Programming Call Vectoring Features in Avaya Aura® Call Center Elite

Release 7.0.x
Issue 1
May 2016

© 2014-2016, Avaya, Inc.
All Rights Reserved.

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

“Documentation” means information published in varying mediums which may include product information, operating instructions and performance specifications that are generally made available to users of products. Documentation does not include marketing materials. Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of Documentation unless such modifications, additions, or deletions were performed by or on the express behalf of Avaya. End User agrees to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked websites referenced within this site or Documentation provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on Avaya hardware and software. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya’s standard warranty language, as well as information regarding support for this product while under warranty is available to Avaya customers and other parties through the Avaya Support website: https://support.avaya.com/helpcenter/getGenericDetails?detailId=C20091120112456651010 under the link “Warranty & Product Lifecycle” or such successor site as designated by Avaya. Please note that if You acquired the product(s) from an authorized Avaya Channel Partner outside of the United States and Canada, the warranty is provided to You by said Avaya Channel Partner and not by Avaya.

“Hosted Service” means an Avaya hosted service subscription that You acquire from either Avaya or an authorized Avaya Channel Partner (as applicable) and which is described further in Hosted SAS or other service description documentation regarding the applicable hosted service. If You purchase a Hosted Service subscription, the foregoing limited warranty may not apply but You may be entitled to support services in connection with the Hosted Service as described further in your service description documents for the applicable Hosted Service. Contact Avaya or Avaya Channel Partner (as applicable) for more information.

Hosted Service

THE FOLLOWING APPLIES ONLY IF YOU PURCHASE AN AVAYA HOSTED SERVICE SUBSCRIPTION FROM AVAYA OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE), THE TERMS OF USE FOR HOSTED SERVICES ARE AVAILABLE ON THE AVAYA WEBSITE, HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO UNDER THE LINK “Avaya Terms of Use for Hosted Services” OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, AND ARE APPLICABLE TO ANYONE WHO ACCESSES OR USES THE HOSTED SERVICE. BY ACCESSING OR USING THE HOSTED SERVICE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE DOING SO (HEREINAFTER REFERRED TO INTERCHANGEABLY AS “YOU” AND “END USER”), AGREE TO THE TERMS OF USE. IF YOU ARE ACCEPTING THE TERMS OF USE ON BEHALF A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS OF USE. IF YOU DO NOT HAVE SUCH AUTHORITY, OR IF YOU DO NOT WISH TO ACCEPT THESE TERMS OF USE, YOU MUST NOT ACCESS OR USE THE HOSTED SERVICE OR AUTHORIZE ANYONE TO ACCESS OR USE THE HOSTED SERVICE.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTPS://SUPPORT.AVAYA.COM/LICENSEINFO, UNDER THE LINK “AVAYA SOFTWARE LICENSE TERMS (Avaya Products)” OR SUCH SUCCESSOR SITE AS DESIGNATED BY AVAYA, ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AVAYA CHANNEL PARTNER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AVAYA CHANNEL PARTNER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA CHANNEL PARTNER; AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS “YOU” AND “END USER”), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE (“AVAYA”).

Avaya grants You a license within the scope of the license types described below, with the exception of Heritage Nortel Software, for which the scope of the license is detailed below. Where the order documentation does not expressly identify a license type, the applicable license will be a Designated System License. The applicable number of licenses and units of capacity for which the license is granted will be one (1), unless a different number of licenses or units of capacity is specified in the documentation or other materials available to You. “Software” means computer programs in object code, provided by Avaya or an Avaya Channel Partner, whether as stand-alone products, pre-installed on hardware products, and any upgrades, updates, patches, bug fixes, or modified versions thereto. “Designated Processor” means a single stand-alone computing device. “Server” means a Designated Processor that hosts a software application to be accessed by multiple users. “Instance” means a single copy of the Software executing at a particular time: (i) on one physical machine; or (ii) on one deployed software virtual machine (“VM”) or similar deployment.

License type(s)

Concurrent User License (CU). End User may install and use the Software on multiple Designated Processors or one or more Servers, so long as only the licensed number of Units are accessing and using the Software at any given time. A “Unit” means the unit on which Avaya, at its sole discretion, bases the pricing of its licenses and can be, without limitation, an agent, port or user, an e-mail or voice mail account in the name of a person or corporate function (e.g., webmaster or helpdesk), or a directory entry in the administrative database utilized by the Software that permits one user to interface with the Software. Units may be linked to a specific, identified Server or an Instance of the Software.

Heritage Nortel Software

“Heritage Nortel Software” means the software that was acquired by Avaya as part of its purchase of the Nortel Enterprise Solutions Business in December 2009. The Heritage Nortel Software is the software contained within the list of Heritage Nortel Products located at https://support.avaya.com/LicenseInfo under the link “Heritage Nortel Products” or such successor site as designated by Avaya. For Heritage Nortel Software, Avaya grants Customer a license to use Heritage Nortel Software provided hereunder solely to the extent of the authorized activation or authorized usage level, solely for the purpose specified in the Documentation, and solely as embedded in, for execution on, or for communication with Avaya equipment. Charges for Heritage Nortel Software may be based on extent of activation or use authorized as specified in an order or invoice.

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation, Software, Hosted Service, or hardware provided by Avaya. All content on this site, the documentation, Hosted Service, and the product provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software unless expressly authorized by Avaya. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil offense under the applicable law.

Virtualization

The following applies if the product is deployed on a virtual machine. Each product has its own ordering code and license types. Note that each Instance of a product must be separately licensed and ordered. For example, if the end user customer or Avaya Channel Partner would like to install two Instances of the same type of products, then two products of that type must be ordered.

Third Party Components

“Third Party Components” mean certain software programs or portions thereof included in the Software or Hosted Service may contain software (including open source software) distributed under third party agreements (“Third Party Components”), which contain terms regarding the rights to use certain portions of the Software (“Third Party Terms”). As required, information regarding distributed Linux OS source code (for those products that have distributed Linux OS source code) and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply is available in the products, Documentation or on Avaya’s website at: https://support.avaya.com/Copyright or such successor site as designated by Avaya. The open source software license terms provided as Third Party Terms are consistent with the license rights granted in these Software License Terms, and may contain additional rights benefiting You, such as modification and distribution of the open source software. The Third Party Terms shall take precedence over these Software License Terms, solely with respect to the applicable Third Party Components to the extent that these Software License Terms impose greater restrictions on You than the applicable Third Party Terms.

The following applies only if the H.264 (AVC) codec is distributed with the product. THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD (“AVC VIDEO”) AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

Service Provider

THE FOLLOWING APPLIES TO AVAYA CHANNEL PARTNER’S HOSTING OF AVAYA PRODUCTS OR SERVICES. THE PRODUCT OR HOSTED SERVICE MAY USE THIRD PARTY COMPONENTS SUBJECT TO THIRD PARTY TERMS AND REQUIRE A SERVICE PROVIDER TO BE INDEPENDENTLY LICENSED DIRECTLY FROM THE THIRD PARTY SUPPLIER. AN AVAYA CHANNEL PARTNER’S HOSTING OF AVAYA PRODUCTS MUST BE AUTHORIZED IN WRITING BY AVAYA AND IF THOSE HOSTED PRODUCTS USE OR EMBED CERTAIN THIRD PARTY SOFTWARE, INCLUDING BUT NOT LIMITED TO MICROSOFT SOFTWARE OR CODECS, THE AVAYA CHANNEL PARTNER IS REQUIRED TO INDEPENDENTLY OBTAIN ANY APPLICABLE LICENSE AGREEMENTS, AT THE AVAYA CHANNEL PARTNER’S EXPENSE, DIRECTLY FROM THE APPLICABLE THIRD PARTY SUPPLIER.

WITH RESPECT TO CODECS, IF THE AVAYA CHANNEL PARTNER IS HOSTING ANY PRODUCTS THAT USE OR EMBED THE G.729 CODEC, H.264 CODEC, OR H.265 CODEC, THE AVAYA CHANNEL PARTNER ACKNOWLEDGES AND AGREES THE AVAYA CHANNEL PARTNER IS RESPONSIBLE FOR ANY AND ALL RELATED FEES AND/OR ROYALTIES. THE G.729 CODEC IS LICENSED BY SIPRO LAB TELECOM INC. SEE WWW.SIPRO.COM/CONTACT.HTML. THE H.264 (AVC) CODEC IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO: (I) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD (“AVC VIDEO”) AND/OR (II) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION FOR H.264 (AVC) AND H.265 (HEVC) CODECS MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

Compliance with Laws

You acknowledge and agree that it is Your responsibility for complying with any applicable laws and regulations, including, but not limited to laws and regulations related to call recording, data privacy, intellectual property, trade secret, fraud, and music performance rights, in the country or territory where the Avaya product is used.

Preventing Toll Fraud

“Toll Fraud” is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or is not working on your company's behalf). Be aware that there can be a risk of Toll Fraud associated with your system and that, if Toll Fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Toll Fraud intervention

If You suspect that You are being victimized by Toll Fraud and You need technical assistance or support, call Technical Service Center Toll Fraud Intervention Hotline at +1-800-643-2353 for the United States and Canada. For additional support telephone numbers, see the Avaya Support website: https://support.avaya.com or such successor site as designated by Avaya.

Security Vulnerabilities

Information about Avaya’s security support policies can be found in the Security Policies and Support section of https://support.avaya.com/security.

Suspected Avaya product security vulnerabilities are handled per the Avaya Product Security Support Flow (https://support.avaya.com/css/P8/documents/100161515).

Downloading Documentation

For the most current versions of Documentation, see the Avaya Support website: https://support.avaya.com, or such successor site as designated by Avaya.

Contact Avaya Support

See the Avaya Support website: https://support.avaya.com for product or Hosted Service notices and articles, or to report a problem with your Avaya product or Hosted Service. For a list of support telephone numbers and contact addresses, go to the Avaya Support website: https://support.avaya.com (or such successor site as designated by Avaya), scroll to the bottom of the page, and select Contact Avaya Support.

Trademarks

The trademarks, logos and service marks (“Marks”) displayed in this site, the Documentation, Hosted Service(s), and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, its licensors, its suppliers, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the Documentation, Hosted Service(s) and product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Avaya, the Avaya logo, Avaya one-X® Portal, Communication Manager, Application Enablement Services, Modular Messaging, and Conferencing are either registered trademarks or trademarks of Avaya Inc. in the United States of America and/or other jurisdictions.

All non-Avaya trademarks are the property of their respective owners. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Contents

Chapter 1: Introduction ... 14
Purpose ... 14
Chapter 2: Call Vectoring fundamentals ... 15
Limitations of traditional ACD call processing ... 15
Call management ... 18
Call flow ... 18
Caller control ... 19
Split queue priority levels ... 19
Call queuing to splits ... 20
Agent work mode ... 20
Calling party feedback ... 21
Dialed Number Identification Service ... 22
Vector processing ... 22
Vector Directory Number ... 22
VDN variables ... 23
VDN Time Zone Offset ... 23
VDN Override ... 24
VDN in a Coverage Path ... 24
RONA to a VDN ... 25
Observing VDNs ... 25
Vector control flow ... 26
Termination versus stopping ... 26
About Call Vectoring commands ... 27
Call Vectoring commands ... 27
Call Vectoring benefits ... 30
Chapter 3: Call Vectoring examples ... 32
Customer service center ... 32
Automated attendant ... 34
Data in/voice answer and data/message collection ... 34
Distributed call centers ... 37
Help desk ... 39
Insurance agency ... 39
Warranty service with EAS ... 42
Notify callers about queue position ... 45
Scenario solution ... 46
Resort reservation service ... 47
Specific number dialing ... 47
General number dialing ... 48
Callback provisions ... 48

Attendant routing example.... 49
Vector administration.... 50
Local attendant group access code.... 50
Incoming trunk calls to attendant group.... 51
Incoming LDN calls.... 51
QSIG Centralized Attendant Service.... 52
CAS branch.... 52
CAS main.... 52
Night station service.... 53
Holiday Vectoring example.... 54
NCR example.... 56
Primary vector for NCR example.... 57
Status poll vector for NCR example.... 57
Interflow vector for NCR example.... 57
BSR using EWT and agent adjustments example.... 58
Primary vector for BSR using EWT and agent adjustments example.... 59
Status poll vector for BSR using EWT and agent adjustments example.... 59
Interflow vector for BSR using EWT and agent adjustments example.... 60
Dial by Name.... 60
Use of vectors in business scenarios.... 62
Emergency and routine service.... 63
Late call treatment suggested solution.... 64
Messaging option.... 66
Chapter 4: How to improve performance.... 67
Looping examples.... 68
Audible feedback examples.... 68
Look-Ahead Interflow examples.... 69
Check examples.... 70
After business hours example.... 72
Look-Ahead Interflow example.... 72
Chapter 5: Basic Call Vectoring.... 74
Basic Call Vectoring command set.... 74
Treatment commands.... 75
Routing commands.... 75
Branching or programming commands.... 75
Basic Call Vectoring considerations.... 75
Call Vectoring feature availability.... 76
Chapter 6: Variables in Vectors.... 80
Variable parameters.... 80
Implementing vector variables.... 81
Command syntax for vector variables.... 83
Announcement command.... 83
Collect command.... 83

Converse-on command.... 84
Disconnect command with vector variables.... 85
Goto commands.... 85
Route-to number command.... 87
Set command.... 88
Wait command.... 89
VIV requirements.... 89
Definition of local, global, and local persistent variables.... 89
About local variables.... 90
About global variables.... 90
About local persistent variables.... 91
System-assigned vector variable types.... 91
System-assigned definition.... 91
ANI type variable.... 91
ASAIUUI type variable.... 92
DOW type variable.... 93
DOY type variable.... 94
Stepcnt type variable.... 94
TOD type variable.... 95
VDN type variable.... 96
VDNTime type variable.... 97
User-assigned vector variable types.... 98
User-assigned definition.... 99
Collect type variable.... 99
Value type variable.... 102
VIV interactions and considerations.... 103
VIV administration.... 104
Example Variables for Vectors screen.... 104
Required variable administration entries.... 104
Performing optional FAC administration for value variables.... 106
VIV job aid.... 106
VIV vector examples.... 108
Example application using time and day variables.... 108
Example application using a value variable.... 111
Example applications using global collect variables.... 112
Example applications using vdn type variables.... 114
Example application using a vector variable in other commands.... 115
Example application using a vector variable in the converse-on command.... 115
Chapter 7: VDN variables.... 117
VDN variable fields.... 117
Using VDN variables with vector commands.... 118
Announcement command.... 118
Converse-on command.... 118

Disconnect command.... 118
Goto commands.... 119
Route-to command with VDN variables.... 120
Set command with VDN variables.... 121
Wait command with VDN variables.... 121
Case studies.... 121
Using one vector for different announcements.... 121
Business case.... 122
Chapter 8: Vector subroutines.... 126
The goto command and subroutines.... 126
The @ step parameter.... 127
Example 1: Test for working hours.... 127
Incoming call processing vector example.... 127
Checking working hours vector subroutine example.... 128
Chapter 9: ANI/II-digits routing and CINFO.... 129
CINFO command set.... 129
ANI routing.... 130
ANI basics.... 130
Use of ANI with Vector Routing Tables.... 131
Use of ANI without Vector Routing Tables.... 132
II-digits routing.... 132
II-digits basics.... 133
II-digits codes.... 134
II-digits routing example.... 137
CINFO.... 138
CINFO basics.... 138
CINFO vector example.... 140
CINFO interactions.... 140
Chapter 10: Multi-National Calling Party Number prefixes.... 142
CPN prefix routing example.... 142
Chapter 11: How to create and edit call vectors.... 144
Call Vector screen basic administration.... 144
How to view vector variable information.... 146
Viewing vector variable information.... 146
Variable display fields.... 147
Variable display examples.... 148
Inserting a vector step.... 149
Deleting a vector step.... 150
Entering a comment out indication to an existing vector step.... 150
Removing a comment out indication.... 151
How to create and construct a vector.... 152
Step 1: Queuing a call to the main split.... 152
Step 2: Providing feedback and delay announcement.... 153

Step 3: Repeating delay announcement and feedback.... 154
Step 4: Queuing a call to a backup split.... 155
Step 5: Limiting the queue capacity.... 156
Step 6: Checking for non business hours.... 157
About duplicate VDNs.... 157
Creating duplicate VDNs.... 158
About duplicate vectors.... 158
Creating duplicate vectors.... 158
Removing calls from queues.... 159
Chapter 12: Vector management.... 160
3.0 Enhanced Vectoring requirements.... 160
Adjunct Routing requirements.... 160
Advanced Vector Routing requirements.... 160
ANI/II-Digits requirements.... 161
Basic Call Vectoring requirements.... 161
Call Prompting requirements.... 161
CINFO requirements.... 162
G3V4 Enhanced Vectoring requirements.... 162
Holiday Vectoring requirements.... 162
Look-Ahead Interflow requirements.... 162
Network Call Redirection requirements.... 163
Variables in Vectors requirements.... 163
VDN variables requirements.... 163
Vectoring (Best Service Routing) requirements.... 163
Administering **Vector Disconnect Timer**.... 164
Changing and testing a vector.... 164
Identifying links to a vector.... 165
Finding all occurrences of a digit string.... 165
Chapter 13: Call Vectoring commands.... 167
About Avaya Call Center packages.... 167
Communication Manager options required to enable vector commands.... 167
Vector command description.... 170
command.... 171
adjunct routing link command.... 173
The adjunct routing link process.... 173
adjunct routing link command feature interactions.... 177
adjunct routing link command interactions with CMS.... 177
adjunct routing link command interactions with BCMS.... 179
announcement command.... 179
Basic operation for the announcement command.... 179
announcement command considerations.... 180
Delay announcements.... 180
Forced announcements.... 181

Information announcements.... 181
Recording announcements.... 181
Considerations for DTMF transfer and connect applications.... 183
Answer supervision considerations.... 183
busy command.... 184
Answer supervision considerations for the busy command.... 184
busy command feature interactions.... 185
busy command interactions with CMS.... 185
busy command interactions with BCMS.... 185
check command.... 185
check split command.... 187
check skill for available agents with level preference.... 188
Answer supervision considerations for the check command.... 189
check command feature interactions.... 189
check command interactions with CMS.... 189
check command interactions with BCMS.... 190
collect digits command.... 191
Answer supervision considerations with the collect digits command.... 194
collect digits command feature interactions.... 194
collect digits command interactions with CMS/BCMS.... 195
consider command.... 195
User adjustments.... 196
Events that clear best data.... 196
consider command considerations.... 197
Answer supervision considerations for the consider command.... 197
consider command feature interactions.... 198
consider command interactions with CMS/BCMS.... 198
converse-on command.... 198
Call flow and specifications for converse - VRI calls.... 200
Data 1 and Data 2 values administered within the converse-on command.... 209
converse-on split command.... 210
Answer supervision considerations for the converse-on split command.... 212
converse-on split command feature interactions.... 213
converse-on split command interactions with CMS.... 217
converse-on split command interactions with BCMS.... 217
disconnect command.... 217
Answer supervision considerations for disconnect command.... 218
disconnect command feature interactions.... 218
disconnect command interactions with CMS.... 219
disconnect command interactions with BCMS.... 219
goto step and goto vector command.... 219
goto step and goto vector commands operation.... 223
Time adjustments using goto conditionals.... 225

Comparing none, #, and numeric digits..... 226
Media gateway, port network, and server vector conditionals..... 227
messaging command..... 230
Using a messaging step in a vector..... 231
Leaving a recorded message..... 231
Answer supervision considerations for the messaging command..... 232
messaging command feature interactions..... 232
messaging command interactions with CMS..... 233
messaging command interactions with BCMS..... 233
queue-to command..... 234
queue-to split command..... 236
Answer supervision considerations for the queue-to command..... 238
queue-to command feature interactions..... 238
queue-to command interactions with CMS..... 239
queue-to command interactions with BCMS..... 239
reply-best command..... 240
Answer supervision considerations for the reply-best command..... 240
reply-best command interactions with CMS/BCMS..... 241
return command..... 241
When return destination information is not stored..... 241
Memory full conditions..... 242
route-to command..... 242
Operation details for the route-to command..... 244
Conditional route-to statements..... 248
Destinations for the route-to command..... 248
Command completion and failures..... 249
About the number field..... 250
Abbreviated Dialing special characters..... 250
Using the route-to command for NCR..... 251
Coverage parameter..... 252
route-to number command..... 252
Answer supervision considerations for the route-to command..... 254
route-to command feature interactions..... 254
route-to command interactions with CMS..... 257
route-to command interactions with BCMS..... 258
set command..... 258
Variable, digits buffer, and asaiuui..... 259
set command considerations..... 264
Advanced set command rules and applications..... 265
set command examples..... 271
stop command..... 288
Answer supervision considerations for the stop command..... 289
stop command feature interactions..... 289

stop command interactions with CMS ... 289
stop command interactions with BCMS ... 289
wait-time command ... 289
wait-time command basic operation ... 290
Call delay with audible feedback ... 291
Multiple audio or music sources on delay ... 291
Call delay with continuous audible feedback ... 292
Multiple music sources on hold ... 292
wait-time command considerations ... 293
Chapter 14: How to improve performance ... 296
Looping examples ... 297
Audible feedback examples ... 297
Look-Ahead Interflow examples ... 298
Check examples ... 299
After business hours example ... 301
Look-Ahead Interflow example ... 301
Chapter 15: Call Vectoring job aid ... 303
Vector commands job aid ... 303
... 303
adjunct routing link ... 303
announcement ... 304
busy ... 304
check ... 304
collect digits ... 305
consider ... 305
converse-on ... 305
disconnect ... 305
goto step and goto vector ... 306
messaging ... 309
queue-to ... 309
reply-best ... 309
return ... 310
route-to ... 310
set ... 311
stop ... 312
wait-time ... 312
Vector variables job aid ... 313
ANI ... 313
ASAIUUI ... 313
Collect ... 313
DOW ... 314
DOY ... 314
Stepcnt ... 314

TOD..... 314
Value..... 315
VDN..... 315
VDNTime..... 315
Chapter 16: Resources..... 316
Documentation..... 316
Finding documents on the Avaya Support website..... 316
Training..... 317
Viewing Avaya Mentor videos..... 317
Support..... 318
Glossary..... 319

Chapter 1: Introduction

Purpose

The document describes how to create and edit call vectors. The document also describes Call Vectoring commands.

