coverage parameter determines whether coverage should apply during routing. If coverage applies, and if the digits entered are valid, the following occurs:

- Ringback is provided.
- Vector processing terminates.
- Normal termination and coverage are implemented.

NOTE:
For detailed information about the operation of the **route-to** command with or without coverage for the different destinations see Table 23 on Page 183.

Answer Supervision Considerations

Generally, answer supervision is provided when the destination answers the call. The exception to this involves incoming trunk calls routed to another non-ISDN-PRI trunk. Such calls provide answer supervision when the outgoing trunk is seized.

Feature Interactions

When COR checking is applied to a *route-to number* or *route-to digits* step, it is the COR of the latest VDN that is used.

The **route-to** command may specify the AAR or ARS access codes. The COR associated with the latest VDN is used to determine the Partitioned Group Number (PGN) time-of-day routing chart. The PGN determines the choice or route tables used on a particular call.

The command may call the AUDIX extension. If this happens, the call is treated as a direct call to AUDIX, and the calling party may retrieve his or her messages.

If the call covers to a VDN, the command supports a remote AUDIX interface to a local hunt group extension that is assigned as a remote AUDIX hunt group. The "remote AUDIX hunt group" (which has no members and cannot be vector-controlled) forwards the call to the remote AUDIX destination in the same manner as when the hunt group is assigned as a point in the coverage path.

If the command is directed to a station with bridged appearances, the bridged appearance button lamps are updated.

The following destinations always result in a failure, and vector processing continues at the next step:

- Controlled trunk group
- Code calling FAC
- Facility test call
- TAAS access code
- Priority access code
- Loudspeaker paging access code
- Station Message Detail Recording (SMDR) account code
- Voice message retrieval access code.

If the command is executed and Direct Outward Dialing (DOD) is in effect, the COR of the latest VDN is compared with the COR of the called facility to determine if the call is permitted. If access is not permitted, the command fails and vector processing continues. In the case where a COR requiring the entry of account codes is assigned to a VDN, and the command is executed by the associated vector, the command is unsuccessful, and vector processing continues at the next step.

The individual extension number assigned to an attendant console can be used as the command's argument.

A call processed by the command can wait in the individual attendant queue and is subsequently removed from vector processing.

The command can access both public and private networks.

If the command dials the attendant, and if the system is in night service, the call routes to the DID Listed Directory Number (LDN) night destination.

The command can place AAR/ARS calls that implement subnet trunking, which is the routing of calls over trunk groups that terminate in switches with different dial plans.

Authorization codes are disabled with respect to routing via VDNs. In other words, if authorization codes are enabled, and a **route-to** command in a prompting vector accesses AAR or ARS, and the VDN's FRL does not have the permission to utilize the chosen routing preference, no authorization code is prompted for, and the **route-to** command fails.

If the command routes the call without coverage to a display station, the station displays the following: "a = Originator Name to VDN Name."

If the command calls a station that is a member of a pickup group, the call can be picked up by another pickup group member.

The command is considered a neutral vector command whenever one of the following is true:

- Termination is unsuccessful.
- Trunk is not seized.

For a call that covers or forwards to a VDN, the **route-to with coverage y** command functions the same as the **route-to with coverage n** command. For a covered or forwarded call, the coverage option for the command is disabled since such a call should not be further redirected.

A *route-to with cov y* to a station that has call forwarding activated is forwarded.

Service Observing can be initiated with Call Vectoring using the **route-to** command. See "Service Observing" on Page 50 for detailed instructions.

NOTE:
Appendix F gives a detailed description of the feature interactions for the *route-to* number with and without coverage command.

BCMS Interactions

A call advanced to another position via the command is tracked as "outflow" in the VDN Report. A call answered by an attendant via the command is also tracked as "outflow."

Stop

Purpose

Halts the processing of any subsequent vector steps.

Syntax

stop

Requirements

Basic Call Vectoring or Call Prompting software must be installed.

Operation

After the **stop** command is processed, any calls already queued remain queued, and any wait treatment (for example, silence, ringback, music) is continued. Any calls not queued are dropped under the same scenario.

If a tone detector is allocated to the call, and if the **stop** command is encountered, the tone detector is disconnected. However, current call processing continues (that is, the call is not dropped). The caller continues to hear the feedback that was provided before the *stop* command was encountered.

NOTE:
An implicit stop is processed following the last administered command in a vector.

Answer Supervision Considerations

The command has no effect on answer supervision.

Feature Interactions

None.

BCMS Interactions

None.

Wait-time

Purpose

Delays the processing of the next vector step if a specified delay time is included in the command's syntax. Also provides feedback (in the form of silence, ringback, or music) to the caller while the call advances in queue.

Syntax

wait-time <seconds> secs hearing <treatment>

wait-time <minutes> mins hearing <treatment>

wait-time <hours> hrs hearing <treatment>

Valid Entries

NOTE:
R6 systems support a wait time in seconds only. R7 systems support a wait time in seconds, minutes, or hours.

seconds (R6): *0 through 998* (2-second increments); *0 through 8* when using the i-silent treatment.

seconds (R7): *0 through 999* (1-second increments); *0 through 8* when using the i-silent treatment.

minutes (R7): *0 through 480* (1-minute increments).

hours (R7): *0 through 8* (1-hour increments).

treatment: silence, ringback, music, i-silent.

When music is indicated as a treatment, it refers to the system music, not an alternate music source.

extension: The valid extension number of an alternate audio/music source.

Requirements

Basic Call Vectoring and Call Prompting software must be activated. Also, a music-on-hold port must be provided for the music treatment.

Examples

wait-time *85* secs hearing *music*

wait-time *12* mins hearing *54795* then *continue*

wait-time *2* hrs hearing *music*

Operation

The specified feedback is given to the caller, and vector processing waits the specified time before going on to the next step. If the time specified is *0*, feedback is provided without any delay in the processing of the next vector step. The feedback given to the caller continues until any one of the following occurs:

- Subsequent vector step (containing *wait-time* or *announcement*) changes the treatment.
- Vector processing encounters a *disconnect* or *busy* command.
- Call is routed to another location or to a step that includes an announcement (for example, *collect digits*).
- Call is routed to another VDN.
- Call is delivered to a destination (starts ringing at an agent's terminal).

Answer Supervision Considerations

If the *music* or audio source treatment is included in the command, answer supervision is triggered. If the command is encountered and answer supervision was sent previously, the caller hears the treatment specified in the current command. If, for a CO trunk user, the command with *silence*, *ringback*, or *i-silent* treatment is encountered prior to answer supervision, the caller continues to hear ringback from the CO.

Feature Interactions

When the command is implemented with music as the treatment, the system-wide music-on-hold feature must be administered. Otherwise, the caller hears silence.

Feedback continues while a subsequent vector step queues for an announcement or for a tone detector.

NOTE:
An implicit wait of 0.2 seconds (with no change in the feedback to the caller) is provided after every seven vector steps if one of these steps does not suspend vector processing. (The following steps, if successful, do not suspend vector processing: *queue-to split*, *check split*, *goto step*, *goto vector* and *wait-time 0 seconds*. The following steps, if unsuccessful, also do not suspend vector processing: *check split*, *route-to*, and *messaging split*. The only commands that suspend vector processing are the following: *announcement*, *wait-time > 0 seconds*, *collect digits*, and *converse-on split*.

BCMS Interactions

The command is not tracked on BCMS.

Criteria for Success/Failure of Call Vectoring Commands

Table 12 summarizes the success and failure criteria for various vector commands. Before you write or evaluate vectors, it is important to understand the information in this table.

Table 12. Call Vectoring Command Success/Failure Criteria

Command	Success/Failure Criteria	Vector Processing Disposition
announcement	Fails if specified announcement is unadministered, not recorded, or busied out. Otherwise, succeeds.	Continue vector processing with the next sequential step. Play the announcement, then continue at the next sequential step.
busy	Always succeeds.	Exit vector processing, then play the busy tone for 45 seconds before dropping the call. (Unanswered CO trunk calls receive 45 seconds of ringback.)

Continued on next page

Table 12. Call Vectoring Command Success/Failure Criteria — *Continued*

Command	Success/Failure Criteria	Vector Processing Disposition
check split	Fails if any of the following are true: ■ Vector conditional is false. ■ Split's queue is full. ■ Split is not vector-controlled. ■ Call is already queued at the specified priority to the specified split. ■ Call is already queued to three different splits.	Continue vector processing with the next sequential step.
	Otherwise:	
	Succeeds, and the call is terminated to an agent.	Exit vector processing, and pass control to call processing.
	Succeeds, and the call is queued or requeued in the specified split at the specified priority.	Continue vector processing with the next sequential step.
collect digits	Fails if any of the following are true:	
	■ Call originates from an outside caller who is not using a touch-tone telephone.	Call Prompting timer takes effect, command times out, and vector processing continues at the next vector step.
	■ No tone detector is in the system, or the tone detector queue is full.	Continue vector processing at the next step.
	■ Caller enters fewer digits than the maximum specified.	Call Prompting timer takes effect, command is terminated, and any digits collected prior to the timeout are available for subsequent processing.
	Otherwise, succeeds.	Continue vector processing at the next step.

Continued on next page

Table 12. Call Vectoring Command Success/Failure Criteria — *Continued*

Command	Success/Failure Criteria	Vector Processing Disposition
converse-on split	Fails if any of the following are true: ■ Converse split queue is full. ■ Converse split is not vector-controlled. ■ Auto-available split is in effect, and all agents are logged out by Redirection on No Answer.	Continue vector processing with the next sequential step.
	Otherwise: Succeeds, call is delivered to the converse split, and (if administered) digits are outpulsed to the VRU. The caller is connected to the VRU, the voice response script is executed, and (if necessary) digits are outpulsed to the switch.	Continue vector processing with the next sequential step.
disconnect	Always succeeds.	Play the announcement (if specified). Then drop the call.
goto step	Fails if the step condition is not met.	Continue vector processing with the next sequential step.
	Succeeds if the step condition is met.	Continue vector processing with the destination step.
goto vector	Fails if the step condition is not met.	Continue vector processing with the next sequential step.
	Succeeds if the step condition is met.	Continue vector processing with the first nonblank step of the destination vector.

Continued on next page

Table 12. Call Vectoring Command Success/Failure Criteria — *Continued*

Command	Success/Failure Criteria	Vector Processing Disposition
messaging split	Fails if any of the following are true: ■ Specified split is not an AUDIX split or a Message Server Adjunct (MSA) split. ■ Specified extension is invalid. ■ Messaging split queue is full. ■ Messaging split is not vector controlled and has no working agents (none logged in or all in AUX work mode). ■ Communications link with the messaging adjunct is inaccessible.	Continue vector processing with the next sequential step.
	Otherwise, succeeds.	Terminate vector processing.

Continued on next page

Table 12. Call Vectoring Command Success/Failure Criteria — *Continued*

Command	Success/Failure Criteria	Vector Processing Disposition
queue-to split	Fails if any of the following are true: ■ Split's queue is full. ■ Split is not vector-controlled. ■ Call is already queued at the specified priority to the specified split. ■ Call is already queued to three different splits. Otherwise:	Continue vector processing with the next sequential step.
	Succeeds, and the call is terminated to an agent.	Exit vector processing, and pass control to call processing.
	Succeeds, and the call is queued or requeued in the specified split at the specified priority.	Continue vector processing with the next sequential step.
stop	Always succeeds.	Exit vector processing. Control is passed to normal call processing. Any queuing or treatment in effect remains in effect. Call is dropped if not queued.
wait-time	Always succeeds.	Connect the specified treatment and pass control to the delay timer. Any feedback is continued until other feedback is provided.

NOTE:
Complete operational details for the **route-to** commands are provided in Appendix F.

Call Vectoring Management

B

Call Vectoring management involves a number of different considerations and tasks. This appendix describes these considerations/tasks. Specifically, the following topics are discussed:

- Call Vectoring feature requirements
- Enabling the vector disconnect timer
- Upgrading to a Call Vectoring environment
- Changing the vector
- Testing the vector.

Call Vectoring Feature Requirements

The tables appearing on the next several pages indicate the forms and the hardware required for implementing each of the Call Vectoring features. Details on completing the forms can be found in the *DEFINITY® ECS Administrator's Guide*.

Table 13. Basic Call Vectoring Requirements

Feature	Form(s)	Hardware
Basic Call Vectoring	■ Vector Directory Number Form ■ Hunt Group Form ■ Call Vector Form ■ Feature Related System Parameters Form	Announcement capabilities require either: ■ TN750 Integrated Announcement circuit pack(s), **or** ■ External announcement facility (analog announcements). Also, each analog announcement requires a port on an analog line circuit pack. See the *DEFINITY® ECS System Description* for a list of available analog circuit packs.

NOTE:
The Basic Call Vectoring option must be enabled on the System-Parameters Customer-Options form before the associated forms and the fields on the forms can be administered.

NOTE:
The TN750 Integrated Announcement circuit pack provides 16 ports for listening to announcements. The system provides for the installation of multiple TN750C Integrated Announcement circuit packs. See “Managing Announcements” in the *DEFINITY® ECS Administrator’s Guide* for more details.

Table 14. Call Prompting Requirements

Feature	Form(s)	Hardware
Call Prompting	■ Vector Directory Number Form ■ Hunt Group Form ■ Call Vector Form	Announcement capabilities require either: ■ TN750 Integrated Announcement circuit pack(s), **or** ■ External announcement facility (analog announcements). Also, each analog announcement requires a port on an analog line circuit pack. See the *DEFINITY® ECS System Description* for a list of available analog circuit packs.

NOTE:
The Basic Call Vectoring and Call Prompting option(s) must be enabled on the System-Parameters Customer-Options form before the associated forms and the fields on the forms can be administered.

Enabling the Vector Disconnect Timer

Call Vectoring makes available a Vector Disconnect Timer, which can be set for any amount of time between 1 and 240 minutes inclusive. The timer is enabled by selecting the timer field in the Feature-Related System-Parameters form. The timer is started when vector processing is started. Once the timer runs out, the call is dropped. The timer is canceled when vector processing terminates.