This document is intended for implementation engineers and system administrators.

Chapter 2: Call Vectoring fundamentals

Call Vectoring is the process of defining vector programs for call routing and call treatment.

Call vectors are a series of user-defined commands that you can use to route internal or network calls and to determine the treatment for each call. You can route calls to on-network or off-network destinations, or to staffed ACD agents.

Call processing depends on how you implement Avaya Aura® Communication Manager and the Call Vectoring software. The success of call processing depends on the following factors:

- Call management: Availability of resources, such as agents, splits, software, and hardware, to process a call.
- Vector processing: The method of processing calls which includes VDN usage, vector control flow, and intelligent use of the vector programming capabilities.

Note:

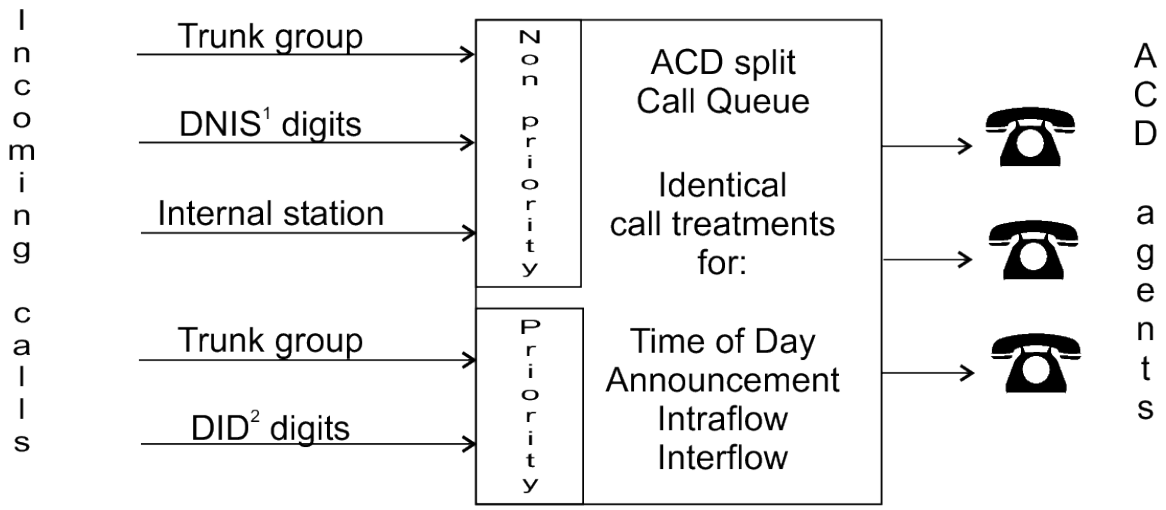
The document contains sample vectors that illustrate vectoring features and capabilities. You must customize the samples before use.

Related links

Limitations of traditional ACD call processing on page 15
Call Vectoring benefits on page 30

Limitations of traditional ACD call processing

In the traditional Automatic Call Distribution (ACD) approach, all calls within a queue receive identical announcements and intraflow parameters. The following figure depicts a simplified version of the traditional ACD call processing approach.



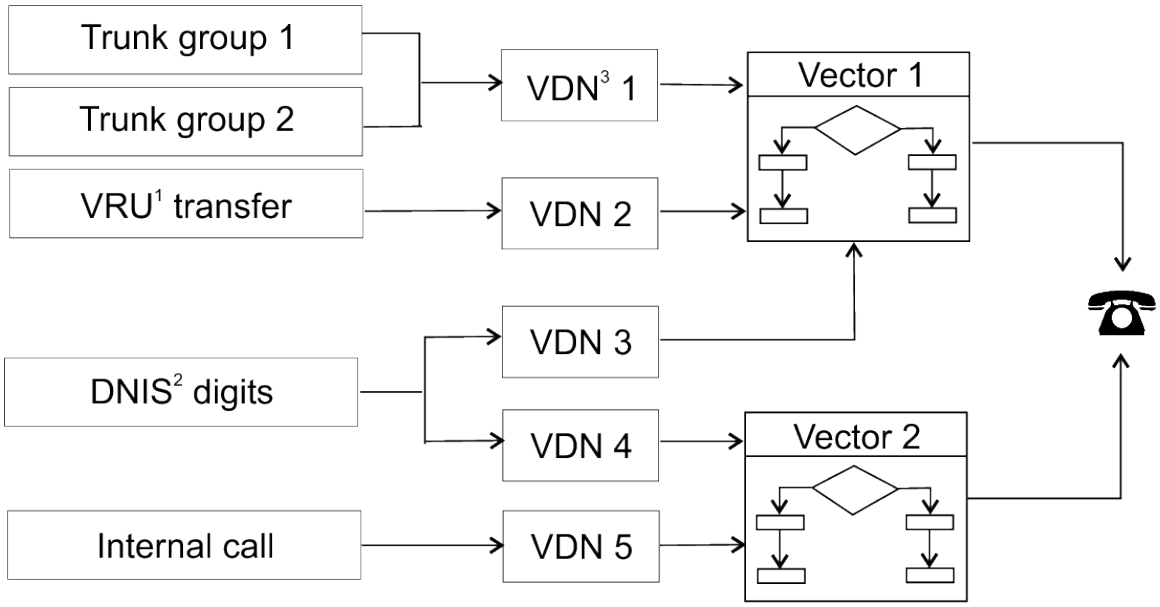
1. Dialed Number Identification Service (DNIS)
2. Direct Inward Dialing (DID)

With Call Vectoring, each call is unique and the call treatment is based on a number of factors, including the number that the caller dials, the number that the caller calls from, the number of calls in queue, the time of the day, and the day of the week. Call Vectoring also applies to calls that are handled by the same agent group.

Call Vectoring comprises the following basic components:

- Vector Directory Numbers (VDN)
- Vectors
- Vector commands

The components direct incoming calls and ASAI event reports and requests to the destinations. The components also specify the call treatment. You can set up Call Vectoring as shown in the following figure.



1. Voice Response Unit (VRU)
2. Dialed Number Identification Service (DNIS)
3. Vector Directory Number (VDN)

If you set **`Call Vectoring`** to `y`, Communication Manager directs the incoming call to a VDN, which is a soft extension number that directs the call to a specific vector. The VDN represents a call type or category, for example billing or customer service. The VDN defines the type of service required by the caller. You can use multiple VDNs to point to the same vector or to different vectors, depending on whether you want the relevant calls to receive the same or different treatment.

The following is an example of the function of a vector:

```
1. goto step 3 if calls-queued in split 9 pri l < 20
2. busy
3. queue-to split 9 pri l
4. wait-time 12 seconds hearing ringback
5. announcement 2921
6. wait-time 998 seconds hearing music
```

A vector can contain up to 99 command steps. You can link multiple vectors to extend the processing capabilities or to process calls to the same or different destinations. Any number of calls can use the same multiple vectors.

Related links

Call Vectoring fundamentals on page 15

Call management

When a Vector Directory Number (VDN) directs an incoming call to a vector, the commands in the vector determine call routing and call treatment. Processing starts at the first step and proceeds through the vector steps. Processing skips empty steps and stops after execution of the last step.

One vector can direct the call to another vector or VDN, which in turn can direct the call to another vector. The call direction between vectors can continue up to a maximum of 10,000 vector steps for each call. When a call enters vector processing, a loop counter tracks the number of executed vector steps. If the loop counter exceeds 10,000, the system executes the **`stop`** command to terminate processing of the vector.

Call flow

Calls enter a vector and execute the steps sequentially beginning with Step 1 of the vector, unless there is a **`goto`** step. Most steps take microseconds to execute. Steps with **`announcement`**, **`wait-time`**, and **`collect digits`** commands are exceptions. A 1-second wait occurs when processing is suspended. Note that the wait-time with 0 seconds is not an explicit wait. The announcement, wait-time > 0 and collect digits are explicit wait steps which suspend vector processing. The 15–step counter resets after one of these steps.

Multiple split queuing

You can queue a call to up to three splits using multiple split queuing.

Intraflow

You can use Intraflow to redirect calls that are unanswered at a split within a predefined time to more than one other split on the same Communication Manager. If redirection depends on a condition, the process is called conditional intraflow.

Interflow

You can use Interflow to redirect calls that are directed to a vector to an external or non local split destination. This destination is represented by a number that is programmed in the relevant vector. Calls can be routed to an attendant or attendant queue, a local extension, a remote extension, that is, Uniform Dialing Plan (UDP), an external number, or a VDN.

Look Ahead Interflow (LAI)

Implement LAI for call centers with multiple ACD locations that are connected by way of ISDN PRI. With this method, a call can interflow only if a remote location is better equipped to handle the call. LAI occurs only when the conditions at the receiving Communication Manager are met.

Best Service Routing (BSR)

With BSR, Communication Manager can compare specified splits or skills, identify the split or skill that provides the best service to a call, and deliver the call to the resource. Communication Manager queues the call queues if agents are unavailable in the identified split or skill.

BSR is available in singlesite and multisite versions. Singlesite BSR compares splits or skills on Communication Manager to find the best resource to service a call. multisite BSR extends this

capability across a network of Communication Manager, comparing local or remote splits or skills and routing calls to the resource that provides the best service.

Adjunct Routing

With adjunct routing, Communication Manager requests a routing destination from an adjunct processor. When you enable this feature, Communication Manager sends a message with information on the calling party to the ASAI adjunct. The adjunct determines from the databases the best place for Communication Manager to send the call and passes the routing information back to Communication Manager.

Caller control

You can use the following methods to enable temporary transfer of call management control to the caller:

Caller-selected routing

With this method, a caller can enter information in the form of dialed digits from a touchtone phone or from an internal rotary phone that is located on the same Communication Manager. The capability is available if you set **Call Prompting** to `y`. A recorded announcement is used for prompting. Once the caller enters the digits, Communication Manager routes the call to the correct destination. The caller-selected routing method reduces the number of transferred calls significantly, enhancing customer experience.

If you set **Call Prompting** and **Call Vectoring (CINFO)** to `y`, a vector collects the caller entered digits (ced) that are passed from the network in an ISDN message. You can use the digits to enhance caller control in the same way as the digits that Communication Manager collects directly do.

Messaging

The caller can choose to leave a voice message if the call is not answered. When you set **Messaging** to `y`, control passes to the messaging system split.

Split queue priority levels

If you set **Call Vectoring** to `n`, Communication Manager queues calls at one of the following two priority levels: Medium or High. If you set **Call Vectoring** to `y`, Communication Manager queues the call to one of the following four priority levels: Top, High, Medium, or Low. Within each priority level, calls are processed sequentially.

Communication Manager assigns a higher priority to direct agent call and delivers the call before directing the call to a split. The exception is when you set **Call Handling Preference** to `y` and not administer the skill that receives direct agent call as the highest skill level of the agent. A direct agent call is an ACD call that is directed to a specific agent, and not to any available agent in the split.

Note:

If a call is already queued to more than one split that serves as a backup split, Communication Manager requeues the call at the new priority level that is indicated in the vector command step.

Call queuing to splits

Basic Call Vectoring simultaneously queues calls to up to three splits at any one of the four priority levels. This process is called multiple split queuing. The first split to which a call queues is the main split and the second and third splits are designated as backup splits. With multiple split queuing, you can utilize agents efficiently and provide better service to callers.

The following events occur when an agent becomes available in any split to which the call queues:

- The call connects to the agent.
- Communication Manager removes the call from the other queues and terminates announcements, music, ringback, or other audio sources.
- Vector processing terminates.

Related links

Multiple split queuing on page 236

Agent work mode

Use Call Vectoring to make call management decisions based on the following real-time agent work modes:

- Staffed-agents: Agents logged in to an ACD split.
- Available-agents: Agents logged in and ready to receive an ACD call.

The agent work modes appear as conditions within the **`check split`** and **`goto`** commands. You can use the commands to check the number of staffed or available agents.

If a hunt group is not monitored:

- Agents in the hunt group do not have login, logout, or work modes.
- Staffed-agents are synonymous with administered.
- The available agents work mode is the number of agents ready to receive a hunt group call.

For ACD calls, relevant work mode defines agent states. The following list describes the modes.

After Call Work (ACW)

The agent is unavailable to receive an ACD call for any split. Use the mode when the agent performs ACD call-related activities. You can implement the mode on a timed basis, in which case the mode is known as Timed ACW. Communication Manager automatically places the agent in the ACW mode after the agent completes a call while in the manual-in work mode. You can place agents automatically in the ACW mode for an administered period of time following the completion

of each ACD call. Administer the changes on the Vector Directory Number (VDN) and Hunt Group screens.

Auto-In Work

An agent is available to receive calls. With auto-in work mode, the agent can receive a new ACD call immediately after disconnecting the previous call. When you enable **Multiple Call Handling**, an agent in the auto-in work mode can choose to receive an ACD call by placing the active call on hold.

Auxiliary-Work

An agent is unavailable to receive an ACD call for the specified split. Use the mode when the agent performs activities that are not associated with the ACD, such as taking a break.

Manual-In Work

The agent is available to receive calls. After the agent disconnects from an ACD call, the agent is automatically placed in the ACW mode. When you administer Multiple Call Handling (MCH), an agent in the manual-in work mode receives additional ACD calls by placing an active call on hold.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

Calling party feedback

The caller hears a feedback as a vector processes the call. The feedback depends on one of the following origin classification of the call:

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

For an internal or a non CO call, the caller hears no feedback until the call reaches one of the following vector steps:

- For a **`wait`** command with the system music, ringback, an alternate music or audio source, the caller hears the system music, ringing, or the music or audio associated with an administered port.
- For an **`announcement`** command, the caller hears the specified announcement.
- For a **`busy`** command, the caller hears a busy signal.
- When the call rings at a station, the caller hears a ringback tone.

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

- Announcement.
- Wait with system music or an alternate music or audio source.
- Call answered. The caller hears the agent or Voice Response Unit (VRU) answering the call.

For a CO call that includes answer supervision after processing of an **announcement** or a **wait-time** command, the caller can hear any of the following:

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

Dialed Number Identification Service

In the traditional ACD call processing approach, each agent in a split is trained to answer calls for a specific purpose such as technical support. However, a call center can train agents to address multiple types of calls. For example, if you have three splits with five agents each, that is, 15 agents in total, to handle three types of calls, you can use only 11 or 12 agents in a combined split.

Dialed Number Identification Service (DNIS) is a network service that identifies the number the caller dials and passes the touchtone or Dual Tone Multi Frequency (DTMF) digits to Communication Manager. The unique number can be sent to an agent, a host computer with ASAI applications, or used to provide different call treatment methods.

Call Vectoring receives the DNIS number from the network and interprets the number as a VDN. When Communication Manager delivers the call to the agent terminal, the unique name assigned to the VDN appears on the agent terminal indicating the type of response required from the agent.

Vector processing

If you use Call Vectoring, more than one programmed sequence of commands called vectors processes the calls.

Vector processing includes the following:

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

Vector Directory Number

Communication Manager receives the digits dialed by the caller from the network and translates the dialed digits as a VDN which defines the service required by the caller. The VDN also serves as the application number that Communication Manager sends to the reporting adjuncts for call and agent handling statistical report generation.

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

VDN implementation notes

The following list describes special situations due to the type of Communication Manager implementation that causes differences in the available fields on the VDN screen.

- The **Data for the Orig Annc** column is available only when you enable **VDN of Origin Announcement** on the System-Parameters Customer-Options screen.
- To list all VDNs using the same BSR application plan, enter the **`list VDN BSR xxx`** command, where xxx is the number of the BSR application plan used by more than one VDN.

You can assign VDNs to incoming trunk groups or send the VDNs in digit form to Communication Manager by a public or private network. Digits sent to Communication Manager can come from the serving CO or toll office through DID or DNIS. The digits can also come from another location through dial-repeating tie trunks, or an internal caller can dial the digits. For a non-ISDN or non-SIP call, the last four digits of the number are sent to the system. For an ISDN or SIP call, the entire 10-digit number is sent to the system.

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

You can administer the parameters you require on the Vector Directory Number screen.

VDN variables

With VDN variables, you can use VDNs with a smaller set of vectors.

VDN Time Zone Offset

VDN Time Zone Offset is designed for call centers with locations in different time zones. You can program a single vector with Time of Day (TOD) conditional steps that handle each time zone based on the *active* VDN for the call.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

VDN Override

VDN Override changes the active VDN for a call. The active VDN defines the VDN used for parameters that are associated with the call, such as the VDN name, VDN skills, Tenant Number (TN), BSR application, and VDN variables.

You can use VDN Override to make the routed-to VDN become the active VDN by routing the call through Policy Routing Tables (PRTs) or **`route-to number/digits`** vector command. To use VDN Override, administer the **Allow VDN Override** field on the Vector Directory Number (VDN) screen.

When a call is placed or routed to a VDN, Communication Manager defines the first VDN that receives the call as the active VDN for the call. If the **Allow VDN Override** field for that VDN is set to no, routing the call to a subsequent VDN does not change the active VDN for the call. However, if the field is set to yes, routing the call through a **`route-to number`** or **`route-to digits`** step in the assigned vector or through the assigned PRT instead of a vector, for example Percentage Allocation PRT, Communication Manager changes the routed to VDN extension as the active VDN.

VDN overriding can be sequential if the next routed to VDN has the field set to yes. For example, if VDN1 has the **Allow VDN Override** field set to no, VDN2 has the field set to yes, and VDN3 has the field set to no, VDN3 is the active VDN for the call when the call routed to VDN4. If VDN4 has the field set to yes and the call is then routed to VDN5, VDN5 becomes the active VDN.

Besides defining what VDN to use to obtain the Vector Directory Number (VDN) screen field settings, you can specify the active VDN as a keyword in some vector commands. When a vector step with the keyword *active* is executed, Communication Manager replaces the VDN extension for the *active* VDN, as defined by the VDN Override rule, with the keyword when Communication Manager processes the vector command.

Use the keyword *active* as:

- The VDN extension for the **`goto`** command **`counted-calls`** conditional
- The **`goto`** command **`rolling-asa for VDN`** conditional
- The **`messaging`** command mailbox extension
- The VDN vector variable type assignment

You can also assign the keyword *latest*, that is, the last VDN routed-to in the same vector commands or variable, but the VDN Override settings do not change the *latest* VDN.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

VDN in a Coverage Path

You can assign a VDN as the last point in a coverage path. When you assign a VDN, an incoming call is directed to the coverage point and Call Vectoring or Call Prompting processes the call if either field is enabled. You can assign call coverage to an external location or the caller can control the type of coverage.

You can use VDN in a Coverage Path (VICP) for the following types of applications:

- Sending direct agent calls or personal calls to an agent in an Expert Agent Selection (EAS) environment.
- Routing coverage calls off-premises using 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 an AUDIX™ at night.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

Related links

Option with VDN as the coverage point on page 237

RONA to a VDN

Redirection on No Answer (RONA) redirects a ringing ACD call after an administered number of rings. RONA prevents a call from ringing indefinitely at a terminal when an agent does not answer the call. When a call is redirected, Communication Manager puts the agent in the Auxiliary (AUX) work mode and the agent is unavailable to receive ACD calls. In the case of Auto-Available Splits (AAS), Communication Manager redirects the call and logs the agent out.

You can administer a VDN as the destination of a RONA processed call. On the Hunt Group screen, enter the destination VDN for a RONA call in the **Redirect to VDN** field. All calls that are redirected by RONA are sent to the same administered VDN. If you do not administer a destination VDN, but enter the number of rings for a redirection, Communication Manager redirects the call back to the split or skill.

Unanswered direct agent calls follow the coverage path that you administer for the agent. If you do not administer a coverage path, Communication Manager redirects the direct agent calls to the VDN that you administer as the first primary skill of the agent.

You can also set up a generic VDN to process calls that are redirected due to RONA, ROIF, and ROOF.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

Observing VDNs

With Service Observing, supervisors can observe VDNs. A supervisor can select a VDN and bridge onto the call that has just entered vector processing for the VDN. The supervisor can observe one call at a time. The observer hears all tones, announcements, music, and speech that the caller and the agent hear and say, including call prompting and caller dialing. The observer also hears the VDN of Origin Announcements (VOAs). Communication Manager makes an observing connection to a call in vector processing and maintains the connection throughout the life of the call until the call is disconnected or the observer hangs up. Communication Manager maintains the connection even if the call is routed or transferred externally.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

Vector control flow

The vector process starts at the first step in the vector and proceeds sequentially through the vector unless the process encounters a **goto** command. The vector process skips any blank steps and automatically stops after processing the last step in the vector.

Call Vectoring provides the following three types of control flow that pass vector processing control from one vector step to another:

- Sequential: The control flow passes vector processing control from the current vector step to the following step. Most vector commands enable a sequential flow through the vector.

  Note:

  For any vector command that fails, the control automatically passes to the following step.

- Unconditional branching: The control flow unconditionally passes control from the current vector step to either a preceding or a succeeding vector step or to another vector. For example, **goto step 6 if unconditionally**.
- Conditional branching: The control flow conditionally passes control from the current vector step to either a preceding or a succeeding vector step or to another vector. This type of branching is based on the testing of threshold conditions. For example, **goto vector 29 @step 1 if staffed-agents in split 6 < 1**.

Call Vectoring has an execution limit of 10,000 steps. Once a call enters vector processing, a loop counter tracks the number of executed vector steps. If the loop counter exceeds 10,000, a **stop** command executes to end the vector process.

Termination versus stopping

When vector processing terminates, the call leaves the vector. Vector termination can result from a number of events such as when a call is:

- Ringing at an agent station.
- Abandoned by the calling party.
- Subject to a forced disconnect or busy command.
- Successfully routed to an extension or to an off-premises number.

The termination of vector processing differs from stopping. The **stop** command or executing the final step in the vector causes stopping. Termination differs from stopping in the following ways:

- If a call is queued, termination removes the call from the queue.
- A **stop** command prevents the processing of new vector steps but leaves the call in queue, and the calling party continues to receive feedback, such as a ringback.

- If vector processing stops and the call is not queued, the call is dropped.

About Call Vectoring commands

Call Vectoring commands perform the following call-related functions:

Providing call treatments

You can provide audible feedback, including silence, ringback, system music, an alternate audio or music source, or a busy tone to the caller. You can administer a recorded announcement to indicate that agents are unavailable to answer the call or to provide some information to the caller. You can also initiate an AUDIX session.

You can delay vector processing for a specific number of seconds before the next vector step is executed.

Routing calls

You can administer queuing of calls to more than one split if an agent fails to answer calls immediately. You can also provide an option to the caller to leave a recorded message. You can administer routing of calls to a number programmed in the vector or to digits collected from the caller.

Branching or programming

You can administer conditional or unconditional branching from one vector step to another step or to another vector. Conditional branching is done according to a number of conditions, for example, the number of available agents in a split, the number of calls in a split queue, and the number of the phone the call is made from. You can stop the vector process when necessary.

Collecting and acting on information

(Optional) You can administer collection of touchtone digits to serve as the basis for further vector processing. For example, the caller can enter certain touchtone digits to reach a specific agent.

Executing VRU scripts

You can execute voice scripts on a VRU for the caller. Voice scripts provide the caller with information or prompts. The caller can then provide a response to a voice script, for example, by entering touchtone digits.

Call Vectoring commands

- **`Adjunct Routing`**: Relays messages to an ASAI adjunct that requests for routing instructions. The command is available only when you enable the CallVisor ASAI capabilities and the **Call Vectoring (Basic)** field.
- **`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 or skill for possible termination of the call to that resource. The command connects to an agent in the split or skill or puts the call in queue

at the specified queuing priority level if the condition specified as part of the command is met. You can administer queuing of a call to up to three different splits or skills simultaneously.

- **`Collect Digits`**: Collects up to 16 digits that are entered by the caller during vector processing, sent by the network, or received from an adjunct. You can play an optional announcement first when the digits are being collected directly from the caller.
- **`Consider location`**: Retrieves the Expected Wait Time (EWT) and relevant agent data to identify the best remote location in multisite Best Service Routing applications. You must write one **`consider`** step for each location that you want to check.
- **`Consider split/skill`**: Retrieves the EWT and relevant agent data to identify the best local split or skill in singlesite Best Service Routing vectors. You must write one **`consider`** step for each split or skill that you want to check.
- **`Converse-on split`**: Integrates VRUs with Communication Manager. You can use the command to execute voice response scripts while a call remains in queue and to pass data between Communication Manager and the VRU.
- **`Disconnect`**: Ends call treatment and removes the call. You can optionally use the command to administer an announcement that plays immediately before the **`disconnect`** command.
- **`Goto step`**: Branches a vector execution conditionally or unconditionally to a preceding or succeeding step in the vector. Conditional branching is determined by a number of factors such as the number of calls that are queued in the split, the number of staffed agents who are in the split, and if the call arrives at a time of day that is in a holiday table.
- **`Goto vector`**: Branches a vector execution conditionally or unconditionally to another vector.
- **`Messaging split`**: Makes provisions for a caller to leave a message for an extension.
- **`Queue-to unconditionally`**: Queues a call to a split or skill and assigns a queuing priority level to the call in case agents are unavailable. When vector processing reaches the command, Communication Manager connects the call to an agent in the split or skill or moves the call to a queue.
- **`Queue-to attd-group`**: Queues a call to the specified attendant group. The command is available only for attendant vectors. When vector processing reaches the command, Communication Manager connects the call to an available agent within the group or moves the call to the attendant group queue if agents are unavailable.
- **`Queue-to attendant`**: Queues a call to a specific attendant. The command is available only for attendant vectors. The call queues to the agent only if the agent is a member of the Tenant Number (TN) associated with the call.
- **`Queue-to hunt group`**: Queues a call to up to three hunt groups. When vector processing reaches the command, Communication Manager connects to an agent in the hunt group or moves the call to the hunt group queue.
- **`Reply best`**: Returns data to another Communication Manager in response to a status poll. You can use the **`Reply best`** command only in status poll vectors for multisite Best Service Routing applications.

- **`Route-to digits`**: Routes the call to the destination that is specified by a set of digits that is collected from the caller or the VRU by the previous **`collect digits`** step.
- **`Route-to number`**: Routes the call to the destination specified by the administered digit string.
- **`Stop`**: Terminates processing of subsequent vector steps.
- **`Wait time`**: Specifies whether the caller hears a 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 syntax.

Condition testing within vector commands

Call Vectoring commands are implemented according to a tested condition that comprises part of the command. In other words, 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 executed and the next vector step is processed.

The following list provides a set of conditions that comprise the conditional portion of a Call Vectoring command:

- The number of staffed agents in a split.
- The number of available agents in a split.
- The number of calls to a split queued at a given priority.
- The time that the oldest call is waiting in a split.
- Whether or not a call receives special holiday processing.
- The Average Speed of Answer (ASA) for a split or a VDN.
- The Expected Wait Time (EWT_ for a split or a call that has entered vector processing.
- A reduction in EWT if a call is queued to a backup resource.
- The number of calls in a queue that are eligible for interflow processing using interflow q-pos.
- The number of active calls that have been routed by a VDN.
- The caller identity, that is, Automatic Number Identification (ANI).
- The type of originating line, that is, Information Indicator (II) digits.
- The caller entered digits (CINFO) that Communication Manager receives from the network or an ASAI or VRU adjunct.
- The time-of-day and the day of the week 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.

The available set of conditions is dependent on the features you administer on the System-Parameters Customer-Options screen.

Based on the condition, specific comparison operators and a threshold can be in effect. Examples of comparison operators are < (less than), > (greater than), = (equal to), <= (less than or equal to), >=

(greater than or equal to), <> (not equal to), and in or not-in. A threshold is a range of accepted numerical entries.

Call Vectoring benefits

Call Vectoring benefits	Examples
Call treatment	
Implement special treatment based on the time of day, the day of the week, and on holidays. For example, routing calls to a different vector when one location is on holiday.	Customer service center on page 32 Distributed call centers on page 37
Automatically change treatment according to either how long the call has been waiting or in response to the changing traffic or staffing conditions.	Automated attendant on page 34 Data in voice answer and data message collection on page 34 Distributed call centers on page 37 Help desk on page 39
Provide multiple or recurring information, or delay announcements that are selected according to the time of day or day of the week, call volume, or staffing conditions.	Customer service center on page 32
Set up and test special call treatments for events such as sales, advertising campaigns, holidays, snow days.	Holiday Vectoring example on page 54
Provide the caller with a menu of choices.	Data in voice answer and data message collection on page 34 Help desk on page 39
Queue calls to up to three splits simultaneously to improve the ASA and agent productivity.	Customer service center on page 32 Distributed call centers on page 37
Implement routing to local or distant destinations.	Customer service center on page 32 Data in voice answer and data message collection on page 34 Distributed call centers on page 37 Help desk on page 39
Connect callers to a voice mail or messaging system either automatically or per caller request.	Data in voice answer and data message collection on page 34
Call routing	

Table continues…

Call Vectoring benefits	Examples
Reduce call transfers by accurately routing callers to the desired destination.	Data in voice answer and data message collection on page 34
Provide up to four ACD queuing priority levels and the ability to change the queuing priority dynamically, thereby, providing faster service for selected callers.	Customer service center on page 32 Data in voice answer and data message collection on page 34 Distributed call centers on page 37
Reduce agent or attendant staffing requirements by: (1) automating some tasks, (2) reducing caller hold time, and (3) having agents in one split provide service multiple call types.	Data in voice answer and data message collection on page 34
Information collection	
Provide customized or personalized call treatment using information collection and messaging.	Automated attendant on page 34 Data in voice answer and data message collection on page 34 Help desk on page 39
Collect information for use by an adjunct or by agent display.	Help desk on page 39
Collect caller-entered or customer database-provided CINFO digits from the network.	CINFO vector example on page 140

Related links

Call Vectoring fundamentals on page 15

Chapter 3: Call Vectoring examples

Example	Feature
Customer service center	Basic Call Vectoring
Automated attendant	Call Prompting
Data in or voice answer and data or message collection	Call Prompting and Basic Call Vectoring
Distributed call centers	Look-Ahead Interflow (LAI) and Basic Call Vectoring
Help desk	Adjunct Routing, Call Prompting, and Basic Call Vectoring
Insurance agency or any service agency	Basic Call Vectoring, Call Prompting, Rolling Average Speed of Answer (ASA), Expected Wait Time (EWT), Vector Directory Number (VDN) calls, and Automatic Number Identification (ANI) routing
Warranty service with EAS	Basic Call Vectoring and Expert Agent Selection (EAS)
Resort reservation	Basic Call Vectoring, Adjunct Routing, Call Prompting, and EAS
Local attendant group access code	Attendant Vectoring
Incoming trunk calls to attendant group	Attendant Vectoring
Incoming Listed Directory Number (LDN) calls	Attendant Vectoring
QSIG Centralized Attendant Service (CAS)	Attendant Vectoring
Night station service	Attendant Vectoring
Holiday Vectoring	Holiday Vectoring
Network Call Redirection (NCR)	Multisite Best Service Routing (BSR) and NCR
BSR using Expected Wait Time (EWT) and Agent Adjustments	Multisite BSR
Dial by Name	Basic Call Vectoring and Call Prompting

Customer service center

In the example, a customer service center is open on weekdays from 8 a.m. to 5 p.m. The center has two separate phone numbers for regular and priority customers. The following vector examples show how calls to the customer service center are handled.

Example application - customer service center

```
VDN (extension=1021  name=Customer Serv  vector=21)
Vector 21:
      1. goto vector 29 @step 1 if time-of-day is all 17:00 to all 08:00
      2. goto vector 29 @step 1 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 seconds hearing ringback
      6. announcement 3521
      7. wait-time 50 seconds hearing music
      8. announcement 3522
      9. goto step 7 if unconditionally
     10. busy

VDN (extension=1022  name=Priority Cust  vector=22)
Vector 22:
      1. goto vector 29 @step 1 if time-of-day is all 17:00 to all 08:00
      2. goto vector 29 @step 1 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 seconds 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 seconds hearing music
     11. goto step 7 if unconditionally
     12. route-to number 0 with cov n if unconditionally

No VDN
Vector 29:
      1. announcement extension 3529
      2. wait-time 10 seconds hearing silence
      3. disconnect after announcement 3529
```

When a *priority customer* places a call, the system accesses Vector 22. The first two steps of Vector 22 determine if the call arrives during non business 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 29. 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 the system accesses Vector 29, the caller hears an announcement twice and the call is then disconnected.

If the call is placed during business hours, step 3 of Vector 22 determines if the number of high-priority calls that are queued in the main split exceeds 10. If more than 10 calls are in queue, control is sent to step 12, which routes the call to an attendant. If less than 10 calls are in queue, the call is queued to the main split as indicated in step 4. An appropriate announcement plays (step 5), followed by a wait period in step 6.

If the call is not answered after the wait time specified in step 6, steps 7 and 8 attempt to queue the call to a backup split (splits 2 and 3, respectively). The call is queued to either split if the oldest call in the split has been waiting fewer than 20 seconds.

Even if the call is queued to one of the backup splits, the call is passed to steps 9 through 11, which implement an announcement-wait cycle that continues until either an agent answers the call, or the caller abandons the call.

When a *non priority customer* places a call, *vector 21* is accessed. Vector 21 provides a treatment similar to that provided by Vector 22, with the following exceptions:

- Backup splits are not queried for non priority calls
- Priority calls are assigned a higher priority in the queue

- Priority calls route to an operator (step 12 of Vector 22) when too many calls are queued, but non priority calls route to a busy signal (step 10 in Vector 21).

Automated attendant

With Automated Attendant, a caller can enter the extension of the agent. The caller can enter up to 16 digits from a touch-tone telephone.

Call centers use Automated Attendant if call centers do not have DID trunks and if the callers know the extension of the agents. Automated Attendant reduces call center costs, as call centers do not require live attendants.

Example application - automated attendant

```
 1. wait-time 0 seconds hearing ringback
 2. collect 5 digits after announcement 30001 [You have reached Ridel Publications.
Please dial a 5-digit extension or
    wait for an attendant.]
 3. route-to digits with coverage y
 4. route-to number 0 with cov n if unconditionally
 5. stop
```

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 ringback in the event that a TouchTone Receiver (TTR) is not immediately available. A TTR must be connected in order for the **`collect digits`** command to take effect. Once a TTR is connected, the caller is prompted to enter the destination extension of the party, that is, 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 enters an invalid extension number, the **`route-to number`** command in step 4 routes the call to an attendant. However, as long as the destination is a valid extension, the **`route-to digits`** command succeeds, coverage applies, and vector processing terminates.

Note:

Even if the destination is busy, vector processing terminates because coverage call processing takes effect.

Data in/voice answer and data/message collection

The scenario 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 the vector uses Call Prompting to determine the desired service. The following three services are offered to the customers:

- New Account: To open new accounts.

- Account Inquiry: To make inquiries about respective accounts.
- Net Asset Value (NAV): To receive information on the NAV of the company funds.

If the caller selects *account inquiries*, the caller is prompted to enter the account number. The agent can use the **callr-info** button to view the account number.

Note:

If the agent has a two-line display telephone, the account number is automatically displayed on the second line. Some supported display telephones include 6416, 6424, 8410, 8434 and CallMaster® set.

The scenario includes the use of the following three applications supported by Call Prompting:

- Data In/Voice Answer (DIVA): The caller can receive information on a topic selected at the prompt. The caller selects the desired topic by entering the appropriate digits.
- Data Collection: Collects the digits entered by a caller. The digits are all numeric and one such example is an account number or US Social Security Number.
- Message Collection: The caller can leave a recorded message instead of waiting for the call to be answered.

The following four vectors illustrate how the mutual fund company handles telephone calls. The vector is programmed to check if queue slots are available.

Example application - mutual fund company

```
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 are sorry. All our operators are
busy at the moment. Dial 1 to leave
         your name and telephone number.]
      6. goto step 10 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 are sorry, we cannot take your message now. Please hold or
call back later.]
     12. goto step 4 if unconditionally
```

DIVA and data/message collection vector example

```
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 7 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. collect 1 digits after announcement 4020 [We are sorry. All our operators are
busy at the moment. Dial 1 to leave
         your name and telephone number.]
      8. goto step 12 if digit = 1
      9. announcement 3537
     10. wait time 50 secs hearing music
     11. goto step 8 if unconditionally
     12. messaging split 5 for extension 4000
     13. announcement 3538 [We are sorry, we cannot take your message now. Please hold or
call back later.]
     14. goto step 4 if unconditionally

 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.]
```

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 that a tone detector is not immediately available. Once a tone detector is connected, the **`collect digits`** command provides an announcement that requests the caller to enter 1, 2, or 3, depending on the service. If the caller enters a digit other than 1, 2, or 3, or if the caller fails to enter any digits within 10 seconds, the command fails and the call is routed to an attendant (step 6). If the caller enters 1, 2, or 3 within 10 seconds, the call is routed to the vector specified in the **`route-to number`** command, which appears in steps 3, 4, and 5.

For instance, when prompted, the caller enters 3 to learn about the NAV of the company funds. The **`route-to number`** command in step 3 and in step 4 fail, because in each case, the digit that is 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 that is tested 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 NAV and disconnects the call.

The process just described, wherein the caller receives information after entering a digit at the prompt, is an example of the DIVA application.

Returning to the main vector, if caller wants to make an inquiry and enters 2 when prompted, 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 the 6-digit account number. The command then collects the digits. Whether or not the caller enters the digits correctly, the **`queue-to split`** command in step 4 queues the call. If an agent does not immediately answer the call, the standard announcement in step 5 plays and a delay is provided in

step 6. The announcement in step 7 provides the caller with the option to either leave a message or hold. The caller is prompted to enter 1 to leave a recorded message. If the caller does not enter 1, the **goto step** command in step 8 fails and an announcement-wait cycle is implemented by steps 9, 10, and 11 until the call is answered or abandoned. If the caller does enter 1 within 10 seconds, step 8 passes control to step 12. The **messaging split** command in step 12 attempts to connect the caller to an AUDIX or a message center split. If the connection is made, the caller hears ringback and can then leave a message. If the connection is not made, the step is unsuccessful and step 13 provides an announcement that indicates that a connection is not made. Thereafter, the **goto step** command in step 14 sends the call control back to step 6, which leads the caller back to the steps to leave a message.

The process just described, wherein the caller enters digits that comprise an official number, 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 does not have to request the caller for an account number.

Finally, if another caller wants to open an account and enters 1 when prompted in the main vector, 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, step 3 provides an appropriate announcement and step 4 provides a delay period. The announcement in step 5 provides the caller with the option to either leave a recorded message or hold. This is an example of the Message Collection application. The caller is prompted to enter 1 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. If the connection is made, the caller hears ringback and can then leave a message. If the connection is not made, the step is unsuccessful and step 11 provides an announcement that indicates that a connection is not made. Thereafter, the **goto step** command in step 12 sends call control back to step 4, which leads the caller back into the steps to leave a message.

Distributed call centers

The example involves two call centers located in New York and Denver. Calls to the New York call center are queued to up to two splits. If calls remain unanswered for a period of time, a Look-Ahead Interflow (LAI) call attempt is made to the Denver call center. If there are less than 10 queued calls in Denver, the LAI call attempt is accepted and serviced at Denver. Else, the call is denied and remains in queue in New York until an agent becomes available. The two vectors below illustrate the process.

Example application - distributed call centers

```
Sending Communication Manager:
VDN (extension=1080   name=New York Office   vector=80)
Vector 80:
       1. goto step 11 if calls-queued in split 1 pri m > 5
       2. queue-to split 1 pri m
       3. announcement 3580 [All our agents are busy at the moment. Please hold.]
       4. wait-time 6 seconds hearing music
```

```
       5. route-to number 913035661081 with cov n if unconditionally
       6. check split 2 pri m if calls-queued < 5
       7. wait-time 6 seconds hearing music
       8. announcement 3581 [All our agents are still busy. Please hold.]
       9. wait-time 60 seconds hearing music
      10. goto step 5 if unconditionally
      11. busy

Receiving Communication Manager:
VDN (extension=1081   name=Denver Inflow   vector=81)
Vector 81:
       1. goto step 7 if calls-queued in split 3 pri l > 10
       2. wait-time 0 seconds hearing music
       3. queue-to split 3 pri h
       4. announcement 3582 [We apologize for the delay. Please hold.]
       5. wait-time 60 seconds hearing music
       6. goto step 5 if unconditionally
       7. disconnect after announcement none
```

Vector 80 is on the sending Communication Manager from a call center in New York, while vector 81 is on the receiving Communication Manager at a call center in Denver.

In the sending Communication Manager, the call is queued to split 1 at a medium priority (step 2) if the condition in step 1 is met. If the condition is not met, the call is routed to busy in step 11.

If the call is queued, but not immediately answered, the caller hears an announcement (step 3) and music (step 4). If the call is still not answered at this point, step 5 places a LAI call attempt to the receiving Communication Manager, on which vector 81 resides.

Step 1 in the receiving Communication Manager determines whether the call can be serviced in Denver. If the number of calls queued at any priority in split 3 is greater than 10, vector 81 does not service the call. In such a case, control is passed to step 7, which rejects the LAI call attempt. However, if the test in step 1 succeeds, the call is queued by the receiving Communication Manager in split 3 at a high priority (step 3) and the LAI call attempt is accepted. Accordingly, the call is removed from the main split queue in New York and the control is passed to the Denver Communication Manager, where vector processing continues at step 4.

If the receiving Communication Manager does not accept the LAI call attempt, control is passed to step 6 of the sending Communication Manager. The step then queues the call to split 2 at a medium priority, provided there are less than five calls queued in that split. Thereafter, the customary announcement-wait sequence plays (steps 7, 8, and 9). Finally, if necessary, step 10 sends the control back to step 5, which makes another LAI attempt and the cycle is repeated.

Note:

To prevent confusion, the treatment provided at the receiving Communication Manager must be consistent with the treatment provided at the sending Communication Manager. In the distributed call centers example, note that the caller hears music and never ringback or silence, at the sending Communication Manager. Accordingly, music must be featured at the receiving Communication Manager.

Help desk

The scenario involves a help desk at a computer firm. The help desk is configured into three groups. One group handles hardware problems, the second group handles software problems, and the third group handles general problems. The information provided in the Adjunct Switch Application Interface (ASAI) route request, that is, calling party number, called number, and collected digits, is used to route the call to an appropriate agent. Such an agent can be the one who last serviced the caller, or the next available agent for the specific caller. Also, based on the traffic conditions of Communication Manager and the caller entered digit, the call can be diverted to other destinations such as other ACD splits, announcements, or to another Communication Manager.

Example application - help desk

```
1. wait-time 0 seconds hearing ringback
2. collect 1 digits after announcement 4704 [Welcome to the TidyBits Computer Corporation
help desk. For hardware problems,
   dial 1. For software problems, dial 2. For general queries, dial 3.]
3. adjunct routing link 12
4. wait-time 4 seconds hearing ringback
5. route-to number 3710 with cov y if digit = 1
6. route-to number 3720 with cov y if digit = 2
7. route-to number 3730 with cov y if digit = 3
8. route-to number 0 with cov n if unconditionally
9. stop
```

Step 1 of the vector contains the **`wait-time`** command to provide the caller with ringback in the event that a TTR is not immediately available. A TTR must be connected for the **`collect digits`** command to take effect. In step 2 of the vector, the caller is prompted to enter 1, 2, or 3, based on the type service required by the caller. Thereafter, the **`adjunct routing link`** command in step 3 instructs Communication Manager to send a route request to the adjunct processor. The route request contains the called party number, the calling party number, and the digit that is collected in step 2, along with the other pertinent information for adjunct routing. If 1, 2, or 3 is not entered and if the adjunct does not return a route, the call is eventually routed to an attendant (step 8).