Enabling the timer allows queued calls that have not been answered within a determined amount of time to be dropped. For more information, refer to *DEFINITY® ECS Administrator's Guide*.

Upgrading to a Call Vectoring Environment

If you are already equipped with ACD and want to use Call Vectoring, the ACD environment must be upgraded to a Call Vectoring environment. This involves installing VDNs, vectors and hunt groups for the desired Call Vectoring feature(s).

The set of guidelines that follows is intended to serve as a general procedure for upgrading to a Call Vectoring environment. For complete details of this process, refer to *DEFINITY® ECS Administrator's Guide*.

1. Verify the vector options on the Customer Option Form.
2. Add the VDNs.
3. Evaluate the number of queue slots assigned to each split. Usually, you want to assign enough queue slots to allow all calls processed by Call Vectoring to be queued. (See the considerations for Basic Call Vectoring in Appendix C for more details.)
4. Change hunt-groups to be vector-controlled.
5. Administer the vectors and at least one test hunt group.
6. Test all of the vectors to be installed.
7. Change the trunk groups, night destinations, and so on, to use the VDNs.

Changing and Testing the Vector

Vectors currently being used to process calls should not be changed because changes would have an immediate and uncertain effect on the treatment that the calls are receiving. Instead, a new vector should always be written.

In testing the vector, you should not consider the entire vector at once. Rather, you should first figuratively divide the vector into portions, then test each of these portions until the entire vector is tested.

After the new vector is thoroughly tested, the vector should be brought into service by changing the VDN to point to the new vector.

The set of following guidelines is intended to serve as a general procedure for changing and testing vectors. For complete details of this process, refer to *DEFINITY® ECS Administrator's Guide*.

1. Check that a current version of the translation data is available.
2. Create a new VDN that points to the new vector. This VDN, which is temporary, is necessary to test the new vector.
3. Administer the new vector. Vector commands should be added and tested, one command at a time, starting with the first command. Be sure that each line is correct before proceeding to the next one.
4. Test the new vector with the new VDN. This ensures the new vector will function correctly when the vector is installed.
5. Install the new vector by changing the old VDN's vector assignment so that the VDNs now point to the new vector. Calls that are already being processed by the old vector will continue to be handled by that vector until the vector terminates vector processing.
6. Once all the calls are handled, remove the old vector and the VDN that was used for testing.

Considerations for the Vectoring Features

C

This appendix contains several lists of considerations you should bear in mind when using the Call Vectoring features. These considerations are intended to help you get the highest degree of productivity from Call Vectoring.

Basic Call Vectoring Considerations

The following are considerations you should keep in mind when working with Basic Call Vectoring:

- Make the split queues large enough so that all incoming calls queue and are not dropped. If a queue is too small, a **queue-to split** or a **check split** command might fail to queue a call due to a lack of available queue slots. Accordingly, it is also always a good practice to include in the vector a step that checks a split's queue before queuing occurs and a corresponding step that provides alternate treatment if the queue is full. To check the queue size, you can use a **goto** command (for example, *goto Step 5 if calls-queued in split 20 pri l > 30*). The alternate treatment, which, if needed, is usually accessed by the **goto** command that checks the queue size, can queue the call to a backup split, provide a busy signal, and so on.
- A default treatment or a *route-to* destination step should be supplied after a **route-to** command in case the first destination is unavailable.
- Calls should not be queued to an unstaffed split (unless this is intended by the customer) without some alternate treatment.
- Interflow calls should not be permitted to interflow back and forth between a remote switch vector and a local switch vector. This process could cause a single call to use up all available trunks.

- After an announcement is provided, the audible feedback (such as music) should be re-attached.
- For ease-of-use purposes, each specific vector function or operation should be included in a separate vector and linked via one or more **goto vector** commands.
- In creating a vector, commands can be chosen and arranged in a manner such that answer supervision is delayed as long as possible. This should be done to keep down the service cost.
- The caller should always be provided with initial feedback (usually ringback).
- Direct agent calls merit special attention because such calls can affect call queuing. Although direct agent calls take up a queue slot, they are not always reported as using such a slot on BCMS reports (discussed in Appendix E). For example, a direct agent call is never counted toward the total of queued calls within a split (that is, the *calls-queued* test condition has no effect on this type of call).
- If it is necessary for a caller to hear an entire voice response script before talking to an agent, the caller should not be queued until after the *converse-on* step is executed.
- Audible feedback should be provided prior to a *converse-on step* whenever a large number of digits are to be outpulsed to the VRU.

Call Prompting Considerations

The following list includes considerations you should keep in mind when working with Call Prompting:

- To enter the digits requested via a **collect digits** command, outside callers must have a touch-tone telephone. For such callers using rotary dialing, a 10 second inter-digit timeout takes effect, and the **collect digits** command is omitted. As a precaution, a default treatment (for example, **route-to** attendant command, **queue-to split** command) should always be provided in the vector script unless the script is created exclusively for users of touch-tone telephones.
- If a caller does not enter the full number of digits specified in a *collect digits* step, an administered timeout occurs. Thereafter, vector processing continues with subsequent vector steps, and an attempt is made to process the call using the digits that have been collected. If the digits entered do not represent a valid destination, and if Automated Attendant is being implemented via a **route-to digits** command, the **route-to digits** command fails, and vector processing continues at the next step, which should be a default treatment.
- It may be prudent to take steps in case a **route-to** attendant command fails, such as providing a disconnect announcement.
- From time to time, all of the system's touch-tone receivers might be in use. As a result, you should avoid starting your main vector with a **collect digits** command, since the caller on a DID or tie trunk in this case receives no audible feedback if he or she has to wait for a receiver to become available. Accordingly, it is a good practice to include some treatment (for example, a *wait-time 0 seconds hearing ringback* step) before the initial *collect digits* step.

Transferring Calls to VDNs Considerations

Care needs to be taken when writing a vector to which callers will be transferred.

To understand why care is needed, it is necessary to understand how a transferred call is treated. There are three main steps in a call transfer.

1. The transferring party presses the transfer button. The caller is put on hold. A second call is created with the transferring party as the originator.
2. The transferring party dials the VDN extension. Vector processing starts. The transferring party, not the caller, hears the initial vector provided feedback, if any.
3. The transferring party presses the transfer button for the second time. The two calls merge. The transferring party is dropped from the call. The caller becomes the originator of the new call. The caller now begins to receive vector provided feedback.

Between transfer steps 2 and 3 there is always a small but finite amount of time during which it is the transferring party who is connected to the vector. Insert a delay of sufficient length to allow the transferring party to complete the transfer.

A delay is not required before a *collect x digits after announcement* step because a collect announcement is restarted for the caller when the transfer is complete.

Feature Capacities

With DEFINTY BCS and GuestWorks, the Call Vectoring and ACD feature capacities are a subset of the Call Vectoring and ACD features as supported on the DEFINITY ECS offer. The following tables illustrate the feature capacities for each of the features based on the different switch types.

Table 15. Call Vectoring Feature Capacities

Item	R6vs/csi/si	R6r	R7csi/si	R7r
	BCS/GuestWorks Issue 4		BCS/GuestWorks Issue 5	
Priority levels	4	4	4	4
Recorded announcements/analog sources for vector delay	128	256	128	256
Steps per vector	32	32	32	32
VDNs	4	8	10	20
Vectors per system	4	8	10	20
Number of immediately-collected digits for Call Prompting	16	16	16	16
Number of dial-ahead digits for Call Prompting	24	24	24	24

Table 16. ACD Feature Capacities

Item	R6vs/csi/si	R6r	R7csi/si	R7r
	BCS/GuestWorks Issue 4		BCS/GuestWorks Issue 5	
Logged-in ACD agents	150	150	150	150
Logged-in splits per agent	4	4	4	4
Announcements per split	2	2	2	2
Announcements per system	128	256	128	256
Queue slots per group	200	999	200	999
Queue slots per system	1500	15000	1500	15000
Splits	99	600	99	600
ACD members per split	200	999	200	999
Split members per system	150	150	150	150
Measured agents	20	20	25	25

Call Vectoring Features Not Supported

The following DEFINITY ECS Call Vectoring features are not supported with DEFINITY BCS or GuestWorks.

- G3V4 Enhanced (for example, specifying a priority level with the oldest-call-wait condition)
- Advanced Vector Routing
- ANI/II Digits Routing
- Call Information Forwarding (CINFO)
- Best Service Routing™
- Adjunct Routing
- VDN of Origin Announcement
- VDN Return Destination.

Troubleshooting Vectors

D

This appendix serves as a troubleshooting guide for Call Vectoring. The first part of the appendix includes two tables that indicate and explain unexpected operations within Call Vectoring that the customer may encounter. The first table focuses on the Call Vectoring features, while the second table focuses on the Call Vectoring commands. The second part of the appendix contains a table that focuses on **converse-on** command debugging. Finally, the third part of the appendix contains procedures for tracking many of the unexpected operations within Call Vectoring that are discussed in the two tables.

Unexpected Feature Operations

Table 17 indicates and explains unexpected operations within Call Vectoring that you may encounter.

NOTE:
For solutions to these unexpected operations, refer to Chapter 3, Chapter 4, Appendix A, Appendix C, and Appendix F.

Table 17. Unexpected Feature Operations

Feature/Area	Customer Observations	Causes
General Vector Processing	Vector stuck.	1000 steps executed.
		No default treatment in the vector.
	Audible feedback lasts longer than the delay interval.	Last vector step.
		Queuing for an announcement.
		Queuing for a touch-tone receiver for a *collect digits* step.

Unexpected Command Operations

Table 18 indicates and explains the unexpected operations the customer may encounter in using the Call Vectoring commands.

Table 18. Unexpected Command Operations

Command Step	Customer Observation(s)	Cause(s)
announcement	Announcement not heard.	Announcement board not present.
		Announcement not administered.
		Announcement not recorded.
		Announcement being rerecorded.
		All ports busied out.
		Announcement restore in progress.
		Link to TN750 down.
	Extra delay before hearing announcement.	Announcement queue full.
		All integrated announcement ports busy.
		Analog announcement busy.
	Vector processing stops.	Analog announcement does not answer.
	Listening to silence after announcement.	Announcement is the last step.
	Incomplete announcement.	Agent becomes available.
busy	Ringback heard instead of busy tone.	Unanswered CO trunk.
check	Call does not enter queue or terminate to agent.	Step condition not met.

Continued on next page

Table 18. Unexpected Command Operations — *Continued*

Command Step	Customer Observation(s)	Cause(s)
check and **queue-to**	Call does not enter queue or terminate to agent.	Queue length specified on the hunt group screen has been exceeded.
		Invalid split.
		Split not vector-controlled.
		Already queued to three different splits.
		No queue.
		Queue or check status indicates space when queue is full due to direct agent calls.
	Call apparently answered in wrong order.	Call being requeued at different priority.
		Call superseded by higher priority call, including direct agent call.
collect digits	Announcement not heard while waiting for digits, but network billing indicates that the call was answered.	Announcement board not present.
		Announcement not administered.
		Announcement not recorded.
		Announcement being rerecorded.
		All ports busied out.
		Announcement restore in progress.
		Dial ahead digit exists.
	Collect step and announcement skipped.	Tone detector not in system.
		Link to PN that has tone detector is down.
		Tone detector queue full.

Continued on next page

Table 18. Unexpected Command Operations — *Continued*

Command Step	Customer Observation(s)	Cause(s)
collect digits (Continued)	Delay before hearing announcement.	All tone detector ports busy, but space in queue.
		Announcement queue full.
		All integrated announcement ports busy.
		Analog announcement busy.
	Vector stuck.	Analog announcement does not answer.
	Dial-ahead digits not recognized.	Dial-ahead digits entered prior to first collection step.
		Call has been transferred.
		Tone detector has been released.
		24 digits have already been provided.
		Call Prompting timeout since the last digit was entered.
	Vector processing halted at collect step; announcement heard again upon return.	Call put on hold, transferred, or conferenced.
	Insufficient digits collected; call routed to intercept.	Caller dialed # too soon.
		Caller dialed * without reentering correct digits.
		Call Prompting interdigit time-out.
	Caller information button denied.	No digits were collected.
		Display not in Normal mode.
	Collect announcement not heard and first collected digit incorrect.	System does not contain up-to-date tone detectors.
	Incomplete announcement.	Agent becomes available.
		First digit dialed.

Continued on next page

Table 18. Unexpected Command Operations — *Continued*

Command Step	Customer Observation(s)	Cause(s)
converse-on split[1]	VRU script not executed.	Queue full. No queue. Invalid split. Split not vector-controlled. VRU down.
	No data returned from VRU.	No tone detectors available on the switch.
	VRU script terminated prematurely.	Agent becomes available. VRU script attempted to transfer the call.
	Wait digits not passed.	Call not queued or no working agents in splits where call is queued.
disconnect	Announcement not heard.	Announcement board not present.
		Announcement not administered.
		Announcement not recorded.
		Announcement being rerecorded.
		All ports busied out.
		Announcement restore in progress.
	Extra delay.	Announcement queue full.
		All integrated announcement ports busy.
		All analog announcements busy.
	Vector stuck.	Analog announcement does not answer.
goto step	Branch is not made to the specified step.	Step condition not met.
		System time not set.
goto vector	Branch is not made to the specified vector.	Step condition not met.
	Vector stuck.	Goto vector with no steps or with all failed steps.

Continued on next page

Table 18. Unexpected Command Operations — *Continued*

Command Step	Customer Observation(s)	Cause(s)
messaging split	Vector stuck (with ringback).	Extension unknown to AUDIX.
	Step skipped, no message left.	AUDIX link down.
		Queue for AUDIX voice ports is full.
	Vector stuck (with busy).	Remote AUDIX link down.
	Messages not found.	Message extension is *none* (message is left for VDN that accessed the vector).
	Delay before AUDIX answers.	All AUDIX ports busy, but space in queue.
	Busy tone.	Queue for AUDIX voice ports is full.
	Step skipped.	Split not AUDIX split anymore.
		Queue for AUDIX voice ports is full.
	Vector stuck (with busy).	Remote AUDIX link down.