If the **`adjunct routing link`** command in step 3 succeeds, the adjunct uses the information included in the route request to select the appropriate route for the call. If the caller enters 1 and the **`adjunct routing link`** command succeeds, the call is routed to agent who handles hardware related queries. For general hardware problems, the call can be routed to a larger group of ACD agents before the call is queued.

Insurance agency

In the example, an insurance company call center handles calls from independent field agents, policy holders with claims, policy holders with need for customer service, and several general service agency type toll–free number client accounts such as 800, 888, 877, 866 number types. Each different type of call has an 800 number that routes the calls to associated VDNs.

Call center requirements:

- Independent field agents require prompt service. The field agents call the company to find the latest rates for specific clients, to set up policies, or to make adjustments. Therefore the

insurance company wants to maintain an Average Speed of Answer (Rolling-ASA) of less than 30 seconds for field agent calls. Field agent calls are given the highest priority in queue.

- Calls to Claims must be separated by area code. Training for the Claims agents is based on the area of the country for the claim. A particular group of agents can be given training for more than one area code. Therefore, area codes must not be tested individually and can be grouped in Vector Routing Tables (VRT).
- The insurance company wants the customer service callers to hear an announcement indicating the wait duration for a service.
- The insurance company also sells spare call center capacity to client accounts. The account contracts are provided on the basis that only so many calls to a particular account are accepted at any given time.

In this example, Rolling ASA routing is used to maintain the rolling ASA objective of less than 30 seconds for field agent calls. ANI or CLID routing is used to partition arriving calls based on any of the following combinations so as to route calls to the appropriate Claims agent:

- Area code, that is, Numbering Plan Area (NPA)
- Area code and Central Office (CO) within the NPA, that is, NPA-NXX
- Entire phone number, that is, NPA-NXX-XXXX

The terms ANI and CLID are, in practice, used interchangeably with respect to vectors. ANI equates to the caller's number when the caller dials a toll-free number, whereas CLID equates to the caller's number when the caller dials a non toll-free number. EWT routing is used to notify callers about the expected wait time if the wait time is more than 60 seconds. VDN Calls routing is used to regulate the number of calls to service agency clients.

The following table shows the VDNs and vectors associated with each type of call.

VDN table for insurance agency or service agency example		
Type of service	**VDN number**	**Vector number**
Field Agents	1001	1
Claims	1002	2
Customer Service	1003	3
Client 1	1004	4
Client 2	1005	5

Note:

To demonstrate the features described in the example, the sample vectors do not include tests for unstaffed or full queues and out-of-hours operation.

Step 1 queues the call to the main split. If the main split is currently answering calls within the target time of 30 seconds, step 2 bypasses all the backup splits and goes to the announcement in step 6. The call is handled by split 10 within the time constraints. However, if the call is not answered by the time that vector processing reaches step 8, the backup splits are checked.

If the Rolling ASA for the main split is more than 30 seconds, steps 3, 4, and 5 check backup splits are executed. The call is queued to any of the splits that have a Rolling ASA of less than 30

seconds. If the call still is not answered by the time vector processing reaches step 8, the backup splits are checked again.

Use the following vector example to route Claims calls by area code.

Claims vector example

```
 VDN 1002 -- Claims calls
 1. goto step 10 if ani = none
 2. goto vector 21 @step 1 if ani = 201+
 3. goto vector 22 @step 1 if ani = 212+
 4. goto vector 23 @step 1 if ani in table 1
 5. goto vector 24 @step 1 if ani in table 2
 6. goto vector 25 @step 1 if ani in table 3
 7. goto vector 26 @step 1 if ani in table 4
 8. goto vector 27 @step 1 if ani in table 5
 9. goto vector 30 @step 1 if unconditionally
10. wait-time 0 seconds hearing ringback
11. collect 3 digits after announcement 10001   [Please dial your area code]
12. goto vector 30 @step 1 if digits = none
13. goto vector 21 @step 1 if digits = 201+
14. goto vector 22 @step 1 if digits = 212+
15. goto vector 23 @step 1 if digits in table 1
16. goto vector 24 @step 1 if digits in table 2
17. goto vector 25 @step 1 if digits in table 3
18. goto vector 26 @step 1 if digits in table 4
19. goto vector 27 @step 1 if digits in table 5
20. goto vector 30 @step 1 if unconditionally
```

Each VRT referenced in the example contains a list of area codes with the “+” wildcard. Each list of area codes is handled by a specific group of agents. Vectors 21 through 27 queue calls to the appropriate group of agents. Vector 30 provides a live agent to screen calls that have area codes that are not listed in any table or vector step. The vector also provides access to an agent when ANI is not available and the caller does not enter an area code when prompted.

The following vector example notifies customer service callers of the expected wait time.

Customer Service vector example

```
 VDN 1003 -- Customer Service calls
 1. goto step 10 if expected-wait for split 32 pri l > 600
 2. queue-to split 32 pri l
 3. wait-time 20 seconds hearing ringback
 4. goto step 8 if expected-wait for call > 40
 5. announcement 1100
 6. wait-time 40 seconds hearing music
 7. goto step 5 if unconditionally
 8. converse-on split 80 pri l passing wait and none
 9. goto step 5 if unconditionally
10. disconnect after announcement 1400
```

In step 1, callers who wait more than 10 minutes are routed to a call back later announcement. Step 4 routes callers to a VRU to be given the EWT announcement while the callers hold the place in the queue.

The following vector examples can be used to regulate the number of calls to service agency clients. In this example, client 1 has contracted for 100 simultaneous calls while client 2 has contracted for only 50 simultaneous calls.

Service Agency Clients vector examples

```
VDN 1004-- Client 1 calls
1. goto step 3 if counted-calls to vdn 1004 <= 100
2. busy
```

```
3. queue-to split 60 pri l
4. wait-time 20 seconds hearing ringback
5. announcement 12000
6. wait-time 60 seconds hearing music
7. goto step 5 unconditionally

VDN 1005 -- Client 2 calls
1. goto step 3 if counted-calls to vdn 1005 <= 50
2. busy
3. queue-to split 60 pri l
4. wait-time 20 seconds hearing ringback
5. announcement 12000
6. wait-time 60 seconds hearing music
7. goto step 5 unconditionally
```

In both the examples vectors, the first step routes calls to queue if the number of contracted calls is not exceeded. Otherwise callers receive a busy signal.

Warranty service with EAS

The scenario involves a major appliance company that offers one year warranties and extended warranties on major appliances such as dishwashers, refrigerators, washers, and dryers. The warranties are printed in English and Spanish to serve customers who speak both languages. 800 numbers are provided for calling both English-speaking agents and Spanish-speaking agents. Bilingual agents with Spanish-speaking skills are hired to back up the groups of English-speaking agents. Agents are first trained on all appliance models of a certain type and then on all appliance models for a room such as the kitchen, and the laundry room.

The skills shown in the following table are required for the warranty service call center.

Skill for a warranty service call center		
Appliance type	**English skill number**	**Spanish skill number**
Kitchen appliances	10	20
Dishwashers	11	21
Refrigerators	12	22
Laundry appliances	30	40
Washers	31	41
Dryers	32	42
Supervisors	100	

The VDN skill preferences are set up as shown in the following table.

VDN skill for the warranty service call center					
VDN skill preference	**Appliance**	**VDN**	**First**	**Second**	**Third**
English	Dishwasher	1100	11	10	20
	Refrigerator	1101	12	10	20
	Washer	1102	31	30	40
	Dryer	1103	32	30	40
Spanish	Dishwasher	1200	21	20	--
	Refrigerator	1201	22	20	--
	Washer	1203	41	40	--
	Dryer	1204	42	40	--

The agent skills are set up as shown in the following table.

Agent skills for the warranty service call center				
Agent	**Skill level 1**		**Skill level 2**	
Kim	42	40	41	30
Michelle	100	--	--	--
Beth	31	--	--	--
Mike	32	--	30	--

Once skills are assigned to VDNs and to agents, calls are directed to the appropriate vector.

The goal of the warranty service call center is to answer 80 percent of the incoming calls within 20 seconds. Accordingly, if a call directed to a vector is not answered by the time the announcement finishes, a second group of agents is viewed, enlarging the agent pool. If the call is not answered within the following 10 seconds, a third group of agents is viewed.

Since the call center has only a few bilingual agents, the management wants to reserve the agents for Spanish-speaking callers only. This can be done by giving Spanish-speaking callers a higher priority in the vector or by assigning a higher skill level to Spanish skills. Also, if a Spanish-speaking caller waits more than 30 seconds for service, a supervisor of the Spanish-speaking skills takes the calls.

The figures depict the setup for the warranty service call service. Specifically, the figures show the vectors and call flows for callers with a broken washer or dryer who need service. Separate vectors are used to provide an announcement in Spanish and in English, see step 2. The same two vectors can be used for callers who need service for broken dishwashers and refrigerators.

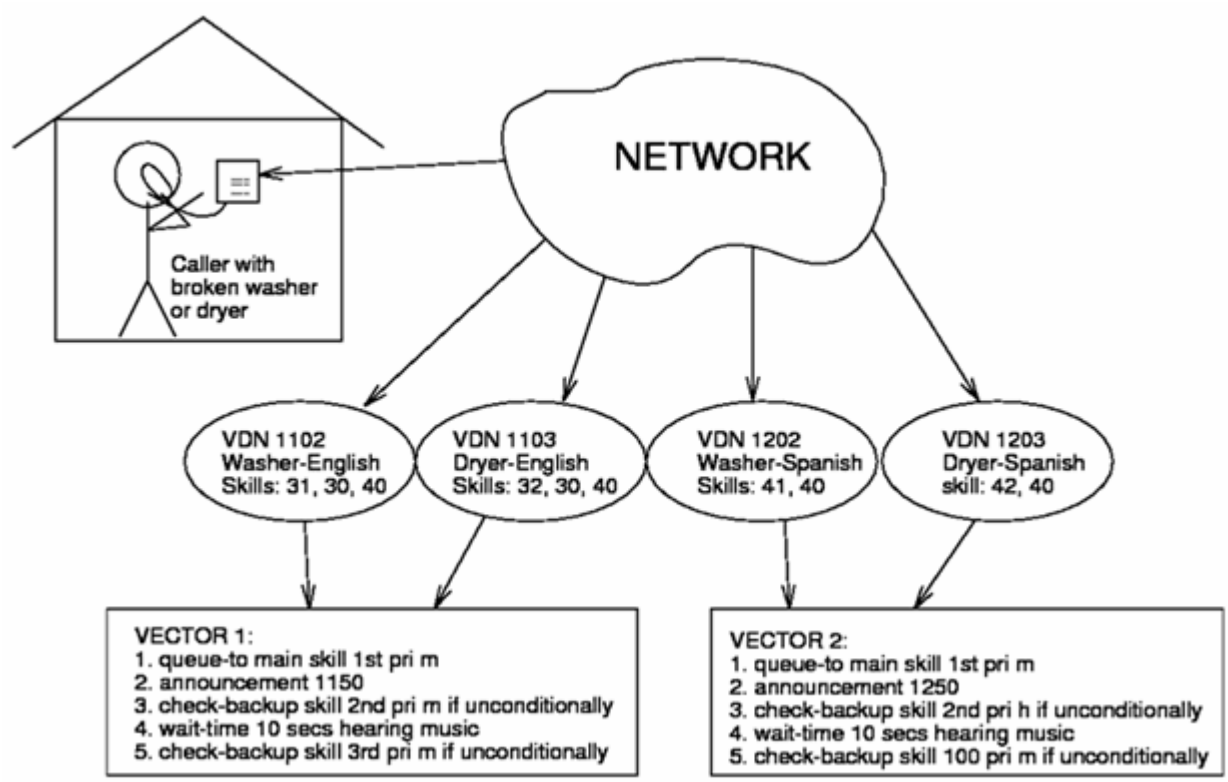


Figure 1: Warranty service call center (part 1)

The next figure depicts how the vector-processed call is directed to the appropriate call queue. The figure also shows how the call is directed to the appropriate agent or agents. The agent skills are indicated below the name of each agent. Dashed lines indicate backup or secondary skills.

Note:

Only a small sample of agents is shown in the example figure.

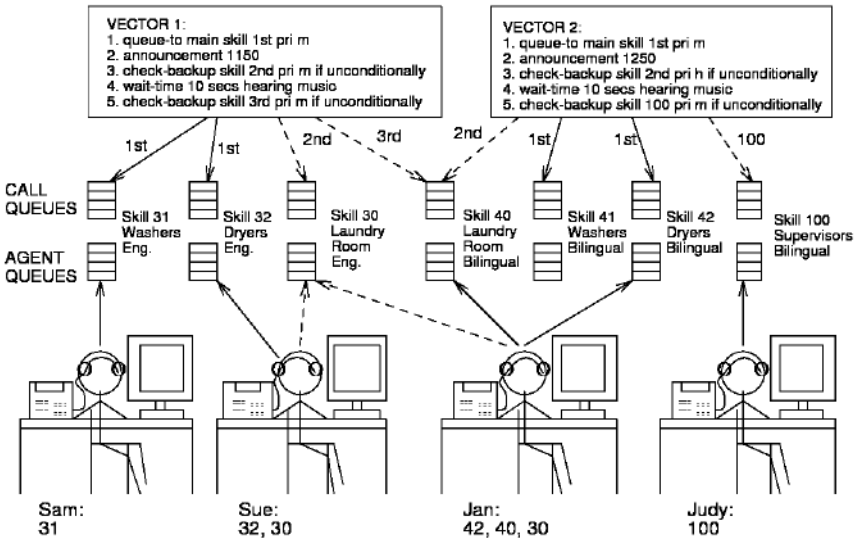


Figure 2: Warranty service call center (part 2)

A Spanish-speaking customer has a broken dryer. The caller dials the call center number. Communication Manager receives and directs the call to VDN 1203, which points to vector 2. The system administrator assigns the VDN skill preferences 42 for dryers and 40 for laundry appliances as the first and second skill preferences respectively.

Once vector processing starts, the **`queue-to skill`** command in step 1 of vector 2 queues the call to the skill group corresponding to the first VDN skill (42-Dryers Bilingual). If an agent with skill 42 is available, the agent answers the call. If the agent is not available, an appropriate delay announcement in step 2 plays. Next, the **`check skill`** command in step 3 attempts to queue the call to the skill group corresponding to the 2nd VDN skill (40-Laundry Appliances Bilingual). If an agent with skill 40 is available, the agent answers the call. Otherwise, a wait period is provided in step 4 and the **`check skill`** command in step 5 checks the specific skill (100-Supervisors Bilingual) for available agents.

Notify callers about queue position

XYZ call center has the following requirements:

- Announce the position of a call in queue to callers.
- Do not use a wait time estimate because the call traffic for this call center is random and the talk times are variable.
- Do not use the existing Integrated Voice Response (IVR) or Voice Response Unit (VRU) equipment.

Scenario solution

The XYZ call center decides to use the interflow-qpos goto step conditional to test the caller position in queue. The interflow-qpos goto conditional checks the caller's position in queue from 1 (next in line) to 9 (8 calls are ahead).

Scenario prerequisites

- Virtual Routing (LAI) must be active.
- Set the **Interflow-Qpos EWT Threshold** field on the Feature-Related System Parameter screen to `0` seconds. The interflow-qpos tests what is defined as the eligible queue. Setting the field to `0` does not exclude calls at the top of the queue.

How to set up the interflow-qpos conditional

- Use the same priority for queuing all calls.
- To add a test such as for working hours in the beginning, use a vector which ends with **`goto vector x unconditionally`**, where vector x has the loop with the announcing steps.
- If you have queue limiting, use a "goto step/vector y if calls-queued in skill 25 pri l > queue_limit" step ahead of step 2 to give the caller an alternate treatment if the call cannot be queued.

```
1. wait-time 0 secs hearing ringback
2. queue-to skill 25 pri l
3. announcement 1000  [All our specialists are busy handling other customers. Your call
is important to us, so please wait.]
4. goto step 6 if interflow-qpos <> 1
5. announcement 1001 [you are the next call to be answered]
6. goto step 8 if interflow-qpos <> 2
7. announcement 1002 [you have 1 call ahead of you]
8. goto step 10 if interflow-qpos <> 3
9. announcement 1003 [you have 2 calls ahead of you]
10. goto step 12 if interflow-qpos <> 4
11. announcement 1004 [you have 3 calls ahead of you]
12. goto step 14 if interflow-qpos <> 5
13. announcement 1005 [you have 4 calls ahead of you]
14. goto step 16 if interflow-qpos <> 6
15. announcement 1006 [you have 5 calls ahead of you]
16. goto step 18 if interflow-qpos <> 7
17. announcement 1007 [you have 6 calls ahead of you]
18. goto step 20 if interflow-qpos <> 8
19. announcement 1008 [you have 7 calls ahead of you]
20. goto step 22 if interflow-qpos <> 9
21. announcement 1008 [you have 8 calls ahead of you]
22. goto step 26 if interflow-qpos <= 9
23. announcement 1009 [There are more than 8 calls ahead of you]
24. collect 1 digits after announcement 3000 [If you would like to leave a callback
message dial 1 otherwise press # or
    continue to wait {10 sec timeout is the default but it is adjustable}]
25. goto vector 200 if digits = 1 [vector 200 provides callback messaging via the
messaging command and related treatment]
26. wait-time 60 secs hearing music {this is optional}
27. announcement 1000 [Our specialists are still busy, please continue to wait]
28. goto step 4 unconditionally
```

Resort reservation service

The example involves a resort company that places a variety of advertisements in magazines for information on a particular resort or state. A call center makes the reservations for the resort company. To satisfy the request of many callers, an effort is made to connect callers to an agent who has visited the resort that the callers are interested in visiting. The resort company has also decided to sell additional sightseeing packages if the agent has a regional accent.

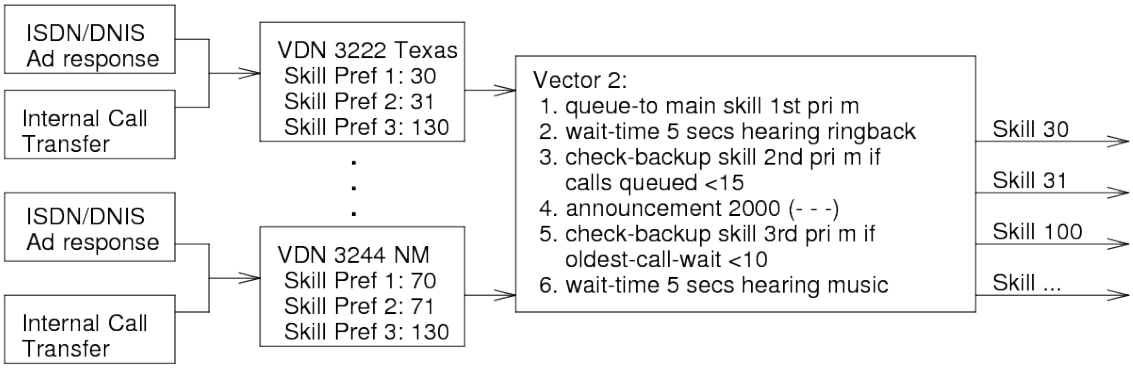
To respond to an advertisement, the caller can dial a number that directly routes the caller to a VDN for that state's resorts. As an alternative, the caller can dial the general number for the resort chain and be serviced using Call Prompting.

Specific number dialing

The call center is set up in such a way that a VDN with an accompanying set of VDN skill preferences is assigned to each state that has a resort. For example, the following table shows how skill preferences are assigned to the Texas VDN 3222.

Texas VDN 3222 skill preferences		
Skill preference	**Skill number**	**Agent skill**
1st	30	Agent who has a Texas accent and has visited resorts in Texas
2nd	31	Agent who has visited resorts in Texas
3rd	130	Any agent who can make the reservation

The following figure depicts how Call Vectoring processes a call to the VDN 3222.



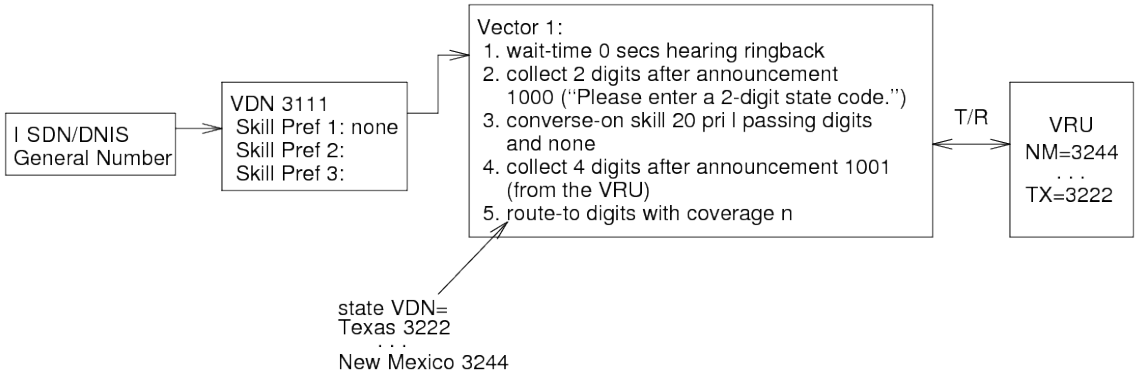
A single VDN for each state is assigned to vector 2. If a caller wants information on resorts in Texas and dials the correct number, for example, 615-3222, the call reaches Communication Manager and is directed to VDN 3222 which points to vector 2.

Once vector processing starts, the **`queue-to skill`** command in step 1 queues the call to the skill group that corresponds to the 1st VDN skill, 30-agent with a Texas accent who has visited resorts in Texas. If an agent with skill 30 is available, the agent answers the call. If the agent is not available,

the **`check skill`** command in step 3 attempts to queue the call according to the stated conditions, if calls-queued < 15, to the skill group that corresponds to the 2nd VDN skill, 31-agent who has visited resorts in Texas. If step 3 fails, the **`check skill`** command in step 5 attempts to queue the call based on the stated conditions, that is, if the oldest-call waiting < 10, to the skill group that corresponds to the 3rd VDN skill, 100-any agent who can take a reservation.