Continued on next page

Table 18. Unexpected Command Operations — *Continued*

Command Step	Customer Observation(s)	Cause(s)
route-to[2]	Step skipped (that is, default treatment).	Invalid local extension.
		No trunks available.
		COR/FRL restricted.
		Digit string inconsistent with networking translation.
		Busy local destination (route to digits without coverage and route to number).
		No digits collected.
		Step condition not met.
	Network reorder.	Digit string inconsistent with public network translation.
	Intercept or reorder tone heard.	Vector processing succeeded routing off switch, but a problem has occurred before routing to its final destination.
	All trunks busy on a quiet system.	Two switches treating each other as a backup switch.
stop	Call dropped.	Call not queued when vector processing stops.

Continued on next page

Table 18. Unexpected Command Operations — *Continued*

Command Step	Customer Observation(s)	Cause(s)
wait-time	Audible feedback longer than delay interval.	Queuing for an announcement or for a tone detector.
		Stop command executed.
	Audible feedback shorter than delay interval.	Agent becomes available.
	Music not heard.	No music port administered.
		Music source disconnected or turned off.
	Alternate audio/music source not heard.	Announcement board not present.
		Audio/Music source not administered.
		Audio/Music source not recorded.
		Audio/Music source being rerecorded.
		All ports busied out.
		Announcement restore in progress.

1. Refer to the "Converse-on Command Debugging" section later in this appendix for more details on converse-on command debugging.
2. Complete operation details for the route-to commands are presented in Appendix F.

Converse-on Command Debugging

Table 19 helps your troubleshooting efforts with the **converse-on** command.

NOTE:
Refer to Appendix G for details on the call flow for converse-VRI calls.

Table 19. Converse Command Debugging

SYMPTOM	CAUSES	EVIDENCE
PLACING A CALL:		
Converse step skipped.	VRU down (Redirection on No Answer).	Vector event.
	Split queue full	Vector event.
Call stuck in converse.	VRU port does not answer, Redirection on No Answer not used.	Check split administration.
	VRU down, Redirection on No Answer leaves call in queue.	Check split status.
DATA PASSING:		
First set of digits not collected.	Converse first delay too short.	Check administration.
	No digits collected.	Vector event.
	Call not queued (qpos).	Vector event.
	VRU timed out awaiting first digit.	VRU error log/trace.
	VRU first digit timeout too short.	Check VRU script. Check converse first data delay.
	Faulty hardware.	Diagnostics

Continued on next page

Table 19. Converse Command Debugging — *Continued*

SYMPTOM	CAUSES	EVIDENCE
Second set of digits not collected.	VRU digit count on first prompt in VRU script does not include "#."	Check VRU script.
	Converse second delay too short.	Check administration.
	No digits collected.	Vector event.
	Call not queued (qpos).	Vector event.
	VRU timed out awaiting first digit.	Vector Event
	VRU error log/trace.	
	VRU first digit timeout too short.	Check VRU script. Check converse second data delay.
	Inter-digit timeout too short on first prompt and collect.	Check VRU script.
	Faulty hardware.	Diagnostics.
Digits incomplete.	Converse data delay too short.	Check administration.
	Faulty hardware.	Diagnostics.
Second set of digits is the same as the first digits passed.	VRUs first prompt timed out.	Check administration.
	Faulty hardware.	Diagnostics.
DATA RETURN:		

Continued on next page

Table 19. Converse Command Debugging — *Continued*

SYMPTOM	CAUSES	EVIDENCE
No digits returned to the switch.	Flash not recognized by switch.	VRU error log/trace.
		Check flash timing on VRU.
	Converse data return FAC not administered.	Check administration.
	VRU does not return FAC.	VRU script. Transfer attempt vector event.
	VRU returns incorrect FAC.	VRU script. Transfer attempt vector event.
	Digit timeout during FAC.	Transfer attempt event.
	Converse data return FAC overlaps with other entries in the dial plan	Check dial plan.
	Faulty hardware.	Diagnostics.
	Digit timeout after FAC.	None unless VRU logs being dropped by the switch.
Not all digits returned to the switch.	Overflow of Call Prompting buffer	Vector Event.
	Faulty hardware.	Diagnostics.
Collect announcement not heard.	Too many digits returned by VRU.	Check VRU script.
	Faulty hardware.	Diagnostics.

Tracking Unexpected Vector Events

If you have an System Administration Terminal (SAT), you can display unexpected vector events. A vector event is an error that results from resource exhaustion or from faulty vector programming, rather than from a switch software error. For example, failures involving the **route-to** command are usually due to an invalid extension entered by the user.

By displaying vector events, you can do the following:

- Diagnose and correct each Call Vectoring problem, as indicated by its corresponding vector event, and thereby
- Eliminate the need for a technician to make on-site visits to do the same.

The following sections explain how you can troubleshoot by tracking unexpected vector events.

Display Events Form

The first step is to initiate the display of vector events. You do this by entering the **display events** command at the `enter command` prompt.

Once the command is entered, the `display events` form appears on the screen as in Screen 43.

```
display events                                          Page 1 of 1    SPE B
                          EVENT REPORT
        The following option control which events will be displayed.
           EVENT CATEGORY
                        Category:  Vector
           REPORT PERIOD
                      Interval: _a_   From: __/__/__:__  To: __/__/__:__
           SEARCH OPTIONS
                  Vector Number: ___
                     Event Type: ____
```

Screen 43. Layout of Display Events Form

The following list indicates the options on the form, comments on these options, and also discusses the field(s) within each option.

- **EVENT CATEGORY**. This option is intended to indicate the class of logged events to be displayed. For our purposes, the default value *Vector* automatically appears in this display-only `Category` field. The value *Vector* indicates that only vector events will be displayed.
- **REPORT PERIOD**. This option allows you to specify a report period. This period consists of an `Interval` field, a `From` date/time stamp, and a `To` date/time stamp. Valid entries for the `Interval` field include *(h)our*, *(d)ay*, *(w)eek*, and *(a)ll*. Both stamps consist of a series of numbers that represent a period of time, as follows: *1 through 12* (month), *1 through 31* (day), *0 through 23* (hour), *0 through 59* (minutes). If the field and stamps are populated, only the vector events that occurred within report period specified are displayed. Otherwise, all vector events are displayed regardless of when they occurred.
- **SEARCH OPTIONS**. This option contains two fields, `Vector Number` and `Event Type`.

  `Vector Number` allows you to specify a vector number. If this field is populated, only vector events that are associated with this vector number are displayed. Otherwise, all vector events are displayed regardless of the vector number with which they are associated.

  `Event Type` allows you to specify the number associated with a particular type of vector event. This number may range from *0* to *999*. If the `Event Type` field is populated, only vector events of the type indicated are displayed. Otherwise, all vector events are displayed regardless of type.

Display Events Report

After you complete the Display Events form, you can generate the Display Events Report by submitting the display request and pressing the Enter key a second time. A sample report appears in Screen 44.

```
                                    EVENTS REPORT
Event  Event                 Event    Event   First        Last         Event
Type   Description           Data 1   Data 2  Occur        Occur        Cnt
  20   Call not queued       12/5     B       09/28/13:43  09/28/13:43   21
 541   Not a messaging split Split 89 4C      09/28/13:43  09/28/13:43  136
```

Screen 44. Display Events Report

The Display Events Report provides details of all the logged vector events that meet the selection criteria supplied by the user. The following list identifies and discusses the fields in the report.

- `Event Type` contains a unique number between *0* and *999* that identifies the type of vector event that occurred.
- `Event Description` contains text that describes the vector event.
- `Event Data 1` is a 9-character fields that contains data in one of two formats:
  - <*number1*>/<*number2*> (for example, *12/5*), where <*number1*> is the vector number associated with the vector event, and where <*number2*> is the step number associated with the vector event. This format is used for events to which an event type in the range of 0 through 499 is assigned.
  - *Split*<*number*> (for example, *Split 89*), where <*number*> is the split associated number associated with the vector event. This format is used for events to which an event type in the range of 500 through 999 is assigned.
- `Event Data 2` is an 8-character field that contains additional data encoded as a hex number (for example, *4C*). This number serves as a call identifier. If two or more events with an identical identifier occur at about the same time, it can be concluded that the events were caused by the same call.
- `First Occur` is an 11-character field that contains the date and time when the vector event first occurred (for example, *09/28/13:43*).
- `Last Occur` is an 11-character field that contains the date and time when the vector event last occurred (for example, *09/28/13:48*).
- `Evnt Cnt` (Event Count) contains a number ranging from *1* to *255* that indicates the total number of vector events of this type that have occurred.

Summary of Vector Events

This section contains Table 20 that does the following:

- Lists the number of each vector event supported by DEFINITY BCS and GuestWorks
- Provides a description and an explanation (and sometimes possible causes and solutions) for each event type.

Table 20. Summary of Vector Events

Event Type	Event Description	Event Explanation
1	Call dropped; call not queued at *stop* step.	Vector processing ended without the call being queued to a split and, as a result, the call cannot be answered. This implies that some default condition was not programmed or that the vector was designed to not always answer the call. Also, call was subsequently dropped.
2	Vector with no steps.	The call encountered a vector with no steps administered.
3	1000 step executed.	This can occur due to the following: ■ Incorrect vector programming (for example, including a series of *goto* steps that point to one another). ■ Excessive repetition of a programmed loop during a single call (for example, recurring announcement-wait loop).
4	Administration change.	The administration of this step occurred while the step was being executed. The call flow for this call is unpredictable. Vectors should not be changed while calls are active.
5	Call dropped by vector disconnect timer.	The call was still in vector processing when the vector disconnect timer expired. The call dropped.
10	Retrying announcement.	During an *announcement* step, a *collect digits* step that contains an announcement, or a *disconnect* step, the announcement was not available, and the announcement queue (if specified) was full. The step is retried at regular intervals.

Continued on next page

Table 20. Summary of Vector Events — *Continued*

Event Type	Event Description	Event Explanation
11	No announcement available.	During an *announcement* step, a *collect digits* step that contains an announcement, or a *disconnect* step, the announcement was not available for one of the following reasons: ■ Announcement was not recorded ■ Analog announcement was busied out ■ Integrated announcement board was not installed ■ Integrated announcement ports were busied out ■ Integrated announcement was being recorded or restored
20	Call cannot be queued.	A **queue-to split**, **messaging split**, or **check split** command failed to queue the call. **NOTE:** Event types *520, 521, 522* and *541* may be observed for the same call at the same time.
21	Queued to three splits.	The call attempted to queue to four splits. Multiple split queuing allows the call to queue to a maximum of three splits simultaneously. If the call queued to one or more splits, and if it should now be dequeued from those splits and then queued elsewhere, one solution is to route the call to a station (which may be administered without hardware). Once this happens, the call is forwarded to the VDN that controls the next stage of the call.
30	No tone detector available.	A **collect digits** command failed for the following reasons: ■ Tone detector port was not available ■ All queue slots were occupied
31	Dial-ahead discarded.	Previously entered dial-ahead digits have been discarded via access of a *converse-on*, *route-to number*, or *messaging split* step.

Continued on next page

Table 20. Summary of Vector Events — *Continued*

Event Type	Event Description	Event Explanation
32	Prompting buffer overflow.	The prompting digit buffer already contained the maximum of 24 digits when additional dial-ahead digits were entered by the caller. These additional digits are not stored.
40	*Messaging* step failed.	A *messaging* step failed because the Messaging Adjunct was not available. **NOTE:** Event types *540* and *541* may be observed for the same call at the same time.
50	*Route-to* step failed.	A *route-to* step failed to reach the intended destination. **NOTE:** Event types *51* and *52* may provide more specific information regarding the reason for the failure. See Appendix F, ‘‘Operation Details for the Route-to Command’’.
51	No digits to route-to.	The *route-to digits* step was unable to route the call because the previous *collect digits* step failed to collect any digits. This could result from an error in vector programming (for example, a *route-to digits* step appears without a preceding *collect digits* step). More often, however, this results because the caller was unable to enter the required digits (that is, the caller was using a rotary telephone), or because the caller was not provided with enough information to do so (as can be the case for auto-attendant applications).
52	No available trunks.	A **route-to** command was unable to reach the specified off-switch destination due to a lack of available trunks.
53	Route-to step failed.	The step was unable to seize a trunk because of a hardware problem or glare.
55	Double coverage attempt.	Coverage option on *route-to* step was ignored because double coverage is not allowed. This may happen when the call has covered to a VDN.

Continued on next page

Table 20. Summary of Vector Events — *Continued*

Event Type	Event Description	Event Explanation
70	Busy step for CO trunk.	A CO trunk call reached a *busy* step in a vector without having previously received answer supervision. As a result, the caller continues to hear ringback rather than the busy tone.
80	Time not set.	A *goto* step with a *time-of-day* conditional was processed, but the switch time was not set.
81	No digits collected.	No digits were collected and a comparison was requested against a digit string or “in-table.” The comparison test was considered false and the next step in the vector was executed.
90	Wait step music failed.	A *wait-time* step with music was accessed, but the music was not connected. Music may not be administered correctly.
91	Wait step ringback failed.	A *wait-time* step with ringback was accessed, but the ringback was not connected.
100	Redirect unanswered call.	The call was sent to an agent via a vector, but, due to the Redirection on No Answer feature, the call was redirected from the ringing agent.
101	Redirect of call failed.	The call was sent to an agent via a vector, but, due to the Redirection on No Answer feature, the call was redirected from the ringing agent. The call could not be redirected.
111	Converse no qpos digits.	On a *converse-on* step with passing type *qpos*, information was not available to populate the field.
112	Converse no prompt digits.	On a *converse-on* step with passing type *digits*, information was not available to populate the field.
113	Converse drop during data.	On a *converse-on* step, the converse agent hung up while data was being passed. This may indicate a port failure.