General number dialing

Callers can dial a general number, for example, 615-3111. The caller is provided service in part using Call Prompting. The following figure depicts how a call to VDN 3111 is processed using Call Vectoring.



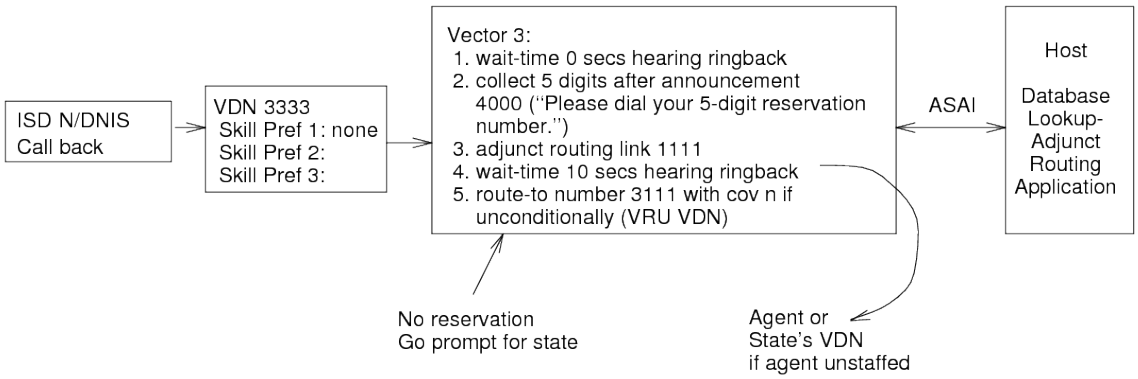
After the number is dialed, the call is directed to VDN 3111, which points to vector 1. Note that there are no skill preferences assigned to VDN 3111, which is the only VDN that is administered to point to vector 1. Therefore, VDN 3111 is used for calls from all states.

The **`collect digits`** command in step 2 of the previous vector first requests the caller to enter the appropriate 2-digit state code and then collects the digits. If the caller enters the correct code for Texas, which is 05, the **`converse-on skill`** command in step 3 delivers the call to the converse skill if there is a queue for the skill and the queue is not full, or if a VRU port is available.

When the VRU port responds, the step outpulses the state code 05 to the VRU using the passing digits parameter that is included in the command. Once the VRU receives this state code, the VRU in turn outpulses the Texas VDN (3222) to Communication Manager. Thereafter, the **`collect digits`** command in step 4 collects the digits that comprise this VDN. Finally, the **`route-to digits`** command in step 5 routes the call to Texas VDN 3222, which points to vector 2.

Callback provisions

After a caller makes a reservation for a resort site, the caller is given a callback number. Such a number is helpful if the caller requires more information or wants to check on some arrangement that was previously made. The following figure depicts one approach for activating callback provisions.



After the number is dialed, the call is directed to VDN 3333, which points to vector 3. Note that there are no skill preferences assigned to VDN 3333. Also, VDN 3333 is the only VDN that is administered to point to vector 3. Therefore, the VDN is used for calls from all states.

The **`collect digits`** command in step 2 of the previous vector first prompts the caller to enter the 5-digit reservation number and then collects the digits. Once the digits are collected, the **`adjunct routing link`** command, if successful, in step 3 causes Communication Manager to send the collected digits, along with other information, to the host in the ASAI adjunct routing request. The host then uses the digits to perform a database lookup for the agent who made the reservation and the resort that corresponds to the reservation. If the agent is currently logged in, the call is automatically routed to the agent. Information on the relevant reservation is displayed at the agent data terminal. If the agent is not logged in, the call is routed to step 5, where the **`route to`** command unconditionally routes the call to the VRU VDN 3111.

Attendant routing example

The following example shows how Attendant Vectoring can be used to route calls in an attendant environment.

Note:

For the following vector examples, set **Tenant Partitioning** to `y`.

VDN 1999 **Vector 1**	**VDN 2999** **Vector 2**	**VDN 3999** **Vector 3**
1. wait-time 0 secs hearing ringback 2. goto step 7 if time-of-day is all 12:00 to 13:00 3. queue-to attd-group 4. goto step 8 if queue-fail	1. wait-time 0 secs hearing ringback 2. queue-to attd-group 3. goto step 7 if queue-fail 4. announcement 9000	1. wait-time 0 secs hearing ringback 2. goto step 8 if time-of-day is all 12:00 to 13:00 3. queue-to attd-group 4. goto step 11 if queue-fail

VDN 1999 Vector 1	VDN 2999 Vector 2	VDN 3999 Vector 3
5. wait 999 secs hearing music 6. busy 7. route-to number 4000 with cov y if unconditionally 8. route-to number 93035381000 with cov y if unconditionally	5. wait 999 seconds hearing music 6. disconnect after announcement 9001 7. queue-to hunt-group 1 8. goto step 11 if queue-fail 9.wait 999 secs hearing ringback 10. busy 11. route-to number 93035381000 with cov y if unconditionally	5. announcement 9000 6. wait 15 seconds hearing music 7. goto step 4 if unconditionally 8. queue-to attendant 6000 9. goto step 11 if queue-fail 10. wait 999 secs hearing ringback 11. route-to number 93035381000 with cov y if unconditionally

Vector administration

- All stations are assigned TN 1 which is associated with attendant group 1, VDN 1999, and music source 1.
- All trunk groups are assigned TN 2 which is associated with attendant group 1, VDN 2999, and music source 2.
- All LDNs are assigned TN 3 which is associated with attendant group 2, VDN 3999, and music source 3.
- Extension 4000 is assigned to a hunt group 1.
- Extension 6000 is assigned to an attendant console for direct access.

Note:

Covered calls can be rerouted to an attendant group for a different tenant partition B when the call does not queue to the attendant group for tenant number A or for the out of hours case, and so on. The vector assigned to the attendant vectoring VDN for tenant A needs to include a failure branch to a “route-to ldn_number with cov y if unconditionally” step to route to the LDN extension of tenant B. The “with coverage” parameter of the route-to step must be set to “cov y” and the original coverage path that covers to the tenant A attendant vectoring VDN must have the “Cvg Enabled for VDN Route-to party?” field set to “y”.

Local attendant group access code

When a station dials the attendant access code, the call is redirected to vector 1. If it is lunch time, the call is sent to a hunt group and vector processing terminates. If it is not lunch time, the call is sent to attendant group 1. If an attendant is available, the call is directed to the attendant and vector processing terminates. Otherwise, the call is queued to the attendant group and the caller hears music from a music source that is assigned to TN 1 until an attendant answers the call. If the call cannot be queued, it is routed to a remote location with coverage, and vector processing terminates.

If the call is unanswered after 999 seconds in the attendant queue, the caller hears a busy signal and vector processing terminates.

Note:

The **`route-to`** command leaves vector processing as soon as the call is successfully routed. So, if it is lunch time the call routes to the hunt group and all hunt group processing applies. If the group is assigned a queue, the call is queued. If the group is not assigned a queue and the coverage criteria is met, the call follows the hunt group's coverage path. If the hunt group is in night service, the call goes to the hunt group's night service destination. If the **`route-to`** command indicates coverage `n`, the hunt group's coverage path is not followed and vector step 7 applies.

Incoming trunk calls to attendant group

When a trunk that has the attendant group assigned as the incoming destination receives a call or when the attendant group addresses the call, the call is redirected to vector 2. The call is then sent to attendant group 1. If an attendant is available, the call is terminated to the attendant and vector processing terminates. Otherwise, the call is queued to the attendant group and the caller hears the announcement followed by music from the music source that is assigned to TN 2. If the call is unanswered after 999 seconds in the attendant queue, the caller is dropped after hearing an announcement and vector processing terminates. If queuing to the attendant fails, the call is queued to hunt group 1. If a member is available to take the call, the call is terminated to the member and vector processing terminates. If a member is not available and the call can be queued, the call is queued and the caller hears ringback until a member answers. If the call is unanswered after 999 seconds in the hunt group queue, the caller hears busy and vector processing terminates. If the call is not queued, the call is routed to the remote location and vector processing terminates.

Incoming LDN calls

When a call is received for an Listed Directory Number (LDN), the call is redirected to vector 3. If it is lunch time, the call is sent to attendant 6000. If the attendant is available, the call is answered and vector processing terminates. If the attendant is not available, the call is placed in queue and the caller hears ringback until the attendant answers the call. If the call is unanswered after 999 seconds in the attendant's queue, the call is sent to the remote location and vector processing terminates. If the call cannot be placed in the queue of attendant 6000, the call is routed to a remote location and vector processing terminates. If it is not lunch time, the call is sent to attendant group 2. If an attendant is available, the call is directed to the attendant and vector processing terminates. Otherwise, the call is queued to the attendant group and the caller hears an announcement followed by music from the music source assigned to Tenant Number (TN) 3 every 15 seconds. If the call cannot be queued, the call is sent to attendant 6000.

Note:

Vector 3 attempts to queue the call to attendant 6000. A **`route-to`** command can also be used, but ensure that an attendant is not assigned a coverage path. To change to a different tenant attendant group see note under Vector administration.

QSIG Centralized Attendant Service

This example shows how you can use Attendant Vectoring with Centralized Attendant Service (CAS).

CAS branch

If a call center wants to always play an announcement at a QSIG CAS branch before routing the call to the QSIG CAS main, an attendant VDN must be administered in the **QSIG CAS Number** field at the branch instead of the number to the QSIG CAS main attendant access code, which is 303-538-0 with an Automatic Alternate Routing (AAR) access code of 9, in this example. The following vector plays an announcement and then routes the call to the QSIG CAS main.

Administration for vector 1 of the attendant VDN is shown in the following Call Vector example.

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

    Number: 1        Name: Night station service vector 4
Multimedia? n     Attendant Vectoring? y      Lock? y
     Basic? n  EAS? n  G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
 Prompting? y  LAI? n G3V4 Adv Route? n  CINFO? n  BSR? n   Holidays? n

01 announcement 9000
02 route-to number     93035380       with cov y if unconditionally
03
04
05
06
07
08
09
10
11
```

CAS main

Calls from a QSIG branch are sent to the QSIG CAS main with the main attendant access code as the destination address. Therefore, these calls automatically become attendant group calls. The VDN to which these calls are redirected depends on the Tenant Number (TN) of the incoming trunk.

Night station service

This example shows how to use Attendant Vectoring for night service.

Night station service vectors 4 and 5

```
change vector 4                                            Page 1 of 3
      CALL VECTOR

    Number: 4        Name: Night station service vector 4
Multimedia? n     Attendant Vectoring? y      Lock? y
     Basic? n  EAS? n  G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
 Prompting? y  LAI? n G3V4 Adv Route? n  CINFO? n  BSR? n   Holidays? n

01 route-to     number 9303538100       with cov y if unconditionally
02
03
04
05
06
07
08
09
10
11
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

change vector 5                                            Page 1 of 3
      CALL VECTOR

    Number: 5        Name: Night station service vector 4
Multimedia? n     Attendant Vectoring? y      Lock? y
     Basic? n  EAS? n  G3V4 Enhanced? n  ANI/II-Digits? n  ASAI Routing? n
 Prompting? y  LAI? n G3V4 Adv Route? n  CINFO? n  BSR? n   Holidays? n

01 route-to     number 6000        with cov n if unconditionally
02 route-to     number 93035381000 with cov y if unconditionally
03
04
05
06
07
08
09
10
11
```

Administration for vector 4 and vector 5 of VDN 4999 is as follows.

- Trunk group 1 is assigned TN 2 which is associated with attendant group 1, and night destination 4999.
- Trunk group 2 is assigned TN 1 which is associated with attendant group 2, and night destination 5999.
- Extension 6000 is assigned to a station.
- System night service is on.

When a non-DID call comes in on trunk group 1, the call is redirected to VDN 4999 which routes it to a remote location.

When a non-DID call comes in on trunk group 2, the call is redirected to VDN 5999 which routes it to station 6000. If station 6000 is unavailable, the call does not cover on station 6000's coverage path. Vector processing continues and routes the call to a remote location.

Note:

When station night service is active, calls are processed according to the administered night destination for the trunk group, not the night destination for the associated TN. In other words, these are not attendant group calls. If the night destination is assigned as `attd` or left unassigned, the calls become attendant group calls and are processed according to the partitions night destination.

Holiday Vectoring example

This is an example of a vector that directs calls to special processing due to a holiday or special event. Holiday Vectoring is an enhancement that simplifies vector writing for holidays.

In this example, a commercial bank is headquartered in Germany and has branches across Europe. The bank recently established a U.S. presence with branches in the New York City metropolitan area. The credit card division operates two 100-agent call centers in Ireland and Germany and one 50-agent call center in the U.S.

All agents in the call centers are bilingual and are assigned to splits that handle calls from both English and German customers. The New York call center is open 24 hours a day, often receiving calls routed from the Irish and German call centers, after the centers close at 6:00 p.m. local time.

Due to a large number of bank holidays in Europe, up to 30 days a year, Holiday Vectoring can be used to create vectors that distribute calls automatically on holidays. Holiday Vectoring saves cost and the time spent on writing vectors for date-related processing. This is turn saves business that is lost due to abandoned calls if vectors are not re-administered for holidays.

The following example indicates that, beginning on December 24 and continuing through 6:00 am on January 2, incoming calls to the call center in Germany will be processed as Christmas holiday calls.

Note:

Because date ranges must be within the same calendar year, New Year's day must be entered as a separate item.

Setting up a holiday table

```
change holiday-table 1                                        page 1 of 1
                                HOLIDAY TABLE

    Number: 1             Name: Bank Holidays

          START                   END
    Month Day Hour Min    Month Day Hour Min     Description
    12    24  00   00     12    31  23   59      Christmas
    01    01  00   00     01    02  06   00      New Year's Day
```

After setting up the Holiday Tables screen modify the vector processing for the holidays. On the Call Vector screen, enter the new goto conditional for the holidays.

When you set the **Holiday Vectoring** field to `y`, a field on the Call Vector screen identifies if the vector on which you are currently working is a holiday vectoring vector, as shown in the following example.

Modifying a vector to route according to a holiday table

```
change vector x                                                  page 1 of 3

                                    CALL VECTOR


    Number: xxx      Name: ____________________________
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing? y
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? y   Holidays? y
 Variables? y                3.0 Enhanced? y

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

The **Vectoring (Holiday)** field is a display-only field and appears only when you set the **Holiday Vectoring** field on the Customer Options screen to `y`. If either **Vectoring (Basic)** or **Attendant Vectoring** are set to `y`, the **Vectoring (Holiday)** field can be set to `y`.

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

    Number: 1        Name: In Germany
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     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   Holidays? y

01 goto         vector 2  if holiday         in    table 1
02 route-to     number 123456789             with cov n if unconditionally
03
04
05
06
07
08
09
10
11
```

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

    Number: 3        Name: In Ireland
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     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   Holidays? y
```

```
01 goto         step   2  if holiday         in    table 2
02 route-to     number 45678          with cov n if unconditionally
03 stop
04 announcement 2721
05
06
07
08
09
10
11
```

The setup for the vector routes the call to a call center in the U.S. For example, if someone in Europe calls the bank before 6:00 a.m. on January 2, the call is routed to the U.S. call center. If someone in Europe calls after 6:00 a.m. on January 2, the call is routed to the German call center.

After you have assigned holiday tables to several vectors, you can use the **`list usage holiday-table`** command to view which vectors and vector steps are using the selected holiday table.

```
list usage holiday-table x
                              LIST USAGE REPORT

     Used By
     Vector                 Vector Number 1                       Step 1
     Vector                 Vector Number 3                       Step 1
```

NCR example

The example shows the primary, status poll, and interflow vectors that redirect calls on the public network using Network Call Redirection (NCR).

An e-Commerce company has three call centers. In an effort to reduce costs, the company implemented NCR to redirect calls on a public network and to reduce the trunking costs between the three call center offices. Best Service Routing (BSR) is implemented to increase the agent utilization.

The company receives calls from a public network. Trunks used to deliver calls from the public network have been assigned Network Call Transfer (NCT) capabilities. NCT occurs after the incoming call is initially answered. With NCT, Communication Manager is required to set up the second leg of the call and then to wait for the second site to acknowledge before requesting the public network to transfer the first leg of the call to the second leg, and before the public network drops the trunks to Communication Manager. The benefit is that Communication Manager retains control over the call and can redirect the call using the trunk-to-trunk method if the NCT invocation fails.

After the second leg of the call is initiated and acknowledged by the public switch, the public network joins the original caller to the redirected-to endpoint and drops both the original call and the second leg of the call at the redirecting Communication Manager.

To activate the NCR feature for each site, the Communication Manager administrator ensures that the **Net Redir** field on the BSR Application Table screen is set to `y` for the location entry.

The company has set up IP trunking to emulate ISDN PRI and uses this capability to poll remote sites for possible NCR.

The following sections provides examples on how to set vectors at each site, to use the public network with NCR, as opposed to IP trunking, to route a call from one site to another.

Primary vector for NCR example

A call arrives at the e-Commerce location 1 and a primary vector processes the call. The vector begins the BSR process by checking the specified resources. The following example shows the primary vector for incoming call processing at the e-Commerce location 1.

Primary vector

```
1. wait-time 0 secs hearing ringback
2. consider split 1 pri m  adjust-by 0
3. consider location 2     adjust-by 30
4. consider location 3     adjust by 10
5. queue-to-best
```

If location 2 returns the lowest EWT, the call is routed to location 2.

Status poll vector for NCR example

To collect information from a remote Communication Manager, the command **`consider`** *location 2* adjust-by *30* in the primary vector places a status poll using IP trunking to the status poll vector on Communication Manager at location 2.

Status poll vector

```
1. consider split 2   pri m   adjust-by 0
2. consider split 11  pri m   adjust-by 0
3. reply-best
```

The status poll returns the information to the origin Communication Manager. The call is not connected to the status poll VDN. Once the remote Communication Manager has returned the necessary information, the **`consider`** series in the primary vector at location 1 proceeds to the next vector step.

Interflow vector for NCR example

Communication Manager selects the site for call routing, and sends the call to the public network. The public network switch sets up a second leg of the call and passes the codeset 0 UUI information in the SETUP message, if this is supported. If you set the **Net Redir** field on the BSR Application screen to `y` for location 1, location 2, and location 3, Communication Manager sends a request to the public switch for call transfer over the public network.

For incoming 800 number calls from MCI DMS-250 network switches, you must ensure that the vector reached by the second leg call is answered immediately and an ISDN CONNect message is

sent. This can be accomplished with a **wait 0 secs hearing music** or an **announcement** step as the first step in the receiving interflow vector.

BSR example of interflow vector on a remote Communication Manager

```
1. announcement    83345
2. consider split  2   pri m  adjust-by 0
3. consider split  11  pri m  adjust-by 0
4. queue-to best
```

The public network connects the second leg of the call to the second site and drops the trunk to Communication Manager.

BSR using EWT and agent adjustments example

A catalog company has three call centers, two in the United States and one in France. BSR is implemented across the sites. The catalog company uses the UCD-MIA call distribution method at each site and uses the UCD-MIA available agent strategy for the VDN that is active for the call. The catalog company uses the adjust-by parameter in the **consider** vector step to select the best agent at any site.

The catalog company uses the adjust-by parameter for call delivery based on the adjusted idle time for the agents, so that a remote site is not selected when agent idle time difference is not significant.

To activate **BSR Available Agent Adjustments**, set the **Available Agent Adjustments for BSR** field on the Feature-Related System Parameters screen to `y`.

To use **BSR Available Agent Adjustments**, you must change the adjust-by value in the **consider** vector steps to include a percentage adjustment appropriate for each call center. In this example, adjust-by values are defined as `0` for the first call center, 20 percent for the second call center, and 20 percent for the third call center. The adjustment applies if there is an agent surplus at more than two call centers.

Note:

If the agent idle time is more than 100 seconds, the idle time is decreased by the assigned percentage. If the agent idle time is less than 100 seconds, the idle time is decreased by the adjustment in seconds.

The following table summarizes how the adjustment affects the idle times for each site.

Idle time adjustment calculations				
Location	**Agent idle time (secs)**	**Adjust by xx percentage**	**Calculation**	**Adjusted idle time (secs)**
Incoming split 1 at location 1	40	0 (Since the adjust-by parameter in the **consider** step is set	0	40

Table continues…

Idle time adjustment calculations				
Location	**Agent idle time (secs)**	**Adjust by xx percentage**	**Calculation**	**Adjusted idle time (secs)**
		to zero, no adjustment is made.)		
Location 2	50	20	50 - 20	30
Location 3	100	20	100 - 20 (20 percent of 100)	80
The agent idle time at location 2 is less than 100 hence, the adjusted idle time is the difference between the two numbers. The agent idle time for location 3 is 100 hence, the adjusted idle time is reduced by 20 percent.				

Primary vector for BSR using EWT and agent adjustments example

An incoming call arrives at location 1 and the primary vector processes the call. The vector begins the BSR process by checking the specified resources. An example primary vector for incoming call processing at location 1 is shown in the following example.

Primary vector with adjustments

```
1. wait-time 0 secs hearing ringback
2. consider split    1  pri m   adjust-by 0
3. consider location 2          adjust-by 20
4. consider location 3          adjust-by 20
5. queue-to-best
```

In this example, the **consider** commands in steps 2, 3, and 4 collect information to compare local split 1 with location 2 and location 3. In each case, an available agent is found and an agent idle time returned. The adjust-by in steps 3 and 4 adjusts the value of the agent idle time. Step 5 queues the call to the best location found.

Status poll vector for BSR using EWT and agent adjustments example

To collect information from a remote Communication Manager, the **consider location 2** adjust-by `20` command in the primary vector places a call to the status poll vector on Communication Manager at location 2. An example of a status poll vector is indicated as follows:

Status poll vector

```
1. consider split 2   pri m  adjust-by 0
2. consider split 11  pri m  adjust-by 0
3. reply-best
```

The status poll returns the information to the first Communication Manager. The call does not connect to the status poll VDN.