Continued on next page

Table 20. Summary of Vector Events — *Continued*

Event Type	Event Description	Event Explanation
116	Converse transfer denied.	A transfer of a call that was active at a *converse-on* step was attempted. The transfer either failed or was denied, and vector processing continued at the next vector step.
117	Agent drops converse.	While active on a *converse-on* step, an agent became available in a split associated with a *queue-to split* or *check split* step. The call was delivered to the nonconverse agent, and the converse agent was dropped.
125	Data return no digits.	On a *converse-on* step, the converse agent activated data return but did not return any digits.
126	Data return timeout.	On a *converse-on* step, the converse agent activated data return but timed out while waiting to return digits. Vector processing continued at the next vector step.
140	Coverage conference denied.	Coverage to a VDN in a coverage path was denied because more than one party was active on the call.
222	System clock change.	The system clock was changed; therefore, any calculations involving time (that is, ASA and EWT) will be inaccurate.
520	Split queue is full.	A **queue-to split**, **check split**, or **messaging split** command was executed, but the call did not queue to the split because the queue (if administered) was full. To prevent this condition, use a *goto step...if calls queued in split...>...* before each *queue-to split* or *check split* step so that an alternative treatment may be provided for these cases.
521	Not vector-controlled.	The split accessed by a **queue-to split** or **check split** command is not vector-controlled. As a result, the step is skipped.

Continued on next page

Table 20. Summary of Vector Events — *Continued*

Event Type	Event Description	Event Explanation
522	AAS split cannot queue.	A **queue-to split**, **check split**, or **messaging split** command was executed on an auto-available split (AAS), but the call did not queue to the split because all the agents were logged out by Redirection on No Answer.
540	AUDIX link down.	AUDIX could not be accessed via a **messaging split** command, because the AUDIX link was down. As a result, the step is skipped.
541	Not a messaging split.	The split administered for the **messaging split** command is not a messaging split (that is, it does not have a messaging type administered). As a result, the step is skipped.
542	Can’t connect idle agent.	The call at the head of the queue cannot be connected to an idle agent.

Interactions Between Call Vectoring and BCMS

E

Call Vectoring interacts with the Basic Call Management System (BCMS) that helps to monitor and report on the activity within Call Vectoring.

BCMS collects and processes ACD information to generate various reports. This appendix is intended to illustrate how this system interprets and reports on activity within Call Vectoring. Special emphasis is placed on interpreting and reporting on this activity as it occurs within splits during a series of Call Vectoring events.

NOTE:
The manual pages in Appendix A provide a summary of the BCMS interactions with each Call Vectoring command (where applicable).

BCMS Tracking in a Call Vectoring Environment

Tracking is the identifying of various call flows and other actions relevant to call handling. For our purposes, there are three classes of call flows: split flows, VDN flows, and vector flows. Also, we are most concerned with tracking in the Call Vectoring environment. The specific types of call flows and actions in this environment that are tracked by BCMS include the following:

- Inflows (flow ins)
- Outflows (flow outs).

The split supervisor can use VDN and vector flows to evaluate how effective vector programming is at the site in question. The supervisor can use split flows to determine the manner in which the splits at the site are handling incoming telephone calls.

Defining and Interpreting Call Flows

The manner in which specific call flows are defined and interpreted depends upon the call flow class in question and the version of the switch being used.

The following sections define and interpret specific call flows according to these parameters.

VDN Inflows and Outflows

Table 21 illustrates how BCMS interprets specific VDN flows for the switch.

Table 21. BCMS Standards for Interpreting VDN Flows

Flow Type	Interpretation
VDN flow in	Not tracked.
VDN flow out	Calls that successfully flow out of a VDN to another VDN or to an external location via a **route-to** command.

Split Inflows, Outflows, and Dequeues

Before we detail how BCMS interprets split flows, we should discuss the term primary split, since this concept plays a significant role in tracking. Primary split is defined as the first split in a VDN to which a call actually queues. Therefore, this split is not necessarily the first split referenced in the vector.

Another split becomes the primary split if either of the following events occurs:

- Call cannot queue to the originally-targeted split because the split has no queue slots available.
- Call leaves the VDN (via a **route-to** VDN command, for example) and is queued to another split as a result.

If the call leaves vector processing and does not queue to another split (as a result of a **route-to** extension command, for example), there is no new primary split.

With this discussion in mind, let's take a look at Table 22 to see how BCMS interprets split flows for the switch.

Table 22. BCMS Standards for Interpreting Split Flows

Flow Type	Interpretation
Inflow	Calls that ring at an agent in a split other than the primary.
Outflow	Calls that are dequeued from a primary split via a **route-to** or **messaging split** command, or by ringing at or being answered by an agent in another split to which the call is also queued.

When a call is not answered (due to an outflow, abandon, busy, or disconnect), the call's disposition is tracked for the primary split as long as the call is still queued when the call abandons, outflows, and so on. However, if the call abandons or outflows from ringing, the disposition is recorded for the split for which it was ringing.

If the primary split in a VDN is unmeasured, an outflow, abandon, busy, or disconnect is not tracked for the call. Also, an answer is not tracked if the call is answered by an agent in the primary split.

Examples of Split Flow Tracking

The following sections provide some examples of tracking in BCMS. Each section first presents a scenario of Call Vectoring events. The scenario is then followed by a table in which the tracking for the various splits involved is recorded. Following each "tracking table," an explanation of the tracking procedure is provided.

The scenarios presented include the following:

- Call is answered by a primary split.
- Call is answered by a nonprimary split.
- Call is abandoned from queue.
- Call is answered by a primary split after a route to VDN.
- Call is answered by a nonprimary split after a route to VDN.
- Call is answered after a route to split.

NOTE:
Inflows, outflows, and dequeues are not tracked for splits administered by the **converse-on split** command. However, if a call is answered both by a converse split and (subsequently) by a nonconverse split, an "answer" is tracked for each split. However, a call is really considered "answered" only when it is answered by a nonconverse split. Therefore, traffic measurements for converse splits should be used only to measure converse split traffic and not to calculate the total number of calls.

Call Answered by a Primary Split. The following scenario involves a call answered by the primary split. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Call is answered in split 1.

Call Answered by a Non-Primary Split. The following scenario involves a call answered by a nonprimary split. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Call is answered in split 2.

Comments:

Outflow is tracked in split 1 because the call is answered by an agent in another split to which the call is queued (that is, split 2). Inflow is tracked in split 2 because the call is answered in this split and the split is not the primary split. When the call is removed from split 3, outflow is not tracked in split 3 because this split is not the primary split.

Call Abandoned. The following scenario involves a call abandoned by the caller. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Call is abandoned.

Comments:

Abandon is tracked in split 1 because this split is the primary split. Tracking is not recorded in splits 2 and 3 because no dequeue tracking item is available.

Call Answered by a Primary Split after a Route To VDN. The following scenario involves a call answered by the primary split after a **route-to** VDN command is executed. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Vector executes a *route-to VDN* step.
3. Call is then queued to splits 4, 5 and 6.
4. Call is answered in split 4.

Comments:

Split 1 is the original primary split, because this is the first split to which the call actually queues. However, split 4 becomes the new primary split because:

- Call leaves the original VDN upon execution of the *route-to VDN* step.
- Split 4 is the first split to which the call queues upon execution of this step.
- Outflow is tracked in split 1 because this split is the original primary split.

Call Answered by the Non-Primary Split after a Route To VDN. The following scenario involves a call answered by the nonprimary split after a **route-to** VDN command is executed. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Vector executes a *route-to* VDN step.
3. Call is then queued to splits 4, 5 and 6.
4. Call is answered in split 5.

Comments:

Outflow is tracked in split 1 because this split is the original primary split. Outflow is tracked in split 4 because this split becomes the *new* primary split after the *route-to VDN* step is executed. Finally, inflow is tracked in split 5 because the call is answered in this split, and the split is not the primary split.

Call Answered after a Route To Split. The following scenario involves a call answered after it is routed to a split via a **route-to digits** or **messaging split** command. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Vector executes a *route-to digits* (or *messaging split*) step.
3. Call is queued to split 4 and answered by an agent in split 4.

Comments:

Outflow is tracked in split 1 because this split is the original primary split, and the call is answered in split 4, which becomes the new primary split.

Evaluating Split Performance

By using the information presented to this point, along with the information from various reports (as discussed in the next section), the split supervisor can answer the following:

- How many ACD calls offered to my split were “mine” (that is, were offered to this split as the primary split)?

  NOTE:
  Split “ACD calls” include Direct Agent Calls.

For BCMS, the number of calls offered to “my” split that were “mine” can be determined via examination of the BCMS Split Report. The algorithm is as follows: ACDCALLS + ABNCALLS + OUTFLOWCALLS - INFLOWCALLS (that is, the total number of calls answered plus the total number of calls abandoned from “my” split tagged as a primary split plus the number of calls that outflowed “my” split tagged as a primary split minus the number of calls answered that were not directed to “my” split tagged as a primary split).

Using BCMS Reports to Evaluate Call Vectoring Activity

BCMS has a real-time split report, split historical reports, real-time VDN reports, and VDN historical reports. The following list identifies and describes several BCMS reports that summarize Call Vectoring activity. For more information on these and other related reports, refer to *DEFINITY® ECS Basic Call Management System (BCMS) Operations*.

- **BCMS Split Report** summarizes the call activity for an entire split. The information can be requested either daily or by the administered time period. Among other information, the report provides the total number of flow ins (inflows) and flow outs (outflows), the calls answered and calls abandoned. The report also provides the average speed of answer time for calls handled by the split during the indicated time period.
- **VDN Summary Report** summarizes statistical information for all internally-measured VDNs. The information can be requested by the administered time interval or daily. The “list bcms vdn” report gives multiple time periods or days for a single VDN. The “list bcms summary vdn” report gives a one-line summary per VDN (with data from the specified times or days), but can give the data for numerous VDNs.

  The report also indicates the total number of flow outs, specifically, the number of calls that route to another VDN or to a destination external to the switch. However, calls that encounter a **goto vector** command are not shown as outflows. No further measurements are taken on the calls once the calls have outflowed. If an outflowed call later abandons, this is not indicated in the report.

  Among other information, the VDN report provides a total for offered calls, answered calls, abandoned calls, and also one for calls that were either “forced busy” or “forced disconnect.”
- **VDN Real-Time Report** provides statistical information including the number of calls currently waiting and the oldest call waiting. The VDN real-time report has the same characteristics as other real-time BCMS reports.

E Interactions Between Call Vectoring and BCMS
Using BCMS Reports to Evaluate Call Vectoring Activity *182*

Operation Details for the Route-to Command

F

The **route-to** command can be programmed with or without coverage. Table 23 in this appendix summarizes the operation of the **route-to** command for each of the destination types and conditions associated with the commands.

Table 23. Switch Route-To Command Operation

	INTERACTION	
CONDITION	**cov = n ANY STEP**	**cov = y ANY STEP[1]**
Invalid Destination[2]	Goes to next step, else stop	Goes to next step, else stop
VDN Extension[3] - Vector Assigned - Vector Has No Steps	 Goes to new vector Stop[4]	 Goes to new vector Stop[4]
Station Extension Idle (all appearances idle) - CF-ALL Active or - CF-DA Applies - Coverage - DA Applies - All Applies - SAC Applies - None of the Above Applies	 Forwards if possible, else next step, else stop[4] Rings idle app. Goes to next step, else stop[4] Rings idle appearance Rings idle appearance	 Forwards if possible, else coverage, else busy Coverage on DA Coverage Coverage Call delivered and is allowed to cover

Continued on next page

Table 23. Switch Route-To Command Operation — *Continued*

CONDITION	INTERACTION cov = n ANY STEP	INTERACTION cov = y ANY STEP[1]
Station Extension Active (with idle 2-way app)		
- CF-ALL Active	Forwards if possible, else next step, else stop[4]	Forwards if possible, else coverage, else busy
- Coverage		
- DA Applies	Rings idle app (no DA timing)	Coverage on DA
- Ext Act Applies	Rings idle appearance	Coverage
- All Applies	Goes to next step, else stop[4]	Coverage
- SAC Applies	Rings idle appearance	Coverage
- None of the Above Applies	Rings idle appearance	Call delivered and is allowed to cover
Station Extension Busy (no idle 2-way app)		
- Extension in Hunt Grp (also see ACD Hunt Grp)	Queues if possible, else next step, else stop[4]	Queues if possible,else coverage, else busy
- CF-ALL Active or -CF-DA Applies	Forwards if possible, else next step, else stop[4]	Forwards if possible, else coverage, else busy
- Call Waiting to Analog Sta Would Apply	Goes to next step, else stop[4]	Call waits
- Coverage		
- Ext Act Applies	Goes to next step, else stop[4]	Coverage
- Ext Bsy Applies	Goes to next step, else stop[4]	Coverage
- All Applies	Goes to next step, else stop[4]	Coverage
- SAC Applies	Goes to next step, else stop[4]	Coverage
- None of Above Applies (or hunt, fwd, or cov dest is unavailable)	Goes to next step, else stop[4]	Busy tone given
Extension with Incompatible COR	Goes to next step, else stop.	Goes to next step, else stop.

Continued on next page

Table 23. Switch Route-To Command Operation — *Continued*

CONDITION	INTERACTION cov = n ANY STEP	 cov = y ANY STEP[1]
Terminating Extension Group		
- All Members Idle	Rings idle appearance	Call delivered and is allowed to cover
- A Member Active on TEG	Goes to next step, else stop[4]	Coverage, else busy
- No Idle App on Any Member	Goes to next step, else stop[4]	Coverage, else busy
Hunt Group Extension		
- Idle Agent	Rings idle appearance	Call delivered and is allowed to cover
- No Idle Agent		
- Call cannot queue	Goes to next step, else stop[4]	
- Call can queue	Call is queued	Busy tone given Call is queued
Extension on Another Node (Uniform Dialing Plan - UDP)		
- Trunk Available	Call delivered	Call delivered
- Trunk Not Available	Goes to next step, else stop[4]	Queues if possible, else reorder
Trunk Access Code (TAC) Destination		
- Trk Grp No Dial Access	Goes to next step, else stop[4]	Routes to local and
- Trunk Available	Call delivered	Call delivered
- Trunk Not Available	Goes to next step, else stop[1]	Queues if possible, else reorder
AAR/ARS FAC Dest. (including Subnet Trkng)		
- Trk Grp No Dial Access	Tries next route	Routes to local attendant
- Trunk Available	Call delivered	Call delivered
- Other Routes Avail	Tries next route	Tries next route
- All Routes Busy		
- No Pattern Queuing	Goes to next step, else stop[4]	Reorder tone given
- Queuing Assigned	Goes to next step, else stop[4]	Queues to pattern

Continued on next page

Table 23. Switch Route-To Command Operation — *Continued*

	INTERACTION	
CONDITION	**cov = n ANY STEP**	**cov = y ANY STEP[1]**
Attendant Queue (dial 0)		
- Idle Atnd	Rings idle appearance	Call delivered and is allowed to cover
- No Idle Atnd		
- Not In Night Svc	Call is queued	Call is queued
- In Night Svc		
- Nite Dest. Assigned	Delivered to night svc.	Delivered to night svc.
- Not Assigned	Call is queued	Call is queued
Individual Attendant Access		
- Atnd Idle	Rings idle appearance	Call delivered and is allowed to cover
- Atnd Busy	Queues if possible else Goes to next step, else stop[4]	Queues if possible, else Busy tone given
Inter-Switch Atnd Calling		
- Trk Grp Controlled	Routes to local atnd	Routes to local atnd
- Trk Available	Call delivered	Call delivered
- Trk Not Available	Goes to next step, else stop[4]	Reorder tone given

1. The call is removed from vector processing (that is, the call is taken out of any split qu feedback, such as music or ringback, is removed) for *with coverage y* interactions, destination is not available. The call is treated as though the destination was directly dialed. This includes coverage, forwarding, call cannot be completed treatments (busy, reorder, and intercept) and displays (answering station sees only caller name and number). A call that has reached this vector via coverage to a VDN will not be further forwarded or otherwise redirected.

2. Invalid destinations include the following: empty (for example, zero collected digits) or invalid *route-to* destination number, unassigned extension number, incomplete number of digits for AAR/ARS pattern, non-AAR/ARS feature access code (FAC), maintenance busy station extension, COR of the VDN that prevents access (for example, origination restricted), FRL of a VDN that is lower than required for the AAR/ARS pattern access, no routes assigned to the AAR/ARS pattern, incompatible calling and destination partitions, ACTGA trunk group destination, or an off-net forwarding destination. If a trunk access code (TAC) destination is involved, and if the TAC is for a CO/FX trunk with a *route-to with coverage n* step, the digits entered must match a valid ARS analysis string. If not, the destination is considered invalid. For other trunk types with a *route-to number* or *route-to digits with coverage n* step, the step succeeds when the trunk is seized (that is, vector processing stops). For a *route-to with coverage y* step, the step succeeds if the TAC is assigned.

3. A call that routes to a VDN via the route-to number with coverage = “yes unconditionally” command behaves like a directly-dialed call instead of a VDN call. Therefore, the terminating station’s display shows only the originating station information and does not show the VDN information (for other types of VDN calls, the terminating station would see the VDN name).

4. The interaction “Stop” means the following: vector processing is stopped, the call remains queued to a split, and the caller continues to hear feedback initiated by a previous step. In the case where the route-to command fails and processing stops (due to a busy station or trunk group destination), retry can be implemented in the vector. Retrying is accomplished by including an unconditional *goto* step as the last step to allow for a loop back to the route to command. Use of an intermediate wait-time command step with appropriate feedback and delay interval is strongly recommended to reduce processor occupancy.

Detailed Call Flow and Specifications for Converse—VRI Calls

G

This appendix discusses the detailed call flow for calls involving a *converse-on* vector step and Voice Response Integration (VRI). This call flow is segmented into the following phases:

1. Converse call placement
2. Data passing (optional)
3. VRU data collection (optional)
4. Script execution
5. Data return (optional)
6. Script completion
7. Switch data collection (optional).

NOTE:
If, during any phase of this call flow, a *converse-on* step is executed while the caller is in the split queue and an agent becomes available to service the caller, the VRU port is dropped, vector processing is terminated, and the calling party is immediately connected to the available agent.

Converse Call Placement

The first action taken by the *converse-on* step is to deliver the call to the converse split. Ringback tone is not heard by the caller. Any audible feedback supplied by vector processing remains until the VRU answers the call and all digits (if administered) have been outpulsed to the VRU. Vector processing is suspended. Callers remain in any nonconverse split queues, and they retain their position in queue while the converse session is active.

If a Call Prompting tone detector is allocated to the call, the tone detector is released. Any dial-ahead digits are discarded. However, any digits collected prior to the *converse-on* step are kept.

Calls to busy converse splits are allowed to queue. The priority of the call in queue is administrable within the *converse-on* step. Again, any audible feedback supplied by vector processing continues until the call is answered by the VRU and any data is outpulsed. Calls to busy converse splits have either no queue or a full queue fail. For this scenario, a vector event is logged, and vector processing continues at the next vector step.

Whenever a *converse-on* step places a call to an auto-available split whose agents are all logged out, the call is not queued. Instead, the *converse-on* step fails, a vector event is logged, and vector processing continues at the next vector step.

NOTE:
Usually, this scenario occurs whenever the Voice Response Unit (VRU) goes down, the ports are members of an Auto-Available Split (AAS) and the Redirection on No Answer feature has taken all the ports out of service.

The originator's display is not changed by the terminating or answering of a converse call. Also, whenever a call is delivered to a display station via a *converse-on* step, the station displays the following information: "Originator Name to VDN Name." Conventional Call Vectoring rules for Override are in effect.

Valid destinations for converse calls must be vector-controlled and include the following:

— Hunt groups

— ACD (including Auto-Available) splits

— AUDIX hunt groups

NOTE:
Even though AUDIX hunt groups are valid destinations for converse calls, they do not need to be vector-controlled.

Undefined and nonvector-controlled hunt group, split numbers are rejected at administration time.

Any attempt to remove a hunt group, split administered within a *converse-on* vector step is denied until the vector has been changed. Also, any attempt to make a hunt group or split nonvector-controlled is denied if the hunt group or split is called by a *converse-on* step.

Data Passing

NOTE:
This phase is optional and is in effect only if the application calls for the switch to pass information in-band to the VRU.

The *converse-on* step may outpulse up to two groups of digits to the VRU. The digits can serve two major purposes, as follows:

- Notify the VRU of the application to be executed
- Share call-related data, such as caller digits collected by the switch.

In many applications both application selection and data sharing are required.

Since in many cases the digit strings are of variable length, the switch always appends a pound sign (#) to the end of each digit string. *Prompt and collect* steps in the VRU script must therefore always be administered to expect the pound sign (#) as the end-of-string symbol and to include the pound sign in the digit count.

Sending the pound sign (#) prevents excessive delays and other problems caused by digit timeouts.

The complete outpulse sequence is summarized as follows:

1. VRU answers the call.
2. Delay for the time administered in the "`Converse first data delay`" field in the System Parameters-Features form occurs.
3. <data_1> is outpulsed.
4. "#" is outpulsed.
5. Delay for the time administered in the "`Converse second data delay`" field in the System Parameters-Features form occurs.
6. <data_2> is outpulsed.
7. "#" is outpulsed.

NOTE:
The length of DTMF tones (digits) and the interdigit pause between tones is administrable on the Feature-Related System Parameters form. The optimum timers for tone and pause depend on the system being used. See the *DEFINITY® ECS Administrator's Guide* for details.

Any audible feedback supplied by the switch is disconnected only after the outpulse sequence is completed. Also, any touch-tone dialing by the calling party during the data passing phase does not result in data corruption.

The following values may be administered for <data_1> and <data_2> within the **converse-on** command:

- **Administered digit string**: This string can contain up to six characters consisting of one or more digits (0 through 9) or asterisks (∗). The pound sign (#) may not be included in a digit string because it is reserved as the end-of-string character. However, a single "#" may be administered.
- **ani**: Not supported in DEFINITY BCS and GuestWorks.
- **vdn**: This data type causes the VDN extension to be outpulsed. In cases where multiple VDNs are accessed, normal VDN override rules determine which VDN extension is outpulsed.
- **digits**: This data type can be used only if Call Prompting is optioned, and it causes the most recent set of digits collected in vector processing to be outpulsed. If no digits are available, the end-of-string pound sign (#) is the only character outpulsed.
- **qpos**: This data type causes the value of the queue position of a call in a nonconverse split to be outpulsed. This value is a variable length data item from which between one and three digits can be outpulsed. Valid ranges for the value are 1 through 200 in G3si, and 1 through 999 in G3r. If the call is not queued, the end-of-string pound sign (#) is the only character outpulsed.

  NOTE:
  The use of this keyword is not recommended with multiple split queuing because any queue position value sent may not be meaningful. However, if the call is queued to multiple nonconverse splits, the value of the caller's queue position in the first nonconverse split is sent.

  This data may be used by the voice information system to inform callers of their position in queue or to decide whether to execute a long or short version of a voice response script.
- **wait**: This data type sends the expected wait time for a call in vector processing that is queued to at least one split. It is a value from 0 to 9999 seconds (variable length, that is, not padded with zeros) always followed by a # digit. If the call is not queued, or is queued only to splits with no working agents, only the # is outpulsed.
- **"#"**: This is the only character outpulsed. Outpulsing this character causes the corresponding prompt and collect command in the voice response script to be skipped.
- **"none"**: This data type causes no characters to be outpulsed. Also, no end-of-string pound character (#) is outpulsed, and no time delays are invoked.

The switch always outpulses a pound character (#) at the end of each digit string. Where "#" is administered, or where the "digits" keyword is administered and the last digit collected from the caller is "#," only one "#" is outpulsed. No "#" is outpulsed when the keyword "none" is administered.

If <data_1> is administered as "none," <data_2> must also be "none."

Any data to be passed to the VRU from the switch is outpulsed in-band. Two time delays on the System Parameter-Features form ("Converse first data delay" and "Converse second data delay") are administrable by customers. These delays may range from 0 through 9 seconds, with a default of zero seconds for the converse first data delay and a default of 2 seconds for the converse second data delay. The delays may be needed to give the VRU time to invoke an application and allocate a touch-tone receiver to receive the passed digits.

If <data_1> is not "none," the converse first data delay timer starts when the call is answered by the VRU. Once the timer expires, the data_1 digits are outpulsed in-band to the VRU, followed by the end-of-string pound sign (#).

If <data_2> is not "none," the converse second data delay timer starts when the end-of-string pound sign (#) from the first digit string is outpulsed. Once the timer expires, the data_2 digits are outpulsed in-band to the VRU, followed by the end-of-string pound sign (#).

No time delays are invoked when the keyword "none" is administered.

NOTE:
The outpulsing of digits is not heard by the caller.

If the VRU hangs up during the data passing phase, the switch will log a vector event, reactivate vector processing at the next vector step, and ensure that the VRU port is accessible for future calls.

Once all digits have been passed to the VRU, any audible feedback is disconnected.

NOTE:
At this point, control has effectively been passed to the VRU.

To ensure the robust operation of the VRU data passing operation, be sure to implement the following recommendations:

- Include the **prompt and collect** command in the VRU script for each data field passed in the *converse-on* step.
- Administer each ***prompt and collect*** command to recognize the "#" character as the end-of-string character.
- Ensure that the number of digits expected is one greater than the number of digits passed to allow for the "#" character, which terminates every converse data field.

  Also, ensure that an announcement is not played in these *prompt and collect* steps.
- Ensure that the first digit timeout in the *prompt and collect* steps is 5 seconds greater than the corresponding converse data delay. (For example, if the *converse-on* step passes two data fields, and if the converse first data delay is 0 secs and the converse second data delay is 4 secs, the first digit timeouts for the two **prompt and collect** commands should be at least 5 and 9 seconds, respectively.)
- Ensure that he interdigit timeout in the *prompt and collect* steps is at least 5 seconds.
- Administer the converse first data delay to give a VRU under a heavy load sufficient time to allocate a DTMF touch-tone receiver after answering the call.
- Administer the converse second data delay to give a VRU under a heavy load sufficient time to complete any tasks between the first and second **prompt and collect** command. (For example, the VRU can invoke a new application if the first data field passed is used to identify the application script to be executed.)
- In general, for *converse-on* steps pass data to the VRU, ensure that the VRU script does not execute any commands between the time when the call is answered and the time when the first **prompt** *and* **collect** command is executed.

VRU Data Collection

When digits are passed from the switch to the VRU, the first VRU script commands executed are **answer phone** and **prompt and collect**. No announcement is programmed for the *prompt and collect* command, and the pound sign (#) is programmed as the end-of-string sign. If two sets of digits (that is, <data_1> and <data_2>) are passed by the switch, there will be two **prompt and collect** commands on the VRU to receive them.

If the first digit string (<data_1>) passed to the VRU is for application selection, the VRU system invokes the appropriate script. If a second digit string (<data_2>) is also used to pass an argument to this selected application, the first command in the *exec'*ed script is a **prompt and collect** command with no announcement prompt programmed and with the pound sign (#) programmed as the end-of-string character.

The "Converse second data delay" is used to give the VRU time to invoke the selected application before the <data_2> digit string is outpulsed.

The application developer should ensure that the administered *converse first data delay* and *converse second data delay* timers allow sufficient time for the VRU to successfully collect all outpulsed digits, even during periods of heavy call volume. Loss of digits from <data_2> is an indication that the converse second data delay timer needs to be increased.

Script Execution

During script execution, digits input by the calling party in response to **prompt and collect** commands are collected by the VRU but are not collected by the switch as dial-ahead digits. Also, audible feedback is determined by the VRU.

If an agent from a nonconverse split becomes available to service the call while the VRU script is being executed, the VRU port is dropped from the call, and the caller is immediately connected to the agent. Any digits collected prior to executing the *converse-on* step are still available and may be displayed using the CALLR-INFO button.

The entire call is dropped if the caller abandons during the execution of a *converse-on* step.

Data Return

This phase is optional and is in effect only if the application calls for the VRU to return information to the switch before returning control to vector processing.

Digits returned by the VRU are treated as dial-ahead digits. The rules for collecting and processing VRU-returned digits are identical to those for collecting and processing Call Prompting digits (see Chapter 4, ‘‘Call Prompting’’).

VRU data return is done in a manner similar to an analog transfer. Specifically, the VRU does an analog switchhook flash, outpulses DTMF digits, and then hangs up. If converse data is returned, the DTMF digits comprise two parts. The first sequence of digits is the converse data return feature access code administered on the Feature-Access-Codes form. The second sequence of digits is the sequence to be passed by the VRU. These digits are collected later during vector processing.