The vector compares splits 2 and 11, identifies the best of the two splits, and sends the information back to the first Communication Manager with the **reply-best** command. You can use the adjust-by parameter on the remote Communication Manager to adjust the Expected Wait Time (EWT) or agent idle time returned by either of the splits. When you apply adjustments at the origin and remote Communication Manager, the two adjustments are added at the originating Communication Manager.

The **consider** command is ISDN-neutral and does not return answer supervision. The status poll call is dropped when the **reply-best** step executes, but the ISDN DISCONNect message returned to the first Communication Manager contains information from the best split at location 2. Once the remote Communication Manager returns the necessary information, the consider series in the primary vector on the first Communication Manager proceeds with the next vector step.

Interflow vector for BSR using EWT and agent adjustments example

Based on the values from the table, location 2 is the best site. The **queue-to best** command in the primary vector interflows the call to the interflow vector at location 2 as indicated in the following example.

Interflow vector on remote Communication Manager

```
1. consider split  2   pri m  adjust-by 0
2. consider split  11  pri m  adjust-by 0
3. queue-to best
```

The interflow vector checks the status of both splits to get the latest information and queues the call to the best split. Note that the **consider** sequences in the interflow vector and the status poll vector are identical except for the last step.

When the call is interflowed, the call is removed from the queue at the origin Communication Manager and any audible feedback at the origin Communication Manager is terminated.

Dial by Name

Use the Dial by Name feature to dial a number by entering the person's name from your touchtone keypad. You can access this feature by using Call Vectoring and the integrated announcement circuit pack. This creates an auto-attendant procedure whereby callers can enter a person's name instead of the extension number. The system processes characters of the name received, and when a match is found, the number is dialed automatically.

Note:

You must set the **Dial by Name** field to `y` to create a vector for this purpose.

A typical scenario includes the following call processing features:

- When a call comes in to the system (usually to an LDN call), a vector routes the call to an announcement that says, *Hello. You have reached A1 Hotel. Press 0 for an operator, press 1*

for front desk, press 2 if you know the extension, press 3 if you know the name, press 4 to choose from a list of extensions, or press 5 to hear the options again.

- If the caller selects 3, the caller is prompted to enter the person's name.
- As soon as a match is found, the call is placed at the person's extension.

You can assign several vectors that define how to handle calls as users select the different prompts. The following example shows 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.

Example Dial by Name vector

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

    Number: 2                 Name: Dial by Name
                   Attendant Vectoring? y             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    Holidays? y

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
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
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
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
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
```

This example includes the following call processing features and functionalities:

1. When someone calls the system, the caller receives a ringback for 2 seconds.
2. Announcement 381 plays. This announcement asks the caller to do one of the following:
   - Press 0 for an operator. If the caller presses 0 or waits for the time-out, the call is routed to the operator.
   - Press 1 for front desk. If the caller presses 1, the call is routed to extension 105, which is the front desk.
   - Press 2 if the caller knows the extension. If the caller presses 2, the call is routed to announcement 382, which prompts the caller to dial the extension.
   - Press 3 if the caller knows the person's name. If the caller presses 3, the following sub-procedure occurs:
     a. Announcement 661 plays prompting that the caller 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 ii.
        - If there is no match, go to iiii.
     b. Announcement 662 plays prompting the caller to 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 iii.
        - If there is no match, go to iv.
     c. Announcement 663 plays prompting the caller to 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 iv.
     d. Since there are still no matches, announcement 660 plays prompting the caller to press 1 to try again, or press 0 to speak to an operator.
   - Press 4 if the caller knows the department (such as housekeeping) that the caller wishes to access. If the caller presses 4, the call is routed to announcement 383, which gives the caller a list of several departments that the caller can dial directly.
   - Press 5 to start again. If the caller presses 5, the caller hears announcement 381 that repeats all of the options.
   - If the caller dials anything else, the call is routed to the operator.

Use of vectors in business scenarios

This section presents several typical business scenarios that involve telephone use. More than one vector is used in the examples to illustrate how to handle each scenario. The vectors presented here

are suggested solutions. Individual call centers use their own unique requirements and budget in selecting and writing vectors.

Emergency and routine service

For emergency calls, write a vector that plays the following announcement: *We are aware of the power outage in the northeastern part of the city. The crew has been dispatched. If you are calling for another reason, please wait for an operator.*

For non emergency calls, write a vector to offer prompts to callers to speak with an agent.

Emergency and routine service suggested solution 1

Call Vectoring

```
 1. wait-time 0 seconds 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 for an operator.]
 3. wait-time 2 seconds 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 seconds hearing music
 7. announcement 4200 [We are sorry. All our operators are busy at the moment. Please
hold.]
 8. wait-time 10 seconds hearing music
 9. goto step 7 if unconditionally
10. disconnect after announcement 4200 [We are sorry. All our operators are busy at the
moment. Please call back later.]
```

In step 2 of the example vector, the **announcement** command plays an emergency information and prompts the caller to hold if the caller wants to speak with an operator on another matter. If the caller holds, the caller hears several seconds of a ringback provided by the **wait-time** command in step 3. Thereafter, the **goto step** command in step 4 checks if 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 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 is provided in 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.

Emergency and routine service suggested solution 2

Call Vectoring and Call Prompting

```
VDN (extension=1030   name=Hub   vector=30)
Vector 30:
      1. wait-time 0 seconds 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."]
      3. route-to number 1031 with cov y if digit = 1
```

```
      4. announcement 3100 ["Invalid Entry. Please try again."]
      5. goto step 2 if unconditionally

VDN (extension=1031   name=Service   vector=31)
Vector 31:
      1. announcement 4000 ["Please hold. We will try to connect you to an operator."]
      2. wait-time 2 seconds 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 seconds hearing music
      6. announcement 4200 ["We are sorry. All our operators are busy at the moment.
Please hold."]
      7. wait-time 10 seconds hearing music
      8. goto step 6 if unconditionally
      9. disconnect after announcement 4200 ["We are sorry. All our operators are busy at
the moment. Please call back later."]
```

Suggested solution 2 involves both Call Vectoring and Call Prompting. The solution also involves two vectors instead of just one vector. The announcement portion of the **`collect digits after announcement`** command in step 2 of vector 30 plays an emergency information, then prompts the caller to press 1 if calling for some other reason.

If the caller holds the line but enters the incorrect touch-tone digit, 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 is entered, the call is not routed to the VDN extension. Instead, control is passed to step 4, where the **`announcement`** command informs the caller of the input error and prompts 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 that was entered by the caller. The digit-input loop, steps 2 through 5, continues for as long as the caller enters an incorrect digit.

If the caller correctly enters digit 1 as requested by the **`collect digits`** command in step 2, the **`route-to number`** command in step 3 sends control to the vector whose VDN extension is 1031, vector 31.

Late call treatment suggested solution

Call centers are staffed by union agents who work under a contract stipulating agents to be free at 5:00 p.m. However, some callers might call just before the closing time. Write a vector to inform such callers about the office hours.

Late call treatment

```
 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 seconds hearing ringback
 7. announcement 100 [We're sorry, all of our agents are busy. Please hold.]
 8. wait-time 998 seconds hearing music
 9. stop
10. announcement 200 [Please call back between 8:00 A.M. and 5:00 P.M., Monday through
Friday.]
11. wait-time 30 seconds hearing music
12. goto step 14 if time-of-day all 1700 to all 1710
13. goto step 11 if unconditionally
```

```
14. disconnect after announcement 300 [We are sorry, our office is now closed. Please
call back between 8:00 A.M. and 5:00 P.M.,
    Monday through Friday.]
15. disconnect after announcement 400 [We are sorry, our office is closed. Please call
back between 8:00 A.M. and 5:00 P.M.,
    Monday through Friday.]
16. disconnect after announcement 500 [We are sorry, we cannot service your call at this
time. Please call back between
    8:00 A.M. and 5:00 P.M., Monday through Friday.]
```

In the example vector, specific treatment is provided for calls that come into Communication Manager after working hours, during the weekend, or as the working day comes to a close.

The **goto step** command in step 1 checks if the call is placed during nonworking hours, during the week. If the call is received 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. If the call is not received at the time specified in Step 1, control is passed to step 2, where another **goto step** command checks if the call is received during weekend hours. If the call is received during weekend hours, a branch is made to step 15. 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. 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 if the current time is between 4:45 p.m. and 5:00 p.m. inclusive (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 an 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.

If the current time is 4:45 p.m. to 5:00 p.m. Step 5 executes a branch to step 10, where an appropriate late caller announcement plays. 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 if the time is currently any time between 5:00 p.m. and 5:10 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.

If the time is currently not between 5:00 p.m. and 5:10 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 5:00 p.m. and 5:10 p.m. (inclusive), or until the caller hangs up. Once step 12 is executed a second after 5:10 P.M., control is passed to step 14 as described previously.

Messaging option

Procedure

1. Write a vector that does the following if the oldest call waiting is in queue for longer than 75 seconds:
   - Sends the call to the messaging system.
   - Plays the following a messaging announcement: `All our MegaSports agents are busy. Please leave your name and phone number.`
2. Plays 30 seconds of ringback for the caller.
3. After ringback, plays an announcement for the caller followed by music.

Result

Suggested solution

Messaging option

```
 1. goto step 8 if oldest-call-wait in split 50 pri l > 74
 2. goto step 8 if calls-queued in split 50 pri l > 20
 3. queue-to split 50 pri l
 4. wait-time 30 seconds hearing ringback
 5. announcement 1000 [“All of our MegaSports agents are busy. Please wait.”]
 6. wait-time 998 seconds hearing music
 7. stop
 8. announcement 2000 [“We are sorry, all our MegaSports agents are busy. Please leave a
message 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 are 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.”]
```

The **`goto step`** command in step 1 of the example checks if the oldest call waiting in split 50 has been waiting for longer than 75 seconds. If so, control is passed to step 8, where the **`announcement`** command first informs the caller that all agents are busy and then requests the caller to either call back during working hours or leave a message for the agent. If the caller chooses 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 the caller can leave a recorded message. If the split queue is full, or if the AUDIX link is out of service, termination to AUDIX is unsuccessful, and vector processing continues at the next vector step. This step, as is the case here, usually contains an announcement which is played to inform the caller that all agents are busy. If the caller is successfully connected to AUDIX, vector processing terminates and a message can be left for the specified mailbox. 4000, in this case.

In step 1, if the oldest call waiting in split 50 has been waiting for fewer than 75 seconds, control is passed to step 2, where another **`goto step`** command checks the number of calls in split 50. If more than 20 calls are queued to split 50, control is passed to step 8. Thereafter, the procedure for the messaging option that is provided in the previous paragraph is implemented. If there are 20 or fewer calls waiting in split 50, control is passed to step 3, where the **`queue-to split`** command queues the call to the split.

Chapter 4: How to improve performance

Improved performance depends on the following basic principles:

- Minimize the number of vector steps to process a call.
- Do not use vector steps which have a substantial probability of failure such as the following:
  - Calls made outside of business hours.
  - Queue to groups with less than desirable resources or characteristics.

Inefficient looping wastes processing resources. For example, performance can be compromised when a vector loops through steps too often. This is especially true with long queue times.

The following are some looping examples with suggestions on how to maximize performance:

- Audible Feedback
- Look-Ahead Interflow
- Check

Examples other than looping are as follows:

- After Business Hours
- Look-Ahead Interflow

All looping examples in this section use only loops within a single vector. You must 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:

Test vectors for performance in addition to call flow.

Related links

After business hours example on page 72
Look-Ahead Interflow example on page 72
After business hours example on page 72
Look-Ahead Interflow example on page 72

Looping examples

Audible feedback examples

Note:

Evaluate the length of the wait period between repetitions of an announcement and increase the length. For optimum performance, add a second announcement after the initial announcement and repeat the second announcement less often.

In the following example, an announcement indicates all representatives as busy. Hold the announcement every 10 seconds as long as the call is in queue.

Example: 10-second announcement interval

```
1. queue-to split 1
2. announcement 2770  ["All representatives are busy. Please hold."]
3. wait-time 10 seconds hearing music
4. goto step 2 if unconditionally
5. stop
```

The following example repeats the announcement every 60 seconds, improving performance.

Example: 60-second announcement interval

```
1. queue-to split 1
2. announcement 2770    ["All representatives are busy. Please hold."]
3. wait-time 60 seconds hearing music
4. goto step 2 if unconditionally
5. stop
```

In the following example, a second announcement indicates all representatives as still being busy. Hold, in addition to the initial announcement, and repeat the second announcement less often, say after every 120 seconds, improving performance.

Example: Follow-up announcement

```
1. queue-to split 1
2. announcement 2770    ["All representatives are busy. Please hold."]
3. wait-time 120 seconds hearing music
4. announcement 2771    ["All representatives are still busy. Please continue to hold."]
5. goto step 3 if unconditionally
6. stop
```

The following table compares the relative processing cost of the three examples by looking at the approximate number of vector steps executed when processing the call. Let us say the first announcement is 3 seconds long and the second announcement is 4 seconds long. The approximate number of vector steps executed for an audible feedback is indicated in the following table.

Initial conditions	Example 10 seconds announcement interval	Example 60 seconds announcement interval	Example follow-up announcement
An agent is available in split 1	1	1	1
Queuing time of 5 minutes	70	15	9

When a call is queued for 5 minutes, the number of vector steps drops dramatically when the time between announcements is increased and drops even more when a second announcement is added and the time between announcements is increased again. 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.

Related links

How to improve performance on page 67
How to improve performance on page 67

Look-Ahead Interflow examples

Use the interflow-qpos conditional to achieve first in, first out (FIFO) or near-FIFO call processing. If you do not have the interflow-qpos conditional, add a wait period between successive LAI attempts and extend the waiting period.

The following example continuously attempts an LAI as long as the call is in queue or until a look-ahead attempt succeeds.

Example: continuous look ahead - no delay

```
1. queue-to split 1 pri l
2. announcement 3000
3. wait-time 20 seconds hearing music
4. route-to number 93035555555 cov n if unconditionally
5. goto step 4 if unconditionally
```

The following example adds a delay so that the LAI attempt occurs every 10 seconds.

Example: look ahead with a 10–second delay

```
1. queue-to split 1 pri l
2. announcement 3000
3. wait-time 20 seconds hearing music
4. route-to number 93035555555 cov n if unconditionally
5. wait-time 10 seconds hearing music
6. goto step 4 if unconditionally
```

The following example increases performance by increasing the delay between the LAI attempts to 30 seconds.

Example: look ahead with a 30–second delay

```
1. queue-to split 1 pri l
2. announcement 3000
3. wait-time 20 seconds hearing music
4. route-to number 93035555555 cov n if unconditionally
```

```
5. wait-time 30 seconds hearing music
6. goto step 4 if unconditionally
```

The following table compares the relative processing cost of the three examples by looking at the approximate number of vector steps executed when processing the call with an announcement that is 5 seconds long.

Table 1: Approximate number of vector steps executed for look-ahead interflow examples

Initial conditions	Example look ahead with no delay	Example look ahead with a 10–second delay	Example look ahead with a 30–second delay
An agent is available in split 1	1	1	1
Queuing time of 5 minutes	up to 1,000	85	30

Related links

How to improve performance on page 67
How to improve performance on page 67

Check examples

When using the **check** command to queue a call to backup splits, ensure that an adequate time has elapsed before checking the backup splits again.

Note:

With EWT, the programming style used in this example is not optimal. The best approach is to use EWT to locate an appropriate split for the call and queue the call.

Continuous check

The following example checks the 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 seconds 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
```

Check with 10 second delay

The following example 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 seconds 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 seconds hearing music
10. goto step 4 if unconditionally
```

Agent availability status cannot change every 10 seconds.

Check with 30 second delay

The following example adds a delay of 30 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 seconds 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 seconds hearing music
10. goto step 4 if unconditionally
```

The following table 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 2: Approximate number of vector steps executed for check examples

Initial conditions	Example continuous check	Example check with 10-second delay	Example check with 30-second delay
An agent is available in split 1	1	1	1
Queuing time of 5 minutes	up to 1,000	190	65

If a call is queued for 5 minutes, the number of vector steps drop dramatically under the following two conditions:

- when a delay is added before checking the backup splits again
- when the length of the delay is increased again

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.

Related links

How to improve performance on page 67
How to improve performance on page 67

After business hours example

Recommendation: Test to see if the destination resources are available (such as during business hours) before queuing.

The following example queues calls to a hunt group regardless of the time of the call. When calls reach an office after business hours, the announcement is repeated until the caller hangs up.

Unconditional queuing to hunt group

```
1. queue-to split 1
2. announcement 5000  ("All agents are busy. Please hold.")
3. wait-time 120 seconds hearing music
4. announcement 5001   ("All agents are still busy. Please continue to hold.")
5. goto step 3 if unconditionally
```

The next example tests for business hours before queuing the call. If the call reaches the office after business hours, an announcement informs the caller of the business hours and the call is terminated.

Queue to hunt group with time-of-day conditional

```
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 seconds 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.")
```

In the first example, unnecessary processing occurs when a call queues after business hours and the call is terminated only when the caller hangs up. As indicated in the second example, it is more economical to test for business hours before queuing a call.

Related links

How to improve performance on page 67
How to improve performance on page 67

Look-Ahead Interflow example

Note:

When using LAI, check if the receiving office is open for business.

Scenario: The sending Communication Manager is in Los Angeles with office hours from 8:00 AM to 5:00 PM PST and the receiving Communication Manager is in New York with office hours from 8:00 AM to 5:00 PM EST (05:00-14:00 PST). There is a three hour difference between the two locations.

The following example routes calls to the New York Communication Manager.

Unconditional LAI

```
1. queue-to split 1
2. route-to number 99145555555 cov n if unconditionally
```

```
3. announcement 2770      (All agents are busy. Please hold.)
4. wait-time 120 seconds hearing music
5. goto step 3 if unconditionally
6. stop
```

The next example tests first to see if the New York Communication Manager is open before requesting a queue to the New York Communication Manager, preventing unnecessary processing.

LAI with Time-of-Day (TOD) condition

```
1. queue-to split 1
2. goto step 4 if time-of-day is all 14:00 to all 05:00
3. route-to number 99145555555 cov n if unconditionally
4. announcement 2770        (All agents are busy. Please hold.)
5. wait-time 120 seconds hearing music
6. goto step 4 if unconditionally
7. stop
```

Refer to the next example if you enabled **Advanced Routing**. In this case, the Expected Wait Time feature can be used to determine if placing an LAI call attempt is useful.

LAI with Expected Wait Time (EWT) and TOD conditions

```
1. queue-to split 1
2. goto step 5 if expected-wait for call < 30
3. goto step 5 if time-of-day is all 14:00 to all 05:00
4. route-to number 99145555555 cov n if unconditionally
5. announcement 2770        (All agents are busy. Please hold.)
6. wait-time 120 seconds hearing music
7. goto step 5 if unconditionally
8. stop
```

In the examples, note that there is no reason to attempt an interflow if the call is answered quickly at the main Communication Manager.

Related links

How to improve performance on page 67
How to improve performance on page 67

Chapter 5: Basic Call Vectoring

Basic Call Vectoring command set

The following table summarizes the commands used for Basic Call Vectoring.

Description	Command
Treatment steps	
Play an announcement	**announcement**
Delay with audible feedback of silence, ringback, system music, or alternate audio or 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**
Routing steps	
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 that is programmed in the vector or to a Service Observing Feature Access Code	**route-to number**
Send a message to an adjunct that requests routing instructions for the call	**adjunct routing link**
Branching or programming steps	
Go to a vector step	**goto step**
Go to another vector	**goto vector**
Return vector processing to the step following the **goto** command after a subroutine call has processed	**return**
Perform arithmetic or string operation and assign resulting values to vector variables or to the digits buffer	**set**
Stop vector processing	**stop**

Treatment commands

Call treatment is a type of feedback that the caller receives if the caller is not immediately connected to an agent. Basic Call Vectoring includes the following call treatment commands:

- **`announcement`**
- **`wait-time`**
- **`busy`**
- **`disconnect`**
- **`converse-on split`**

Routing commands

Basic Call Vectoring includes the following routing commands:

- **`queue-to-split`** and **`check split`**
- **`messaging split/skill`**
- **`route-to number`**

Branching or programming commands

Basic Call Vectoring provides programming methods used within a vector either to create branching patterns in call processing flows or to stop vector processing. Basic Call Vectoring includes the following branching or programming commands:

- **`goto step`** and **`goto vector`**
- **`return`**
- **`set`**
- **`stop`**

Basic Call Vectoring considerations

- Include each vector function in a separate vector and link each function with more than one **`goto vector`** command.
- Design vector commands for maximum delay of answer supervision to keep the service costs low.
- Always provide callers with an initial feedback such as ringback.

- Pay attention to direct agent calls as these calls affect call queuing. BCMS and CMS reports do not count the queue slots occupied by direct agent calls. These calls are not counted in the total queued calls within a split and the calls-queued test condition has no effect on direct agent calls.

  For more information, see *Avaya Aura® Call Center Elite Feature Reference*.
- Use the **`duplicate vdn`** command to create duplicate VDNs from existing VDNs.
- Use the **`duplicate vector`** command to create duplicate vectors from existing vectors.

Related links

About duplicate VDNs on page 157
About duplicate vectors on page 158

Call Vectoring feature availability

Vectoring enhancement	Description
Vectoring (G3V4 Enhanced)	Provides the following capabilities: • The ability to specify a priority level with the oldest-call-wait conditional on the **`check`** and **`goto`** commands. • The use of enhanced comparators, such as <>, >=, and <=, with the **`goto`** and **`route-to`** commands, **none** as an entry for digits checking, and **active** or **latest** Vector Directory Number (VDN) thresholds for indirect VDN references. • The use of the interflow-qpos conditional with the **`goto`** and **`route-to`** commands to achieve First In First Out (FIFO) or FIFO-like call processing. • The use of wild cards in digit strings to match collected digits and Automatic Number Identification (ANI) or Information Indicator Digits (II-Digits). • The use of Vector Routing Tables to match collected digits and ANI or II-Digits. • The use of multiple audio or music sources for use with the **`wait-time`** command.
Vectoring (G3V4 Advanced Routing)	Provides the following capabilities: • Rolling Average Speed of Answer (ASA) routing • Expected Wait Time (EWT) routing • VDN calls routing The field option in the **Vectoring (G3V4 Enhanced)** field on the System-Parameter Customer-Options screen must be **y**. For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

Table continues…

Vectoring enhancement	Description
Vectoring (ANI/II-Digits Routing)	Provides the following additional capabilities. • ANI routing • II-Digits routing The field option in the **Vectoring (G3V4 Enhanced)** field on the System-Parameter Customer-Options screen must be **y**. For more information, see *Programming Call Vectoring Features in Avaya Aura® Call Center Elite*.
Vectoring (CINFO)	Provides the ability to collect Caller Entered Digits (CED) and Customer Database Provided Digits (CDPD) for a call from the network. The field option in the **Call Prompting** field on the System-Parameter Customer-Options screen must be **y**.
Vectoring (Best Service Routing)	Automatically compares splits or skills in Automatic Call Distribution (ACD) environments to find the split or skill that can provide the best service to each caller. You can operate Best Service Routing (BSR) at a singlesite or use BSR with Look-Ahead Interflow (LAI) to integrate a network of geographically distributed locations into a virtual call center.
Vectoring (Best Service Routing) without LAI enabled (singlesite BSR)	Provides the following capabilities: • The use of the **`consider split/skill`** command. • The use of the *best* keyword with **`queue-to`**, **`check`**, and **`goto`** commands. • The wait-improved conditional for **`check`** and **`goto`** commands. For a call that has already been queued, you can use the wait-improved conditional to make any subsequent queuing conditional on the improvement in Expected Wait Time (EWT) and not the EWT of the call in the current queue.
Vectoring (Best Service Routing) with LAI enabled (multisite BSR)	Provides the following capabilities: • The use of the **`consider split/skill`** and **`consider location`** commands. • The use of the **`reply-best`** command to return data to the sending switch in response to a status poll. • The use of the *best* keyword with **`queue-to`**, **`check`**, and **`goto`** commands. • The wait-improved conditional for **`check`** and **`goto`** commands. For a call that has already been queued, you can use the wait-improved conditional to make any subsequent queuing conditional on the improvement in EWT and not the EWT of the call in the current queue. Enhanced Information Forwarding sends existing and new call information, such as Universal Call ID and BSR. With Timed After Call Work (TACW), you can assign a timed ACW interval to a VDN.

Table continues…

Vectoring enhancement	Description
Vectoring (Holidays)	Simplifies vector writing for holidays. With Holiday Vectoring, you can administer 999 different Holiday Tables to make vectoring decisions. Each table can contain up to 15 dates or date ranges.
Vectoring (Variables)	Creates variables that you can use in vector commands to: • Improve the general efficiency of vector administration. • Provide increased application control over call treatments. • Create more flexible vectors to better serve the call center operations. The vector variables are defined in a central variable administration table. You can change the values assigned to some types of variables using special vectors, VDNs or Feature Access Codes (FACs). Different types of variables match different types of call processing needs. Depending on the variable type, variables can use either call-specific data or fixed values that are identical for all calls. In either case, you can reuse an administered variable in many vectors.
Vectoring (3.0 Enhanced)	Provides the following capabilities: • An increase in the number of Auxiliary (AUX) work time reason codes from the limit of 10 codes (0-9) to 100 codes (0-99). • The ability to administer the following ACD options for individual agents: - MIA Across Skills - ACW Agent Considered Idle - Aux Work Reason Code Type - Logout Reason Code Type • Forced Agent Logout from After Call Work (ACW): Automatically logs out an EAS agent who spends too much time in the ACW mode. Enhanced vectoring specifies the timeout period on a system and an agent basis. Enhanced vectoring reports the timeout with a customer-assignable reason code set on a system basis. • Location Preference Distribution: Attempts to route incoming ACD calls to agents located at the same location as the incoming trunk on which the call originated. If there is a choice, calls are routed to agents at a different location only if a locally-routed call cannot meet the administered objectives for speed of answer or service level. • Locally-sourced Music and Announcements: Call centers can use any or all the VAL or vVAL sources in the gateways as sources for the same announcement. This feature can improve the quality of the audio, reduce resource usage, and provide backup for announcements because a working announcement source with the same announcement file can be selected from the sources. For more information, see *Administering Avaya Aura® Call Center Elite*.

Table continues…

Vectoring enhancement	Description
	• Vector Subroutines: For common vector programs. Different vectors use subroutines without duplicating the same sequence in each vector. Subroutines significantly decrease the number of steps and vectors. • VDN Variables: A single vector can support multiple VDNs. • **`Return`** command: Returns vector processing to the step following the **`goto vector`** command after a subroutine is processed. • **`Set`** command: To perform numeric and digit string operations and to assign values to a user-assignable vector variable or to the digits buffer during vector processing. • The addition of the following variable types: - *ani* - *stepcnt* - *vdntime* • The addition of three registered and unregistered vector conditionals with the **`goto step`** or **`goto vector`** commands that are used to set up alternate routing of calls. The three conditionals test the type of server that is processing the vector. The conditionals also test the registration status of media gateways and port networks connected with the server. The three conditionals are as follows: - Media-gateway: Monitors the H.248 Media Gateway registration status. - Port-network: Monitors the port network gateway registration status. - Server: Monitors the type of server currently processing the vector step for the call. • VDN Time-Zone Offset: Designed for call centers with locations in different time zones. You can program a single vector that handles each time zone based on the active VDN for the call.

Chapter 6: Variables in Vectors

Use Variables in Vector (VIV) to achieve the following objectives:

- Improve the efficiency of vector administration
- Provide increased manager and application control over call treatment
- Create vectors to meet the needs of the callers and the call center operations

You can define vector variables in a central variable administration table and change the assigned values by means of special vectors, VDNs or Feature Access Codes (FACs).

Based on the variable type, variables can use either call-specific data or fixed values that are identical for all calls. In either case, you can reuse an administered variable in many vectors.

Related links

Variable parameters on page 80
Implementing vector variables on page 81
VIV requirements on page 89
VIV interactions and considerations on page 103
VIV job aid on page 106

Variable parameters

VIV enhances Call Vectoring by including letters from A to Z and AA to ZZ in many vector commands as conditionals and thresholds.

When you define vector variables in the centralized administration table, ensure that you assign an alphabetical designation to each variable. The designation can range from A to Z or from AA to ZZ . You can define up to 702 variables. Each variable can have only one definition. Once defined, the variables have the same type and assignment characteristics for every vector in which the variables are used. Depending on the variable type, specify some or all of the following parameters when you create a new vector variable.

Parameter	Description
Variable type	VIV provides a number of different variable types that you can use for different purposes. The kind of information associated with a variable can be call related, such as active vdn for the call, asai user information, or the time of day when a call arrives. You can use other types of variables to assign values and then use the as signals for high-level control over call processing operations. For example, you can use a single-digit value variable to test for operational states specific to your call center operations.
Scope	The scope of a variable indicates how variable values are assigned and used in vectors in which the variable appears. Variable scope can be local, local persistent for collect variable only, or global. Local variables use data associated with a call and the value assigned to the variable apply only within the original vector processing for the call. The value is cleared after the call leaves vector processing. Local persistent variables use data associated with a call and apply the data to more than one vector that processes the call. The last assigned value is retained throughout the life of the call. Global variables are system-wide and apply to all vectors in which the variables are used.
Length	Some variables require that you specify a string length that is applied when a value is assigned to the variable. In most cases, the string length actually represents a maximum bound, since most variables can use a value that has a shorter string length than that which is specified.
Start Position	If you create a variable that requires a string length, you must also specify a start position. The start position indicates the beginning position of the digit string to be assigned to the variable. The start position and the string length allow assigning only a portion of the data available to the variable.
Assignment	If you use a variable that has a user-defined value, provide the value in the **Assignment** field of the variables administration table.
Variable Access Code (VAC)	When you define a value variable, you can also set up an associated FAC. You can then dial the FAC to reset the variable assignment.

Related links

Variables in Vectors on page 80

Implementing vector variables

Procedure

1. Define the variable application. Determine how to use the new variable and identify the defining characteristics of the new variable.

   Use the information to identify a variable type that meets the call center needs. For an overview of variable types and purposes, see “Variables in Vectors job aid”.

2. From the system administration terminal, enter the **`change variables`** command to bring up the Variable for Vectors administration table.

3. In the Variable for Vectors administration table, select an unused variable name between A - Z or AA - ZZ. The variable name is used to represent the variable in vector steps. In the table row for the variable that you have selected, enter the following information in the specified fields:
   a. **Description** - Enter a short description.
   b. **Type**- Select the variable type.
   c. **Scope** - Local, local persistent for collect variable only, or global.
   d. **Length** - Enter length of the digit string.
   e. **Start** - Enter digit start position, first position is 1.
   f. **Assignment**- Enter an initial value.
   g. **VAC** - Optional for value variables only.

   Note:

   Based on the variable type you select, some fields can be predefined or not applicable.
4. Perform the following steps to administer a value variable type in the **VAC** field and to use a dial procedure within the local Communication Manager to change the variable assignment using an FAC:
   a. From the system administration terminal, enter the `change feature-access-codes` command. Go to page 7 of the Feature Access Code screen.
   b. Select an unused FAC and note the associated Vector Variable Code (VVx). Possible VVx values range from VV1 to VV9.
   c. In the **Code** field, enter the digits to be dialed when you access the FAC.
   d. Go back to the Variables for Vectors administration table and enter the VVx number in the VAC column for the value variable that you are administering.
5. Program more than one vector with the selected variable using **`goto`** steps and other vector commands such as **`route-to number`**.

   You must conform to the vector syntax rules specified in command syntax for vector variables.
6. You can change the variable assignments.

   Some variables such as the *ani* and *tod* variable types, do not require value reassignments after the variables are implemented in vectors, since values for the variable are always provided by individual callers or Communication Manager. However, with other variable types, you can change the variable assignment even as calls are being processed. For example, if you use a *collect* variable in a vector step, the variable value changes when an announcement prompts a caller to enter new digits or when you use a **`set`** command.

Note:

When *collect* variables are provided specifically for supervisor or manager use, the *collect* variable has a global scope and the variable is applied in a special vector intended for the supervisor or manager.

Related links

Variables in Vectors on page 80

Command syntax for vector variables

Announcement command

You can enter a vector variable between *A-Z* or *AA-ZZ* as an announcement extension in all commands that use an announcement in the extension field.

The following syntax rules apply when you use vector variables with the **announcement** command:

```
announcement [A-Z, AA-ZZ]
collect [1-16] digits after announcement [A-Z, AA-ZZ] for [A-Z, AA-ZZ]
disconnect after announcement [A-Z, AA-ZZ]wait-time
[0-999] sec hearing [A-Z, AA-ZZ] then
[music, ringback, silence, continue]
```

Requirements and considerations

The requirements for using variables in place of specific entries in vector commands are follows:

- You can use a VDN or a vector variable, but not both.
- When the command is executed, the assignment entry for that variable is based on the type assignment administered in the Variables for Vectors table or VDN screen for the active VDN for the call.
- The number that the variable expands to during vector processing must be a valid entry for the command parameter.
- The number that the variable obtains during processing must be a valid announcement extension assigned on the Audio/Announcement screen.

Related links

Variables in Vectors on page 80

Collect command

The following syntax rules apply when vector variables are used with the **collect** command.

```
collect [ced, cdpd] for [A-Z, AA-ZZ]
collect [1-16] digits after announcement [A-Z, AA-ZZ] for [A-Z, AA-ZZ]
```

Requirements and considerations

The requirements for using vector variables with the **collect** command are as follows:

- If none is specified after the for parameter, the **collect digits** command ignores the for parameter.
- The specification in the Variables tables defines what portion of the collected digits is assigned to the variable.
- The "#" digit can be collected and exist in the dial-ahead digits buffer if dialed by the caller. The "#" is assigned to a variable if that is the only digit assigned by the for parameter. This matches the threshold field with a "#" keyword.

  Example: If the caller dials "1#" and the specification for variable *B* starts at digit position 2 when length is more than 1, the single digit "#" is assigned to variable *B* by **collect 2 digits after announcement 1000** for the B command. If the dial-ahead buffer contains a "#" digit, the command **collect 1 digit after announcement 1001 for C** where C is defined as length = 1 and start = 1, then the "#" is assigned to variable C. A **goto step x if B = #** or **goto step x if C = #** is true and the branch to step x is taken. Also, the Variables for Vectors table shows the current value of "#" in the **Assignment** field. However, you cannot assign a value of "#" to a variable using an entry in the **Assignment** field. You can only assign the "#" value to the variable using the **collect … for** command.

Related links

Variables in Vectors on page 80

Converse-on command

The following syntax rules apply when you use vector variables with the **converse-on** command:

```
converse-on split [hunt group, 1st, 2nd, 3rd] pri [l, m, h, t]
passing [A-Z, AA-ZZ] and [A-Z, AA-ZZ]
converse-on split [hunt group]
pri [l, m, h, t]
passing [A-Z, AA-ZZ] and [A-Z, AA-ZZ]
```

Note:

A valid hunt group is an ACD skill or a non-ACD hunt group assigned for AUDIX, remote AUDIX, MSA, or QSIG Message Wait Indicator (MWI).

Requirements and considerations

The requirements and considerations for using vector variables with the **converse-on** command are as follows:

- You can use a variable as a data type in both passing fields. This results in outpulsing the current value of up to 16 digits for each of the specified variables as a DTMF digit stream to the VRU or IVR connected by the **converse-on** command.
- The normal **converse-on** command rules for both passing fields apply. If the variable is defined, the passed DTMF digits are the current assignment of the variable followed by a # DTMF digit. If a variable is not defined, or assigned to none or #, a single # DTMF digit is outpulsed for that data item (treated as though the data type is none) and a vector event 38 (variable not defined) or vector event 213 (no digits in variable) is logged.

Related links

Variables in Vectors on page 80

Disconnect command with vector variables

Variable syntax for the **`disconnect`** command is supported beginning with Communication Manager 3.0. You can use vector variables with the **`disconnect`** command after the announcement extension.

The following syntax rules apply when using vector variables with the **`disconnect`** command.

```
disconnect after announcement [A-Z, AA-ZZ]
```

Related links

Variables in Vectors on page 80

Goto commands

The following syntax rules apply when you use vector variables with the **`goto`** command.

`goto` step 1-99 if **or** **`goto` vector 1-8000 @ step 1-99 if**					
A-Z, *AA-ZZ*	>,<, =, <>, >=, <=	*A-Z or AA-ZZ*			
	in table	*A-Z or AA-ZZ*			
	not in table				
ani	>,>=,<>,=,<,<=	*A-Z or AA-ZZ*			
	in table				
	not in table				
available-agents	in skill	hunt group, skills for VDN: 1st, 2nd, or 3rd	>,>=,<>, =, <,<=	*A-Z or AA-ZZ*	
calls-queued	in skill	hunt group, skills for VDN: 1st, 2nd, 3rd	pri l = low m = mediumh = high t = top	>,>=,<>, =, <,<=	*A-Z or AA-ZZ*
counted-calls	to vdn	vdn extension, latest, active	>,>=,<>,=, <,<=	*A-Z or AA-ZZ*	
digits	>,>=,<>,=,<,<=	*A-Z or AA-ZZ*			

Table continues…

`goto` step 1-99 if **or** **`goto` vector 1-8000 @ step 1-99 if**					
	in table	*A-Z or AA-ZZ*			
	not in table				
expected-wait	for best	>,>=,<>,=,<,<=	*A-Z or AA-ZZ*		
	for call				
	for split	hunt group	pri l = low, m = medium, h = high, t = top	>,>=,<>, =, <,<=	*A-Z or AA-ZZ*
	for skill	hunt group, skills for VDN: 1st, 2nd, or 3rd			
holiday	in table	*A-Z or AA-ZZ*			
	not in table				
ii-digits	>,>=, <>, =, <, <=	*A-Z or AA-ZZ*			
	in table	*A-Z or AA-ZZ*			
	not in table				
interflow-qpos	>, >=, <>, =, <, <=	*A-Z or AA-ZZ*			
oldest-call-wait	in skill	hunt group, skills for VDN: 1st, 2nd, or 3rd	pri l = low, m = medium, h = high, t = top	>, >=, <., =, <, <=	*A-Z or AA-ZZ*
rolling-asa	for skill	hunt group, skills for VDN: 1st, 2nd, or 3rd	>, >=, <., =, <, <=	*A-Z or AA-ZZ*	
staffed-agents	in skill	hunt group, skills for VDN: 1st, 2nd, or 3rd	>, >=, <>, =, ,, <=	*A-Z or AA-ZZ*	
V1-V9	>, <, =, <>, >=, <=	*A-Z or AA-ZZ*			
	in table	*A-Z or AA-ZZ*			
	not in table				
wait- improved for	best	>, >=, <>, =.<, <=			*A-Z or AA-ZZ*
	skill	hunt group, skills for VDN: 1st, 2nd, or 3rd	pri l = low, m = medium, h = high, t = top	>, >=, <>, =, <, <=	

Table continues…

`goto` step 1-99 if **or** **`goto` vector 1-8000 @ step 1-99 if**					
	split	hunt group			

Requirements and considerations

The requirements and considerations for using vector variables with the **`goto`** command are as follows:

- A vector step that uses variable parameters displays command syntax similar to the following example, which tests the current number of counted calls for the active vdn to a user-defined variable “G”:

  `goto step 4 if counted-calls to vdn active <=G`

- Depending on the variable type that you use, the specifications that you provide for it, and the way in which you use it in a vector, the number of potential applications for vector variables is extremely large.

Related links

Variables in Vectors on page 80

Route-to number command

The following syntax rules apply when you use vector variables with the **`route-to number`** command.

`route-to number`	up to 16 digits (0-9) <digits>[A-Z, AA-ZZ] <digits>*<digits>A <digits>#<digits>A <digits>~p<digits>A <digits>~m<digits>A <digits>~s<digits>A <digits>~w<digits>A <digits>~W<digits>A ~r<digits>A ~r+<digits>A	with cov `y or n`	if	digit	>, >=, <>, =, <=	`0-9` or `#`
				interflow-qpos	<, =, <=	`1-9`
				unconditionally		
<digits> notation for 0 or other digits in the range of 0-9. A vector variable [A-Z, AA-ZZ] can be entered. Also shown by an “A” notation. “~r” invokes Network Call Redirection (NCR) over the incoming trunk.						

Table continues…

"~r+" invokes NCR with E.164 numbering notation for incoming SIP trunking when required by the service provider.

Requirements and considerations

The requirements and considerations for using vector variables with the **route-to** command are as follows:

- A variable can be used in the number field as the destination address for the **route-to** command. When the route-to number [A -Z, AA-ZZ] step is executed, the current numerical value or assignment of up to 16 digits is used for the destination. The variable is defined in the Variables for Vectors screen.
- A variable can be used in place of digits with all the possible special characters and digits entered before the variable.
- If the vector variable or resultant destination is not defined or is invalid, the route-to step fails, a vector event 38 (variable not defined) is logged, and vector processing continues at the next vector step. The destination number retrieved from the string of digits of the current value of the variable must be a valid destination as defined by the Communication Manager dial plan. Otherwise, the **route-to** command fails the vector event is logged, and vector processing continues at the next step.

Related links

Variables in Vectors on page 80

Set command

The following syntax rules apply when you use vector variables with the **set** command:

```
set [variables, digits] = [operand1] [operator] [operand2]
```

The following fields can consist of vector variables.

Field	Allows the following vector variables
Variables	User-assigned *A-Z* or *AA-ZZ collect* type vector variable. The *collect* variable type can be global, local, or local persistent. Only *collect* variables can be assigned to by the **set** command. Others variable types can be used as the operands but cannot be assigned a value.
Operand1	• User-assigned *A-Z* or *AA-ZZ collect* vector variable. The *collect* variable type can be either global, local, or local persistent. • System-assigned *A-Z* or *AA-ZZ* vector variables such as *ani*, *asaiuui*, and *doy*.
Operand2	

Related links

Variables in Vectors on page 80

Wait command

The following syntax rules apply when you use vector variables with the **wait** command:

```
wait-time [0-999] sec hearing [A-Z, AA-ZZ]
[music, ringback, silence, continue]
```

Related links

Variables in Vectors on page 80

VIV requirements

VIV works on all platforms and operating systems that are supported by Communication Manager 2.0 and later. VIV has the following licensing and system requirements:

The **MultiVantage G3 Version** field on the System-Parameters Customer-Options screen must have the following settings:

- The **Call Center Release** field must be set to `12.0` or later.
- The **Vectoring (Variable)** field must be set to `y`.

Related links

Variables in Vectors on page 80

Definition of local, global, and local persistent variables

The variable type of a vector determines the scope of the vector variable. Based on the variable type, the scope can be global, local, or local persistent.

Local scope

When a variable has a local scope, the value of the variable is assigned on the basis of call-specific information and is applied only in the vector that currently processes the call.

For example, *asaiuui* variables always have a local scope. If variable *B* is administered as an *ASAI* variable and included in a vector step, variable *B* is assigned the unique ASAI user data value for each new call processed by the vector.

Local persistent scope

When the scope of a *collect* variable type is “local persistent”, that is, the P scope, the variable value is assigned on the basis of call-specific information and is applied in more than one vector that process the inbound call. Unlike an L scope local variable, wherein the value is valid only until the call is being processed by the current vector, the value assigned to the P scope local persistent variable persists until the call disconnects. Applications continue to use the P scope variable and the last assigned value when the call leaves an assigned VDN. The call is either returned to the

assigned VDN through the VDN Return Destination (VRD) feature or is transferred by the answering agent to another VDN for further processing.

Note:

Use variables with local persistent scope to achieve objectives such as the following examples:

- Store the time of the day or the call duration. You can use the information to re-queue a call with higher priority if the call is transferred among many receivers.
- Track the last played announcement or the last entered collect digits. Collected digits store the number of attempts made by the customer for the life of a call.
- Count the number of VRD loops to force a call disconnection if the loop count exceeds a predetermined value. The count prevents calls from continuously reconnecting to the assigned VDN if the caller does not disconnect.