To ensure the robust operation of the VRU data return operation, be sure to follow these recommendations:

- Set the analog flash timing to 600 msecs.
- Ensure DTMF tones last at least 70 msec and interdigit pauses last at least 50 msec. This results in an outpulsing rate up to 8.33 digits per second.
- Hang up line to switch after outpulsing digits. Assume that the switch will wait between 1.2 and 1.5 secs to determine that the hang-up is a disconnect.
- After the flash, ensure that the VRU performs dialtone detection (stutter dialtone) for a sufficient period of time to ensure accurate detection (typically 0.6 to 1.0 secs) before outpulsing the converse data return feature access code.
- If dial tone is not received before the timeout, ensure that the VRU does two more retries of the analog flash. Also, if no dialtone is detected after two retries, ensure that the VRU logs an error.
- Whenever dial tone is detected, ensure the digits of the converse data return feature access code are outpulsed.
- After the converse data return feature access code is outpulsed, the returned digits can be outpulsed without waiting for the second dial tone.
- After the VRU digits are outpulsed, the line to the switch is dropped.

Assuming an outpulse rate of 8 digits per sec (0.125 secs per digit), a 3-digit feature access code and stutter dial tone detection time of 0.6 secs, the maximum of 24 digits passed to the switch should take about 6 secs (1.2 secs disconnect plus 8 secs plus 0.125 secs per digit).

The TN744 or TN2182 Call Classifiers required by the Call Prompting feature are not required for returning digits in-band from the VRU to the switch. Instead, general purpose tone detectors can be used. As long as dial-ahead digits are available, any *collect digits* steps following a *converse-on* step do not require a TN744 or TN2182 to be allocated to the call.

If general purpose tone detectors are not immediately available, and if the call queues for a tone detector, dial tone is not provided. For this scenario, the VRU does not outpulse any digits until a tone detector is available and dial tone is provided.

If there are no general purpose tone detectors available on the switch, and if there is no space in the tone detector queue, the operation fails. Usually, the VRU logs an error and then quits, and vector processing continues at the next vector step. Existing system measurements reports indicate when the system is configured with an insufficient number of tone detectors.

The “Converse Data Return Code” can be followed by a maximum of 24 digits. The VRU touch-tones the code and the digits in-band. However, the code and the digits are not heard by the caller. The digits are stored in the switch as Call Prompting dial-ahead digits. If “x” digits are collected by vector processing before the *converse-on* step is executed, the maximum number of digits that can be returned is reduced to “24-x.” Any additional digits returned by the VRU are discarded. The data return is completed once the VRU hangs up.

The digit string returned by the VRU can consist of the digits (0 through 9) and pound signs (#). The pound sign (#) is interpreted by the *collect digits* step as an end-of-string character. If the digit string being returned is of variable length, the VRU can terminate the string with a pound sign (#) to avoid the 10-second timeout delay that occurs when the digits are collected. If the digit string being returned is “multi-part” (that is, to be collected by multiple *collect digits* steps), and if some of the parts are of variable length, the pound sign (#) can be used to terminate each of the variable length parts.

NOTE:
An asterisk (∗) may be included as part of the converse data return code. However, since the asterisk is interpreted as a “delete” character by the switch, it makes little sense to use it as a returned digit. If it is used as such, all characters returned prior to the asterisk are discarded.

During the data return phase, the caller is temporarily put on hold. Music-on-hold, if administered, is suppressed. Since the caller hears silence during this phase, feedback should be provided to the caller as soon as possible after the *converse-on* step is executed.

Any touch-tone digits dialed by the calling party during the data return phase are discarded. These digits do not cause data corruption, and they are not collected as dial-ahead digits by the switch.

If an interdigit timeout occurs during the data return phase, the switch logs a vector event, keeps the digits already returned, drops the VRU, and reactivates vector processing at the next vector step.

If the timeout occurs before the converse data return code is returned, the operation is the same except that no discarded digits will be available.

Script Completion

The VRU script returns control to vector processing on the switch by simply hanging up the line. In cases where no data is returned to the switch, this is done usually by executing the **quit** command. In cases where data is returned, this occurs whenever the VRU hangs upon completion of the VRU data return operation.

The last set of digits collected before the *converse-on split* step is executed is still available and may be displayed by an answering agent on the nonconverse split by using the CALLR-INFO button.

A VRU script can be programmed to continue running after hanging up the voice line. This after-call work is usually very short, and it may involve either a final message to a host or a final update to a local database. For this scenario, the VRU port (channel) is still associated with the running script even though there is no longer a voice connection.

From the switch's point of view, the agent (port) is available for the next call. If a call is delivered to this port, the VRU does not answer the call until the previous script has completed. As long as the VRU script's after-call work is short in duration, this poses no significant problem for the VRI feature. However, high volume VRI applications with lengthy after call work periods should be avoided, especially if such periods are so lengthy they approach the administered timeout period on the switch for the Redirection on No Answer feature. In such a case, the Redirection on No Answer feature might think that the VRU ports are faulty and therefore start to take these ports out of service.

Switch Data Collection

NOTE:
This phase is in effect only if the VRU returns information to the switch.

Once the VRU script has completed and vector processing is reactivated, the returned digits are collected and processed by vector commands in the usual manner. Since the digits must be collected by a **collect digits** command, data may be returned and processed only if the Call Prompting option is enabled.

The data returned can consist of multiple parts. For example, the VRU could return a stream of seven digits in which a single digit success/fail code is followed by a six-digit account code. For this scenario, the *converse-on* step would be followed by a sequence of vector steps including two *collect digits* steps. The first *collect digits* step would collect one digit and then check the result code; the second *collect digits* step would collect the six-digit account code.

Any touch-tone digits dialed by the calling party during the data collection phase are discarded, do not cause data corruption, and are not collected as dial-ahead digits by the switch.

If VRU data is returned, the calling party is able to touch-tone a response to a switch prompt only after the data collection phase is completed and another *collect digits* step is executed. This is true because each executed *collect digits* step does not allocate a tone detector when dial-ahead digits are present. Since VRU-returned digits are treated as dial-ahead digits, a tone detector is attached to the call only after all returned digits are collected and another *collect digits* step is encountered. Only at this point can the caller hear an announcement for the **collect digits** command and successfully enter digits.

Setting Up Agents in an ACD Hunt Group

H

Managers need some key indicators to measure ACD performance at their site. Usually, in setting up a group of agents, several factors involving call management are considered. The following list identifies and defines the most common of these factors, and it provides a typical question that might be asked. In addition, an insurance company example will be used to discuss the different options in this appendix.

- **Volume**

  Number of calls going in or out of the ACD. (How many calls did split 1 answer?)

- **Productivity**

  Call volume per unit of time. (How many calls did split 1 answer between 8 a.m. and 9 a.m.?)

- **Utilization**

  Overall use of the call group. (What was my agent occupancy?)

- **Accessibility**:

  Availability of lines and agents when customers call the ACD. (Were lines busy when customers called or did the customer have to wait too long?)

- **Quality of Service**:

  Accuracy of information, a pleasant manner, responsiveness to caller concerns, successful completion of business, and efficient time utilization. (Was the caller given good service?)

Call Vectoring Setup

To set up a group of agents that has Call Vectoring, do the following:

1. Determine your group's objectives. Think about how you want your group to handle calls and also about what you want your group to achieve. See Worksheet #1.

   A company's basic goals are to increase profits and market share and to decrease costs. It is best to have more than one objective. (Some customers set and then live by only one objective.) Objectives must then be created to meet the goals. These objectives must be communicated to the Split Supervisor or to the Administrator managing the group.

   The following list provides an example set of objectives:

   - Establish the following measured entities:
     - Average Speed of Answer = 15 seconds
     - Abandon Rate <= 3%
     - Average Talk Time = 2 1/2 minutes
     - ACD calls per agent = 80 to 90 per day
     - Number of calls in queue = 6
     - Percentage of calls answered within the service level = 95%
     - Agent occupancy > 90%
     - Percentage of trunks busy < 3%
   - Generate revenue.
   - Train agents to back up each other.
   - Adequately train agents to provide service that meets customer expectations.

2. Review your existing operation and determine your customer needs (see Worksheet #2: Current Split Operation and Table 24).

3. On the switch, assign a unique Hunt Group number and Call Distribution method to each caller need. This number will be your split number (see Worksheet #3: Customer Needs and Table 24).

4. Assign Dialed Number Identification Service (DNIS) (that is, the number dialed) as a VDN (see Table 24).

   As an option, you can assign one VDN for a main number and use Call Prompting to route the call to the proper split.

Table 24 illustrates the guidelines given up to this point.

Table 24. Customer Needs Guidelines

Customer Needs	Split Number (Hunt Group)	Call Distribution[1]	VDN
New policy	1	UCD	555-6543
Questions about policy, Rate Quotes, Billing	2	UCD	555-6432
Spanish speaking for policy, service, and claims	3	DDC	555-6321
Claims	4	UCD	555-6210

1. Options include Direct Department Calling (DDC) and Uniform Call Distribution (UCD).

Notice that this group has only one split for all Spanish calls. However, resources permitting, you could create a New Policy split, a Service split, and a Claims split, each containing agents who speak Spanish. As an alternative, you could use one main VDN to point to a Call Prompting vector designed to route the calls to the splits.

5. On the switch, assign extensions to the agents' physical terminal locations (see Table 25).
6. On the switch, assign agent extensions to splits (see Table 25).

   More than four splits can be assigned to an agent; however, the agent can log into a maximum of four splits.

Table 25 illustrates the assignment of agent extensions to splits:

Table 25. Agent Extension/Split Assignments

Split (Hunt Group)	Agent Extensions
1 - Sales	1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239
2 - Service	1231, 1232, 1234, 1238, 1239, 1240
3 - Spanish	1238, 1240, 1245
4 - Claims	1238, 1239, 1240, 1241, 1242

7. On the switch, assign a vector to each VDN (see Table 26).

   A VDN can point to only one vector. However, a vector than one VDN pointing to it.

   Table 26 illustrates VDN/vector assignments.

Table 26. VDN/Vector Assignments

VDN	Vector
6543	1 (Sales)
6432	2 (Service)
6321	3 (Spanish)
6210	4 (Claims)

8. On the switch, write your vectors. See Worksheet #4: Vector Design.

   Your vectors should match your objectives. To meet these objectives, you must make a number of relevant decisions (for example, you may decide how soon you want to enlarge an agent pool or what kind of treatment the caller should receive). If your VDN and vector reports do not satisfy your objectives, you must consider your alternatives (for example, you may deem it necessary to train agents or to increase the amount of time elapsed from when a call queues to one split and then to another split).

The following lists indicate the actions produced by two different vectors:

Actions Produced by Vector #1:

a. Tell the caller to select one of the following prompts:
   - 1 = Sales
   - 2 = Service
   - 3 = Spanish
   - 4 = Claims
   - Nothing or 0 = Service

b. Queue the call.

c. Provide an announcement to the caller.

Actions Produced by Vector #2:

a. Queue the call to the correct service at a medium priority.

b. If no agents are available, provide a message and then play music.

c. If the call is not answered within 10 seconds, provide a second message and then play music.

d. If the call is not answered within 7 more seconds, queue the call to the Service split.

e. If the call is not answered within 7 more seconds, queue the call to the Spanish split at a high priority.

NOTE:
A **check split** command queues the call to up to three splits if the conditions are met. If the conditions are not met, the **check split** command may not get read again (if the vector step in which it appears is not executed again).

9. Once your system is up and operational, you will need to monitor it to ensure that you are meeting your objectives. BCMS can be used to monitor some of your objectives. Some objectives will need to be monitored and have adjustments made in real time. For example, if the number of calls waiting, average speed of answer, or percent answered within a service level is not meeting your objectives, you might want to immediately move some agents or direct calls to another vector. Other items such as agent occupancy and percent all trunks busy may only need to be monitored daily to look for trends.

What Are My Call Group Objectives?

Figure 2. Worksheet #1: Call Group Objectives

Split _____

Primary Backup _____	Secondary Backup _____	Tertiary Backup _____

List your customer/caller needs and your agent knowledge levels for this split.

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____

Split _____

Primary Backup _____	Secondary Backup _____	Tertiary Backup _____

List your customer/caller needs and your agent knowledge levels for this split.

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____

Figure 3. Worksheet #2: Current Split Operation

Customer/Call Group Needs	Split Number (Hunt Group)	Call Distribution	VDN

Figure 4. Worksheet #3: Customer Needs

Vector # _____ Name _____ Description _____

Assigned VDNs _____

Assigned Trunk Groups _____

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____
13. _____
14. _____
15. _____
16. _____
17. _____
18. _____
19. _____
20. _____
21. _____
22. _____
23. _____
24. _____
25. _____
26. _____
27. _____
28. _____
29. _____
30. _____
31. _____
32. _____

Figure 5. Worksheet #4: Vector Design

Improving Performance

This appendix provides recommendations on how to write vectors that promote favorable performance practices. Two basic principles to follow are as follows:

1. Minimize the amount of call processing.
   - — Minimize the number of vector steps to process a call.
   - — Use the lower cost steps when possible (refer to Table 29 and Table 30).
2. Avoid vector steps which have a substantial probability of failure.
   - — Calls made outside of business hours
   - — Queues to groups with less than desirable resources or characteristics.

The most wasteful use of processing resources is frequently caused by inefficient looping. For example, performance could be compromised when a vector loops through steps too often. This is especially true with long queue times.

Some examples with looping are discussed and recommendations are given on how to maximize performance. They are as follows:

- Audible Feedback
- Check.

Examples other than looping are also discussed, such as after business hours.

All looping examples in this appendix use only loops within a single vector. It is important to also be aware of looping to other vectors through the use of vector chaining. The same principles can be extrapolated from the looping examples. Creating a flow diagram is often helpful for identifying looping errors.

In addition to the example vectors, tables rating the relative performance costs of specific vector commands are also included.

NOTE:
Remember to test vectors for performance in addition to call flow.

Looping Examples

Audible Feedback

Recommendation: Evaluate the length of the wait period between repetitions of an announcement and increase the length, if possible. For optimum performance, add a second announcement after the initial announcement and repeat the second announcement less often.

The example in Screen 45 repeats the “All representative are busy. Please hold.” announcement every 10 seconds as long as the call is in queue.

```
1. queue-to split 1
2. announcement 2770     (“All representatives are busy. Please hold.”)
3. wait-time 10 secs hearing music
4. goto step 2 if unconditionally
5. stop
```

Screen 45. Example Vector

The example in Screen 46 repeats the announcement only every 60 seconds, thus improving performance.

```
1. queue-to split 1
2. announcement 2770     (“All representatives are busy. Please hold.”)
3. wait-time 60 secs hearing music
4. goto step 2 if unconditionally
5. stop
```

Screen 46. Example Vector with Improved Performance

The example in Screen 47 adds a second announcement, “All representatives are still busy. Please hold.” in addition to the initial announcement and repeats the second announcement less often (every 120 seconds), thus improving performance again.

```
1. queue-to split 1
2. announcement 2770     (“All representatives are busy. Please hold.”)
3. wait-time 120 secs hearing music
4. announcement 2771     (“All representatives are still busy. Please
continue to hold.”)
5. goto step 3 if unconditionally
6. stop
```

Screen 47. Another Example Vector with Improved Performance

Table 27 compares the relative processing cost of the three examples by looking at the approximate number of vector steps executed while processing the call. Assumption is that the first announcement is 3 seconds long and the second announcement is 4 seconds long.

Table 27. Approximate Number of Vector Steps Executed for the Audible Feedback Examples

	Example in Screen 45	Example in Screen 46	Example in Screen 47
when an agent is available in split 1	1	1	1
queueing time of 5 minutes	70	15	9

When a call is queued for 5 minutes, the number of vector steps drops dramatically when the amount of time between announcements is increased (Screen 46), and drops even more when a second announcement is added, and the amount of time between announcements is increased again (Screen 47). When an agent in split 1 is immediately available to answer the call, there is no difference in the number of vector steps for the three examples.

Check

Recommendation: When using check commands to queue a call to backup splits, ensure that an adequate amount of time has elapsed before checking the backup splits again.

The example in Screen 48 checks backup splits continuously as long as the call is in queue.

```
1. queue-to split 1 pri h
2. announcement 3000
3. wait-time 10 secs hearing music
4. check split 21 pri m if available-agents > 0
5. check split 22 pri m if available-agents > 0
6. check split 23 pri m if available-agents > 0
7. check split 24 pri m if available-agents > 0
8. check split 25 pri m if available-agents > 0
9. goto step 4 if unconditionally
```

Screen 48. Example Vector

The example in Screen 49 adds a delay of 10 seconds to ensure that some time has elapsed before checking the backup splits again.

```
1. queue-to split 1 pri h
2. announcement 3000
3. wait-time 30 secs hearing music
4. check split 21 pri m if available-agents > 0
5. check split 22 pri m if available-agents > 0
6. check split 23 pri m if available-agents > 0
7. check split 24 pri m if available-agents > 0
8. check split 25 pri m if available-agents > 0
9. wait-time 10 secs hearing music
10. goto step 4 if unconditionally
```

Screen 49. Example Vector with Improved Performance

Since the agent availability status may not be likely to change every 10 seconds, it may make sense to increase the wait time to 30 seconds, as shown in the example in Screen 50.

```
1. queue-to split 1 pri h
2. announcement 3000
3. wait-time 30 secs hearing music
4. check split 21 pri m if available-agents > 0
5. check split 22 pri m if available-agents > 0
6. check split 23 pri m if available-agents > 0
7. check split 24 pri m if available-agents > 0
8. check split 25 pri m if available-agents > 0
9. wait-time 30 secs hearing music
10. goto step 4 if unconditionally
```

Screen 50. Another Example Vector with Improved Performance

Table 28 compares the relative processing cost of the three examples by looking at the approximate number of vector steps executed while processing the call. Assumption is that the announcement is 5 seconds long.

Table 28. Approximate Number of Vector Steps Executed for Check Examples

	Example in Screen 48	Example in Screen 49	Example in Screen 50
when an agent is available in split 1	1	1	1
queueing time of 5 minutes	up to 1000	190	65

When a call is queued for 5 minutes, the number of vector steps drops dramatically when a delay is added before checking the backup splits again (Screen 49), and drops even more when the length of the delay is increased again (Screen 50). When an agent in split 1 is immediately available to answer the call, there is no difference in the number of vector steps for the three examples.

Other Examples

After Business Hours

Recommendation: Test to see if the destination resources are available (such as during business hours) before queuing.

The example in Screen 51 queues calls to a hunt group regardless of the time of the call. When the call is made after business hours, the announcement is repeated until the caller hangs up.

```
1. queue-to split 1
2. announcement 5000 (“All agents are busy. Please hold.”)
3. wait-time 120 secs hearing music
4. announcement 5001 (“All agents are still busy. Please continue to
hold.”)
5. goto step 3 if unconditionally
```

Screen 51. Example Vector

The example in Screen 52 tests for business hours before queuing the call. If the call is made after business hours, an announcement informs the caller of the business hours and the call is terminated.

```
1. goto step 7 if time-of-day is all 17:00 to all 8:00
2. queue-to split 1
3. announcement 5000 (“All agents are busy. Please hold.”)
4. wait-time 120 secs hearing music
5. announcement 5001 (“All agents are still busy. Please
continue to hold.”)
6. goto step 4 if unconditionally
7. disconnect after announcement 5001 (“Business hours are 8:00 AM to
5:00 PM, Please call back then.”)
```

Screen 52. Example Vector with Improved Performance

In the first example, unnecessary processing occurs when a call is queued after business hours and the call is terminated only when the caller hangs up. As shown in the second example, it is more economical to test for business hours before queuing a call.

Relative Processing Cost of Vector Commands

Some vector commands use more processing resources than others. Table 29 and Table 30 show the relative processing costs of specific vector commands for a csi/si and an r switch, respectively. Whenever possible, use the lower cost vector commands. This will minimize your performance costs and upgrade your performance.

Table 29. Relative Processing Cost of Vector Commands for vs/csi/si Switch

relative performance cost	vector command
high	check
high	collect digits
high	queue-to
high	route-to
medium	announcement
medium	converse-on
medium	goto step
medium	goto vector
medium	messaging
low	busy
low	disconnect
low	stop
low	wait-time

Table 30. Relative Processing Cost of Vector Commands for r Switch

relative performance cost	vector command
medium	check
medium	collect digits
medium	converse
medium	goto vector (table comparison)
medium	messaging
medium	queue-to
medium	route-to
low	announcement
low	busy
low	disconnect
low	goto step
low	goto vector
low	stop
low	wait-time

Glossary and Abbreviations

A

abandoned call
An incoming call in which the caller hangs up before the call is answered.

ACD
See Automatic Call Distribution (ACD).

ACD agent
See agent.

ACW
See after-call work (ACW) mode.

ACD
See Automatic Call Distribution (ACD). ACD also refers to a work state in which an agent is on an ACD call.

ACD work mode
See work mode.

after-call work (ACW) mode
A mode in which agents are unavailable to receive ACD calls. Agents enter the ACW mode to perform ACD-related activities such as filling out a form after an ACD call.

agent
A person who receives calls directed to a split. A member of an ACD hunt group or ACD split. Also called an ACD agent.

agent report
A report that provides historical traffic information for internally measured agents.

agent selection method
The method the switch uses to select an agent in a hunt group when more than one agent is available to receive the next call: UCD-MIA, UCD-LOA, EAD-MIA, or EAD-LOA

Auto-In Work mode
One of four agent work modes: the mode in which an agent is ready to process another call as soon as the current call is completed.

Automatic Call Distribution (ACD)
A feature that answers calls, and then, depending on administered instructions, delivers messages appropriate for the caller and routes the call to an agent when one becomes available.

Automatic Call Distribution (ACD) split
A method of routing calls of a similar type among agents in a call group. Also, a group of extensions that are staffed by agents trained to handle a certain type of incoming call.

AUX-Work mode
A work mode in which agents are unavailable to receive ACD calls. Agents enter AUX-Work mode when involved in non-ACD activities such as taking a break, going to lunch, or placing an outgoing call.

B

BCMS
Basic Call Management System

C

call vector
A set of up to 15 vector commands to be performed for an incoming or internal call.

call work code
A number, up to 16 digits, entered by ACD agents to record the occurrence of customer-defined events (such as account codes, social security numbers, or phone numbers) on ACD calls.

CWC
See call work code.

D

DAC
Dial access code or Direct Agent Calling

DDC
Direct Department Calling

DNIS
Dialed-Number Identification Service

I

internal measurements
BCMS measurements that are made by the system.

M

Manual-In work mode
One of four agent work modes: the mode in which an agent is ready to process another call manually. *See* Auto-In Work mode for a contrast.

N

NQC
Number of queued calls

O

OCM
Outbound Call Management

othersplit
The work state that indicates that an agent is currently active on another split's call, or in ACW for another split.

Q

queue
An ordered sequence of calls waiting to be processed.

queuing
The process of holding calls in order of their arrival to await connection to an attendant, to an answering group, or to an idle trunk. Calls are automatically connected in first-in, first-out sequence.

R

redirection criteria
Information administered for each voice terminal's coverage path that determines when an incoming call is redirected to coverage.

Redirection on No Answer
An optional feature that redirects an unanswered ringing ACD call after an administered number of rings. The call is then redirected back to the agent.

report scheduler
Software that is used in conjunction with the system printer to schedule the days of the week and time of day that the desired reports are to be printed.

S

SAT
System access terminal

split
A group of ACD agents.

split condition
A condition whereby a caller is temporarily separated from a connection with an attendant. A split condition automatically occurs when the attendant, active on a call, presses the start button.

split number
The split's identity to the switch and BCMS.

split report
A report that provides historical traffic information for internally measured splits.

split (agent) status report
A report that provides real-time status and measurement data for internally measured agents and the split to which they are assigned.

staffed
Indicates that an agent position is logged in. A staffed agent functions in one of four work modes: Auto-In, Manual-In, ACW, or AUX-Work.

switch
Any kind of telephone switching system.

T

Tone detector
A device that collects and recognizes touch-tone digits entered from a telephone keypad. The switch uses the TN744 and TN2182 circuit packs for this function.

U

UCD
Uniform call distribution

V

vector directory number (VDN)
An extension that provides access to the Vectoring feature on the switch. Vectoring allows a customer to specify the treatment of incoming calls based on the dialed number.

vector-controlled split
A hunt group or ACD split administered with the vector field enabled. Access to such a split is possible only by dialing a VDN extension.

W

work mode
One of four states (Auto-In, Manual-In, ACW, AUX-Work) that an ACD agent can be in. Upon logging in, an agent enters AUX-Work mode. To become available to receive ACD calls, the agent enters Auto-In or Manual-In mode. To do work associated with a completed ACD call, an agent enters ACW mode.

work state
An ACD agent may be a member of up to three different splits. Each ACD agent continuously exhibits a work state for every split of which it is a member. Valid work states are Avail, Unstaffed, AUX-Work, ACW, ACD (answering an ACD call), ExtIn, ExtOut, and OtherSpl. An agent's work state for a particular split may change for a variety of reasons (example: when a call is answered or abandoned, or the agent changes work modes). The BCMS feature monitors work states and uses this information to provide BCMS reports.

Index

Symbols

sign, 67, 107, 108
  dialed ahead digits, 67
sign with digits, 104
* symbol
  dial-ahead digits, 103
* with digits, 104

A

abbreviated dialing special characters
  route-to, 130, 133
ACD
  capacities, 154
active VDN, 26
adapting
  to a long wait, 14
  to changing call traffic, 14
agents
  available
    definition, 20
  staffed
    definition, 20
  when available, 6
  when not available, 6
announcement command, 31
  classifications of, 37
  example, 38, 39, 40
  success/failure criteria, 140
  syntax, 95
  troubleshooting, 156
announcements, 95
  example, 38, 39
answer supervision considerations
  announcement, 96
  busy, 97
  check-backup, 100
  collect digits, 104
  converse-on, 108
  disconnect, 115
  goto step, 120
  goto vector, 124
  messaging, 126
  queue-to, 129
  route-to, 133
  stop, 136
  wait-time, 138

application
example
automated attendant, 71
basic call vectoring, 70, 75
call prompting, 71, 75
customer service center, 70
data in/voice answer, 75
data/message collection, 75
DIVA and data/message collection, 75
assigning call answering tasks to splits, 22
asterisk (*)
*, use of, 101
automating tasks, 15
availability of agents, 20

B

basic call vectoring
considerations, 149
hardware and software requirements, 146
BCMS, 97
function, 175
interactions with
busy, 97
check-backup, 100
converse-on, 114
disconnect, 116
messaging, 127
queue-to, 129
route-to, 135
reports
BCMS Split Report, 181
VDN Real-Time Report, 181
VDN Summary Report, 181
standards
for interpreting split flows, 177
benefits of call vectoring, 13
better utilization of agents, 19
branching, 9, 31
branching and programming, 30
busy, 31, 97
busy command
success/failure criteria, 140
syntax, 97
troubleshooting, 156

C

call flow method, 18
interflow, 18
intraflow, 18
multiple split queuing, 18

call flows
classes of, 175
converse-VRI calls, 189
defining and interpreting, 176
split inflows, outflows, and dequeues, 176
types that are tracked, 175
VDN inflows and outflows, 176
call group setup
current split operation worksheet, 207
customer needs worksheet, 208
guidelines, 203
key factors, 201
objectives worksheet, 206
vector design worksheet, 209
call not queued at stop step, 169
call prompting
call set, 56
command categories, 56
considerations, 151
digit entry, 58
entering variable length digit strings, 59
functions, 60
treating digits as a destination, 61
using digits on the agent's set, 63
using digits to collect branching information, 62
using digits to select options, 63
hardware and software requirements, 146
purpose, 55
removing incorrect digits, 58
variable length digit string, 58
with VRI, 55
call treatment
customizing, 15
personalization, 15
call vectoring
benefits, 13
capacities, 153
principles, xii
removing incorrect digits, 58, 66
upgrading to, 147
Call Vectoring Features Not Supported, 154
calling
during non-business hours, 12
CALLR-INFO button
format of display, 64
capacities
ACD, 154
call vectoring, 153
chaining of vector steps, 18
changing vectors, 3, 148
check-backup, 31
check-backup command, 10, 98
example, 45
neutral vector command, 100
success/failure criteria, 141
syntax, 98
troubleshooting, 156, 157

checking
availability of split, 11
queue capacity, 11
CMS
standards
for interpreting split flows, 177
collect digits, 19, 101
collect digits command, 31, 57
success/failure criteria, 141
syntax, 101
troubleshooting, 157
collecting and acting on information, 30
command category
for basic call vectoring, 35
for call prompting, 56
command table
for basic call vectoring, 35
for call prompting, 56
connecting to voice mail, 15
considerations
basic call vectoring, 149
call prompting, 151
control flow
type
conditional branching, 29
sequential flow, 28
unconditional branching, 29
converse VRI calls
call flow phase
data passing, 191
data return, 196
script completion, 199
script execution, 195
converse-on command, 31, 105
function, 106
success/failure criteria, 142
syntax, 105
troubleshooting, 159
converse-VRI calls
call flow phase
VRU data collection, 195
creating
a new vector, 3
customizing call treatment, 15, 20

D

defining desired service, 24
deleting
vector step, 4, 5
delivery of queued calls, 7
denying access, 25

digits, 58
collect digits
maximum number, 101
collected prior to timeout, 102
dial-ahead digits with *, 103
entering, 58
dial-ahead digits, 58, 60
variable-length digit strings, 59
including # sign, 103
maximum number, 103
removing
incorrect digit strings, 58
Touch-Tone, 102
with # sign, 104
with *, 104
disconnect command, 31, 115
example, 41
success/failure criteria, 142
syntax, 115
troubleshooting, 159
displaying digits on the agent's set, 60
during peak
calling periods, 19
heavy traffic, 10

E

editing, 4
enabling the vector disconnect timer, 147
encouraging caller to remain on-line, 8
entering
a command
in abbreviated form, 3
dial-ahead digits, 60
digits, 58
use of #, 59
variable-length digit strings, 58, 59
vector steps, 3
evaluating
effectiveness of vector programming, 175
performance, 175
split performance, 180
events, 166, 168
example application
split flow tracking, 177
VDN override, 26
example vector
accessing voice response scripts, 43
automated attendant application, 71
call interflow, 50
conditional branching, 52
customer service center application, 70
delay announcement, 38
delay with audible feedback, 39
dial-ahead digits, 65, 66
disconnecting a call, 41
DIVA and data/message collection application, 76

example vector, (continued)
emergency and routine service application, 79, 80
forced announcement, 38
information announcement, 39
late caller application, 82
leaving recorded messages, 47, 48
messaging options application, 84
multiple split queuing, 45
providing busy tone, 40
stopping vector processing, 53
supplementary delay announcement, 38
treating digits as a destination, 61
unconditional branching, 51
using digits to collect branching information, 62
using digits to select options, 63
example vector step
announcement, 95
check-backup, 98
collect digits, 101
converse-on, 105
disconnect, 115
goto step, 119
goto vector, 123
messaging, 125
queue-to, 128
route-to, 131
wait-time, 138
executing VRU scripts, 30

F

feature interactions
with announcement, 96
with busy, 97
with check digits, 104
with check-backup, 100
with converse-on, 109
with disconnect, 116
with goto step, 120
with goto vector, 124
with messaging, 127
with queue-to, 129
with route-to, 133
with stop, 136
with wait-time, 139
functions
of call prompting, 60

G

goto command
example, 51, 52
success/failure criteria, 142
troubleshooting, 159
goto step command, 31, 117
goto vector command, 31, 121

H

handling multiple calls, 21

I

improving
performance, 17, 211
service, 19, 22
the average speed of answer, 15
inserting vector steps, 4

L

latest VDN, 26
leaving a message, 13, 47
listing existing vectors, 3

M

maximizing performance, 211, 212, 214, 216
example vector, 212, 213, 214, 216
messaging, 31, 125
example, 48
leaving a message, 13
messaging command
example, 47
success/failure criteria, 143
syntax, 125
troubleshooting, 160
multiple call handling, 21

N

naming
a vector, 3
non-business hours
call during, 12
numbering
of vector steps, 5

O

option
VDN override, 26

P

passing digits
to an adjunct, 60
to switch, 25
performance
basic principles for improving, 211
evaluating, 175
effectiveness of vector programming, 175
for split, 180
improving, 212, 214, 216
example vector, 212, 213, 214, 216
looping, 211
maximizing, 211, 214, 216
, 212
processing cost
comparisons, 213, 215
of vector steps, 217, 218
testing vectors, 212
personalizing call treatment, 15
prioritizing calls, 7, 10, 15, 20
processing calls
faster, 13
intelligently, 13
prompting a caller, 26
properties, 24
providing
call treatments, 30, 36
caller feedback, 14
choices to callers, 14
faster service, 15
feedback, 7, 8, 9
initial feedback to caller, 21
night service, 14
security, 25

Q

queue-to command, 31, 128
queue-to main
neutral vector command, 129
queue-to main command
success/failure criteria, 144
syntax, 128
troubleshooting, 157
queuing calls
methods for, 18
to split, 19
maximum number of, 19
without call vectoring, 20

R

receiving feedback about a call, 7
redirecting calls
  methods for, 18
reducing
  caller hold time, 15
  number of needed agents, 22
  staffing requirements, 15
  transferred calls, 15, 19
removing incorrect digits strings, 58
reporting
  agent handling, 24
  call handling, 24
reports
  BCMS
    BCMS Split Report, 181
    VDN Real-Time Report, 181
    VDN Summary Report, 181
requeuing calls, 20
requirements
  software and hardware
    for basic call vectoring, 146
    for call prompting, 146
route-to command, 130
  summary of conditions for destination types, 183
  syntax, 130
  troubleshooting, 161
route-to digits, 31
route-to number, 31
routing calls, 15, 19, 30
  based on DNIS, 23
  overriding specifications, 26

S

security
  providing, 25
  with vector initiated service observing, 34
service observing, 28
silence, 30
  when occurs, 21, 22, 32
split
  backup
    definition, 19
  main
    definition, 19
staffed agent
  for ACD split, 20

staffed agents
basis of call management decisions, 20
check backup command, 20
conditional branching, 29
definition of, 20
for non-ACD hunt groups, 20
goto command, 20
number of, 31
status lamp, 64
CALLR-INFO button, 64
NORMAL button, 64
steps
maximum number of, 29
stop command, 32
example, 53
success/failure criteria, 144
syntax, 136
troubleshooting, 161

T

testing call treatment, 14
testing vectors, 148
tracking
calls, 175
example
split flow, 177
transfer call management control
caller-selected routing, 19
messaging, 19
treating digits as a destination, 60, 61
troubleshooting
1,000 step executed, 169
AAS split cannot queue, 174
administration change, 169
agent
drops converse, 173
all trunks busy on a quiet system, 161
alternate audio/music source not heard, 162
announcement not heard, 156, 159
while waiting for digits, 157
audible feedback
lasts longer than the delay interval, 155
longer than delay interval, 162
shorter than delay interval, 162
AUDIX link down, 174
branch is not made
to the specified step, 159
to the specified vector, 159
busy step for CO trunk, 172
busy tone, 160
call apparently answered in wrong order, 157
call cannot be queued, 170
call does not enter queue or terminate to agent, 156, 157
call dropped, 161, 169
call dropped by vector disconnect timer, 169
call stuck in converse, 163

troubleshooting, (continued)
caller information button denied, 158
Can't connect idle agent, 174
collect
announcement
not heard, 165
not heard and first collected digit incorrect, 158
collect step and announcement skipped, 157
converse
drop during data, 172
no prompt digits, 172
no qpos digits, 172
step skipped, 163
transfer denied, 173
coverage conference denied, 173
data return
no digits, 173
timeout, 173
delay before AUDIX answers, 160
delay before hearing announcement, 158
dial-ahead digits not recognized, 158
dial-ahead discarded, 170
digits incomplete, 164
double coverage attempt, 171
extra delay, 159
before hearing announcement, 156
first set of digits not collected, 163
incomplete announcement, 156, 158
insufficient digits collected
call routed to intercept, 158
messages not found, 160
messaging step failed, 171
music not heard, 162
network reorder, 161
no announcement available, 170
no available trunks, 171
no data returned from VRU, 159
no digits
collected, 172
to route-to, 171
no tone detector available, 170
not a messaging split, 174
not all digits returned to the switch, 165
not vector-controlled, 173
prompting buffer overflow, 171
queued to three splits, 170
redirect
of call failed, 172
unanswered call, 172
retrying announcement, 169
ringback heard instead of busy tone, 156
route -to step failed, 171
route-to step failed, 171
second set of digits
is the same as the first digits passed, 164
not collected, 164
split queue is full, 173

troubleshooting, (continued)
step skipped, 160
no message left, 160
that is, default treatment, 161
steps
display event report, 167
display events form, 166
system clock change, 173
time not set, 172
unexpected
silence after announcement, 156
unexpected intercept or reorder tone heard, 161
vector processing halted at collect step, announcement heard again upon return, 158
vector processing stops, 156
vector stuck, 155, 158, 159
with busy, 160
with ringback, 160
vector with no steps, 169
VRU script
not executed, 159
terminated prematurely, 159
wait digits not passed, 159
wait step
music failed, 172
ringback failed, 172

U

upgrading
to a call vectoring environment, 147
using digits
to collect branching information, 60
to select options, 60

V

valid entries
for check-backup, 98
for collect digits, 101
for converse-on, 105
for disconnect, 115
for goto step, 118
for goto vector, 122
for messaging, 125
for queue-to, 128
for route-to, 130
for wait-time, 137

VDN, 24
active, 26
definition, 17, 24
in coverage path
application uses, 27
latest, 26
override
example application, 26
properties
allow VDN override, 25
AUDIX name, 25
class of restriction (COR), 25
extension, 25
measured, 25
messaging server name, 25
name, 25
vector number, 25
vector
changing existing, 3, 148
creating a new, 3
definition, 24
disconnect timer, 147
editing, 4
events, 166, 168
example, 47
accessing voice response scripts, 43
automated attendant application, 71
call interflow, 50
conditional branching, 52
customer service center application, 70
delay announcement, 38
delay with audible feedback, 39
dial-ahead digits, 65, 66
disconnecting a call, 41
DIVA and data/message collection application, 76
emergency and routine service application, 79, 80
forced announcement, 38
information announcement, 39
late caller application, 82
leaving recorded message, 48
leaving recorded messages, 48
messaging options application, 84
multiple split queueing, 45
providing busy tone, 40
stopping vector processing, 53
supplementary delay announcement, 38
treating digits as a destination, 61
unconditional branching, 51
using digits
to collect branching information, 62
to select options, 63
listing existing, 3
naming, 3
testing, 148

vector applications
table of examples, 69
vector chaining
goto command, 54
multiple vectors, 54
purpose, 54
route-to, 54
vector command
announcement command, 95
announcements, 31
available with
call prompting, 89
call vectoring, 89
basic call vectoring, 35
command table, 35
busy, 31, 97
call prompting, 56
command table, 56
check-backup, 31, 98
collect digits, 31, 101
condition testing, 32
converse-on, 105
converse-on command, 31
deleting, 4
disconnect, 31
disconnect command, 115
function of each, 88
goto step, 31
goto step command, 117
goto vector, 31, 121
maximum number, 3
messaging, 31, 125
parameters, 90
queue-to, 31
queue-to command, 128
route-to, 130
route-to digits, 31
route-to number, 31
stop, 32
success/failure criteria, 140
syntax, 90
wait-time, 32, 137
vector directory number
definition, 17, 24
properties, 24
vector event
advantages of tracking unexpected, 166
displaying, 166, 167
logging of, 167, 168
range of type, 168
report, 167
tracking, 166
unexpected, 166
unique number, 168
with debugging, 163

vector processing
BCMS Report
description, 181
branching, 29, 30, 31
collecting from caller, 31
control flow, 17, 24
types of, 28
factors, 17
failure
converse-on step, 110
resulting in these destinations, 134
maximum number of steps, 29
programming
collecting and acting on information, 30
collecting from caller, 30
providing treatments, 30
routing calls, 30
programming capabilities
branching, 29
stopping, 17, 28, 29, 30, 35, 36, 40, 53
terminating, 48, 49, 50, 51
termination, 31
termination vs stopping, 29
troubleshooting, 155
VDN Real-Time Report
description, 181
VDN Summary Report
description, 181
with coverage, 27, 62
vector step
chaining, 18
conditional branching, 29
deleting, 5
entering, 3
example
announcement, 95
check-backup, 98
collect digits, 101
converse-on, 105
disconnect, 115
goto step, 119
goto vector, 123
messaging, 125
queue-to, 128
route-to, 131
wait-time, 138
inserting, 4
maximum number, 18
numbering, 5
sequential flow, 28
stopping, 29
terminating, 29
termination vs stopping, 29
unconditional branching, 29
vector-controlled split, 46, 48

voice response script, 35, 42
accessing, 43
checking amount of time for execution, 44
execution of, 43
interruption of, 44
VRI
advantage of, 42
capabilities, 42
description, 42
VRU, 42
activating a voice response script, 105
advantages of, 42
executing a script, 35
execution of VRU script, 42
normal override rules, 114
offloading recorded announcements to, 112
outpulsing data, 107, 108, 112
outpulsing to extension, 44
passing data between VRU and switch, 42
returning data to the switch, 106
service observing pending mode, 112
storing received data, 106
used as an external announcement, 42
VRU digits
conditional branching, 106
displayed via CALLR-INFO button, 106
extension in a route-to command, 106

W

wait-time, 32, 137
wait-time command
success/failure criteria, 144
syntax, 137
troubleshooting, 162
work mode
after-call-work mode, 20
auto-in work mode, 21
auxiliary-work mode, 21
manual-in work mode, 21