Global scope

Vector variables with a global scope have system-wide values that apply to all vectors in which the variables are used. For example, the value specified for a *tod* variable type is provided by the system clock. Despite the constantly changing value, at any given time the value is identical across all vectors that use the *tod* variable type.

For other variable types with a global scope such as *collect* or *value*, the assigned value is defined by a call center supervisor or an administrator. In this case, the user-defined value applies to all vectors that use the variable type.

About local variables

When you administer a variable with a local scope, the value assigned to the variable is provided from information that is specific to a call. Variable types that are local to the call or caller include *ani*, *asaiuui*, *collect*, *stepcnt*, *vdn*, and *vdntime*.

Note:

ASAI data for a call can be modified by a CTI adjunct when a **`route-to`** adjunct command is used.

About global variables

- Some types of global variables require that you assign values. The value that you assign applies to all vector steps in which the variable is referenced, and all calls that are executing the vector steps. When you change the value, the change is reflected in all vector steps in which the variable is referenced. This rule applies to all global variable types that allow entry in the **Assignment** field on the Variables for Vectors screen or other methods to assign a value.

  Note:

  Some variable types allow you to use the **`set`** command, the collect digits step, an FAC or the active VDN to change the specified value. When you use any of the methods to change

a variable value, the **Assignment** field on the Variables for Vectors screen updates to reflect the new variable value.

- Other types of global variables use dynamic system-retrieved values for which you cannot assign specific values. This rule applies to any global variable type that does not allow entry in the **Assignment** field of the Variables for Vectors screen such as the *tod* and *dow* variable types.

About local persistent variables

Local persistent variables have the same characteristics as local variables except that the assigned value for the *collect* type variable persists until the call disconnects.

System-assigned vector variable types

VIV provides different types of vector variables to meet various needs of call center operations.

Note:

As a call is processed through a vector or chain of vectors, the number of different variable types that can be applied is limited only by the type and number of variables that you administered.

System-assigned definition

This section describes the system-assigned vector variable types. The values for system-assigned vector variables come from the system. The values can come from any of the following methods:

- Communication Manager clock
- Data associated with a call such as asaiuui, and ani
- Processing of call such as stepcnt and vdntime

ANI type variable

The *ANI* variable provides expanded testing of the caller's phone number. When you know who called, you can route the call based on the caller's area code, prefix, or suffix.

Scope

The scope for the *ANI* variable is local.

Example

The following vector example shows how you can use an *ANI* variable to determine the area code of the caller and then route the call to an office that shares the same area code. The following variable specifications are set on the Variables for Vectors screen.

Variable	Description	Type	Scope	Length	Start
A	Concatenates the area code of the caller	*ANI*	L	3	1
C	Destination	*Collect*	L	10	1

Variable A concatenates the incoming call to an area code. For example, if the calling ANI is 3035556002, A = 303. The call is routed to C, which is set to 3035381234.

```
1. ...
2. set C = A CATR 5381234 [C = 3035381234]
3. route-to number C
```

ASAIUUI type variable

The *ASAIUUI* variable is assigned a unique value for each incoming call based on ASAI user information. Once a value is assigned, the value can be modified by an adjunct after a **`route-to adjunct`** vector step. You can also use the **`set`** command to change the assigned value. A common use for an *ASAIUUI* variable in a vector step is to test the assigned value against a threshold value.

Scope

The scope of *ASAIUUI* variables is only local.

Additional information

- Specify a start position for the *ASAIUUI* variable.
- Administer a length value for the *ASAIUUI* variable. Valid length values range from 1 to 16 digits, but if the digit length that extends from the specified start position to the end of the digit string is less than the specified length, the lesser number of digits is assigned. If the digit length that extends from the specified start position to the end of the digit string is greater than the specified length, any digits that extend the specified length are not included in the assigned value.

Example

The following example illustrates a vector step that compares an administered *ASAIUUI* variable D to a four-digit segment of the ASAI user information string that receives special call treatment if the first digit in the sequence is 3 and the last digit is 5:

```
goto step 5 if D = 3??5
```

Where D is an administered *ASAIUUI* variable and the threshold value that D is tested against is a four-digit string that begins with a 3 and ends with a 5.

Example

The following vector example illustrates how to use an *ASAIUUI* variable to provide selective customer treatment based on call-specific information.

In the example, a business wants to identify platinum member customers and provide special call treatment by queuing the customers at a higher level of priority. A CTI adjunct application uses the *ANI* data and other digits dialed by the caller to retrieve a five-digit customer account number. Account codes for platinum members are indicated by a 3 at the first digit position and a 5 at the last position in the five-digit string.

The adjunct includes the five-digit account number with other ASAI data beginning at digit position 4 in the 32-digit ASAI string.

Variable	Description	Type	Scope	Length	Start
P	Caller account code	*ASAIUUI*	L	5	4

The following example illustrates how to apply the administered *ASAIUUI* variable in a vector to implement the intended call treatment:

```
1. goto step 4 if P = 3???5
2. queue-to split 201 pri l
3. goto step 5 if unconditionally
4. queue-to split 201 pri m
5. announcement 3010
6. wait-time 30 secs hearing music
```

In the vector example, step 2 uses the *ASAIUUI* variable as a conditional value to test whether the account code for a call belongs to a platinum member (P = 3???5). If the caller is a platinum member, the call branches to step 4 where the call is placed in queue at a medium priority level. Otherwise, call control passes to step 2, which places the call in queue at a low priority level.

DOW type variable

The *DOW* variable provides the current day of the week. The assigned value can range from 1 to 7, where 1 equals Sunday, 2 equals Monday, and so forth. The values assigned to this variable are obtained from the system clock on Communication Manager.

Scope

The scope for the *DOW* variable is global.

Example

In the following example vector step, if D is the *DOW* type variable, this step verifies that the day of week is in vector routing table 1.

```
goto step 2 if D in table 1
```

The vector routing table can have certain days of the week specified - for example, Sunday = 1 and Saturday = 7. If the variable D = 1 or 7, the goto step condition passes and goes to step 2. Otherwise, the *DOW* is a weekday Monday = 2 through Friday = 6 and the goto continues to the next step.

This example works similarly for day of year and time of day.

DOY type variable

The *DOY* variable provides the current day of the year. The assigned value can range from `1 to 366`. The 366 value is provided for leap years. The values assigned to the variable are retrieved from the system clock on Communication Manager.

Important:

Leap years include an extra day (February 29). Therefore, vectors that are initially set up in non-leap years, and include *DOY* variables with assigned values greater than 59 (February 28) must be shifted forward one day when a leap year begins. Alternately, when *DOY* variables are included in vectors that are initially set up in leap years, the variables must be shifted back one day when a non-leap year begins.

If a value of 366 is assigned to a *DOY* variable and the current year is not a leap year, any **`goto`** step in which the variable is used fails.

Scope

The scope for the *DOY* variable is global only.

Example

In the following example vector step, if D is the *DOY* type variable, the step verifies a day of the year.

```
goto vector 214 if D = 45
```

The example verifies that the day is Valentine's Day. January 31 plus February 14 equals 45. If the *DOY* is Valentine's Day, the call goes to vector 214. Otherwise, the call continues processing the next step.

Stepcnt type variable

The *stepcnt* variable tracks the number of vector steps. Before the number of vector steps reaches the maximum limit, the call can be rerouted instead of being dropped. The *stepcnt* variable can also be used as a loop-control variable. By monitoring the number of vector steps, you can:

- Reroute calls before the maximum limit for the system is reached and prevent calls from being dropped.
- Reroute calls after an action has reached a predetermined limit. For example, calls can be rerouted after an announcement or music has finished playing.

You can:

- Assign a variable between *A*-*Z*, or *AA*-*ZZ*.
- Assign the number of vector steps including the current step.
- Use the variable type anywhere other vector variables or VDN variables are used.

- Use the variable type as a threshold, conditional, or destination or data number, where supported.

Scope

The scope for the *stepcnt* variable is local.

Example

The following vector example illustrates how you to use a *stepcnt* variable to break out of a vectoring loop before a step limit is reached. The following variable specifications are set on the Variables for Vectors screen.

Variable	Description	Type	Scope
C	Sets the step limit	*stepcnt*	L

In step 6, if the system reaches more than 990 vector steps, an announcement plays to inform the customer about the high volume of calls.

```
1. wait-time 0 secs hearing ringback
2. queue-to skill 100 pri l
3. wait-time 10 secs hearing ringback
4. announcement 2000
5. wait-time 60 secs hearing music
6. goto step 8 if C >= 990
7. goto step 4 unconditionally
8. announcement 3000 [We are experiencing an unusually high volume of calls, please leave
your name and number for call back]
9. messaging skill 200 for extension active
```

Note:

Use a value that is less than the maximum number of vector steps, 10,000.

TOD type variable

The *tod* variable provides the current time of day based on a 24-hour format. The assigned value, which can range from 00:00 to 23:59, is received from the Communication Manager clock. The values assigned to this variable are received from the system clock on the Communication Manager.

Communication Manager always returns four digits for the *tod* variable. This includes leading zeros where appropriate. Any comparison to the *tod* variable is also formatted as four digits. To check when the *tod* variable is after 12:30 a.m, compare to 00:30 and not 30.

Scope

The scope for the *tod* variable is global.

Example

In the following example vector step, if *D* is the *tod* variable type, this step verifies the current time of day.

`goto step` 32 if *D* >= 16:55

The example verifies that the time of day is 4:55 p.m. If the time of day is 5 minutes before closing, the call is routed to step 32. Step 32 can be an **`announcement`** step indicating that the call center is closed for the day.

VDN type variable

When a *vdn* variable is used in a **`goto`** step, the extension number value assigned to the variable is based on either the active or latest VDN associated with the call. The number of digits assigned to a *vdn* variable depend on the dial plan used for the system.

The latest value represents the VDN extension number associated with the vector currently in control of the call process and the active value represents the extension number of the current VDN, as defined by VDN override settings.

When administering the variable, you can specify whether to apply the active or latest value to *vdn* variables.

Scope

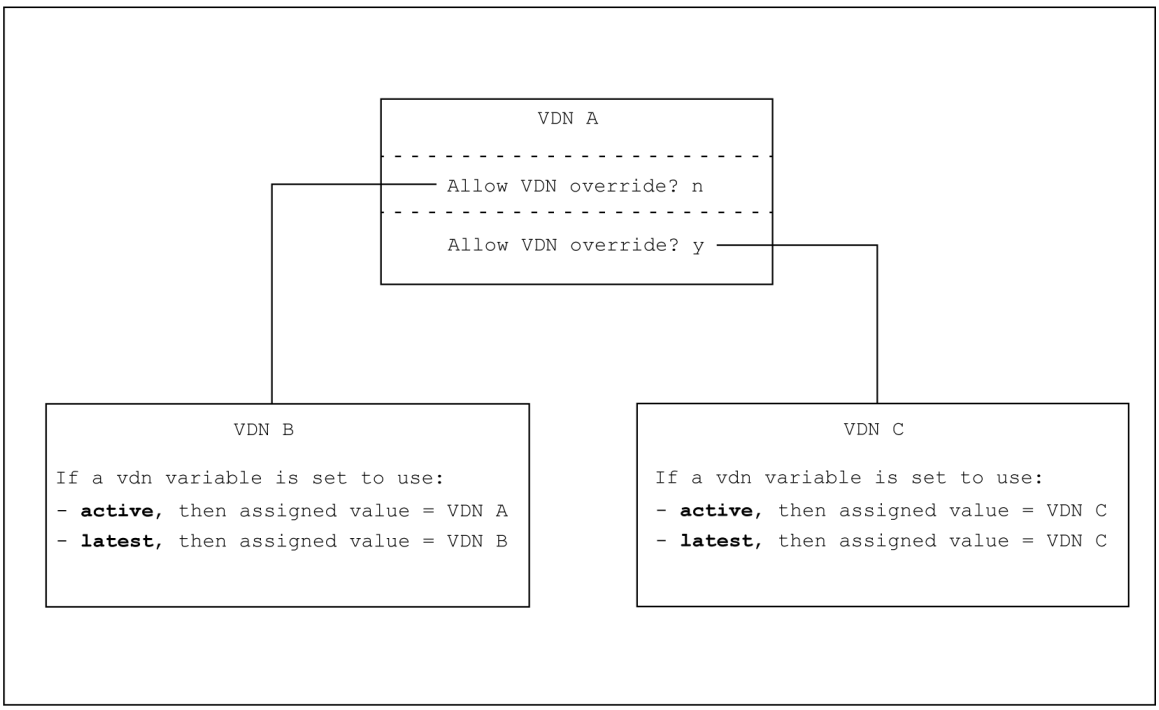
The scope for the vdn variable is only local.

Additional information

When a vdn variable is administered to use the *active* VDN of the current call as its value assignment, VDN override settings can affect the VDN extension number that is actually assigned to the variable.

When the **Allow VDN Override** field is set to `y` on the Vector Directory Number screen for a VDN, the extension number for the “subsequent” VDN to which a call is routed is applied to the call instead of the extension number for the current (latest) VDN. Therefore, the following rules apply for the value assigned to a vdn variable when it is used in a vector:

- If the VDN override setting for the previous VDN is not set to allow overrides, and a vdn variable in the vector associated with the next VDN in the call process flow is set to *active*, then the number for the previous VDN is assigned to the variable. An example of this case is represented in the following figure by the call flow from VDN A to VDN B.
- If the VDN override setting for the previous VDN is set to allow overrides, and a vdn variable used in the vector associated with the next VDN in the call process flow is set to *active*, then the current VDN number is assigned to the variable. An example of this case is represented in the following figure by the call flow from VDN A to VDN C.
- When the vdn variable is set to use the *latest* VDN number, the VDN override setting for the previous VDN has no effect on the value that is assigned to the variable. This case is represented in both of the call flows shown in the following figure.



Example

The following example shows a goto vector step that uses an administered *vdn* variable G to execute a branching step when the VDN extension 4561 is identified:

`goto step` 5 if G = 4561

VDNTime type variable

The *vdntime* variable tests the time taken by the call center to process a call. This also includes any prior time spent in a remote Communication Manager. Administrators use the *vdntime* variable to determine when alternate routing, queuing, or call treatment is needed, based on the total time the call has been in the system.

When the *vdntime* variable is tested in a vector, a value is assigned that is equal to the number of seconds the call has been active in vector processing since the call first reached a VDN. If the processing started in a remote system which forwarded the call to this system using LAI or BSR, the time spent in the prior system is included.

Scope

The scope for the *vdntime* variable is local.

Example

The following vector example shows how you can use a *vdntime* variable to remove a call from a loop after 5 minutes. The following variable specifications are set on the Variables for Vectors screen.

Variable	Description	Type	Scope
T	Call processing time	*vdntime*	L

In step 5, if the T variable is greater than 300 seconds, or 5 minutes, the vector transfers control to step 1 in vector 289.

```
1. queue-to skill 51 pri l
2. wait 30 secs hearing ringback
3. announcement 1000
4. wait-time 60 secs hearing music
5. goto vector 289 @step 1 if T > 300
6. goto step 3 if unconditionally
```

Example

You can use the same approach, as in the previous example, with BSR Local Treatment vectors to break out the local wait treatment loop when the process time of the call exceeds the tolerable time period to take back the call and provide an alternative treatment. The example can be expanded for a call take back as follows:

```
change vector 40                                                          Page 1 of 3
                                    CALL VECTOR

Number: 40                     Name: Local BSR vector
                 Attendant Vectoring? n         Meet-me Conf? n           Lock? n
    Basic? y    EAS? y   G3V4 Enhanced? y      ANI/II-Digits? y   ASAI Routing? y
Prompting? y    LAI? y  G3V4 Adv Route? y       CINFO? n BSR? y       Holidays? y

01 announcement 3000
02 consider skill 4 pri m adjust-by 0
03 consider skill 6 pri m adjust-by 0
04 consider location 1 adjust-by 10
05 consider location 2 adjust-by 10
06 queue-to best
07 announcement 3001
08 wait-time 10 secs hearing music
09 goto step 11 if T > 300
10 goto step 7 if unconditionally
11 route-to number 54010 if unconditionally
```

User-assigned vector variable types

VIV provides different types of vector variables to meet various needs of call center operations.

Note:

As a call is processed through a vector or chain of vectors, the number of different vector variable types that can be applied is limited only by the type and number of vector variables that you have administered.

User-assigned definition

You can change the value of user-assigned vector variables. By contrast, the values for system-assigned vector variables are assigned from the system clock, data about the incoming call, or by the processing of the call.

Collect type variable

One of the ways that the *collect* type variable can be assigned a value is by using the **`collect`** command. When VIV is active on the server system, the **`collect`** command includes a for parameter that precedes the *collect* variable to which you can assign user dialed data to a *collect* type variable entered in this field.

Syntax

The basic syntax for the **`collect`** command when assigning a value to a variable “V” is shown in the following example vector step:

```
collect 2 digits for V
```

where V is a vector variable of type collect, as defined in the Variables for Vectors table.

Note:

Use of variables with **`collect`** command is not required. The default entry that follows the for parameter is none.

Other ways to assign a value to a *collect* type variable is by using the Variables for Vectors table (for the globe scope only) or by assignment using the **`set`** vector command. When used with the **`set`** command the *collect* type variable serves as a general purpose variable for implementing many different kinds of applications. An example using the administration table to assign a value of 14 to variable *V* is shown in the following example excerpt from the table.

Variable	Description	Type	Scope	Length	Start	Assignment	VAC
V	Local collect variable	*collect*	G	2	1	14	NA

Note:

Local or local persistent collect type variables can be assigned using the Variables for Vectors table.

Following is an example of using the **`set`** command to assign a value 14 to a variable *V*:

```
set V = 14 ADD none
```

A *collect* variable can also be used as a threshold value in a conditional test, as shown in the following example vector step:

```
goto step 4 if counted-calls to vdn active <=V
```

Scope

The scope of collect variables can be either local (L), local persistent (P), or global (G). The following rules apply:

- If the scope is local, the assigned value is null until a value is provided during processing for the call. The assigned value is retained through all further call processing steps, including any chained vectors and **`route-to`** VDN commands, until a new value is assigned during vector processing or initial vector processing for the call is terminated, at which time the value is cleared.
- If the scope is local persistent, the assigned value is null until the value is provided during processing for the call. Unlike a variable with local scope, a collect type variable with local persistent scope persists until the call disconnects. The application can continue to use the variable when the call leaves vector processing and then returns to vector processing, such as via VDN return destination or a subsequent transfer to another VDN by the answering agent. Persistent local collect variables retain the last value assigned during vector processing, and after termination of vector processing, internal transfer to another local vector, RONA/ROIF/ROOF return to vector processing or VDN Return Destination return to vector processing, until the call disconnects.
- If the scope is global, the assigned value is retained as a system-wide variable value until it is reassigned, either by changes made to the Variable for Vectors screen, or by a collect digits/ced/cpd for [A-Z, AA-ZZ] vector step designed for that purpose.

Additional information

- When collected data or other digit sequence is assigned to a *collect* variable, the value can be truncated by specifying a start position other than the first digit in the collected data string. A start position must be specified.
- A length value must be administered. Valid length values range from 1 to 16 digits, but if the digit length from the specified start position to the end of the digit string is less than the administered length value, the lesser number of digits is assigned. If the digit length that extends from the specified start position to the end of the digit string is greater than the specified length, then any digits that extend the specified length are not included in the assigned value.
- You can administer a local *collect* variable to persist until the call disconnects. This can be used, for instance, to pass collected digits if the call is transferred to another VDN or to serve as a counter for VDN Return Destination looping to cause the call to be disconnected after a certain number of iterations. Using the *collect* variable, you can limit the VDN Return Destination looping to disconnect the call if the caller does not hang up when required with use of **`set`** command to increment the variable each time the call is returned to the VDN Return destination VDN.

Example

You can use a *collect* variable to set a threshold value that controls how call center resources are allocated to different activities. In the following example, a call center wants to be able to adjust the amount of resources that are dedicated to a promotional sales giveaway campaign so that extra resources are shifted to more profitable sales campaigns during peak call volume hours.

In this example, a *collect* variable is used as a threshold to specify the number of calls allowed for the giveaway campaign, which is initially set to a value of 50.

The *collect* variable is applied as a threshold conditional in a counted-calls vector step in such a way that it can be quickly changed when reallocation of agent resources is necessary.

The specifications that you can provide on the Variables for Vectors screen for the *collect* variable used in this example are shown in the following table.

Variable	Description	Type	Scope	Length	Start	Assignment
G	Allowed calls for give-away campaign	*collect*	G	2	1	50

After the *collect* variable G is administered, you can create a vector that uses the variable as a conditional threshold. A counted-calls step that tests the variable conditional is shown in the following example vector.

```
1. wait-time 0 secs hearing ringback
2. goto step 4 if counted-calls to vdn active <=G
3. busy
4. queue-to skill 30 pri l
5. wait-time 10 secs hearing ringback
6. announcement 1002 [All agents are busy, your call is important.]
7. wait-time 60 secs hearing music
8. goto step 6 unconditionally
```

A second vector is administered so that the call center manager can quickly change the assignment for variable G. As shown in the following example, step 4 uses a **`collect digits`** command to allow an authorized user to change the number of calls allowed for the giveaway campaign.

```
1. wait-time 0 secs hearing ringback
2. collect 4 digits after announcement 10010 for none [Enter your security code]
3. goto step 7 if digits <> 1234
4. collect 2 digits after announcement 10011 for G [Enter the number of allowed active
calls.]
5. announcement 10012 [Your change has been accepted.]
6. disconnect after announcement none
7. disconnect after announcement 10013 [The security code you entered is incorrect.]
```

Example 2

You can use a *collect* variable with scope local persistent (P) in a vector to detect callers that do not hang up when required, and to disconnect their calls. In the following example, a call center has a VDN Return Destination assigned to the incoming call VDN named VDN1. After the agent completes the call, VDN1 forwards the call to VDN2, which in turn connects the call to a survey provided by a VRU device. Callers are required to disconnect the call when the survey is complete, but some callers can still remain connected.

With a local persistent *collect* variable, you can detect the callers for whom the call has already been connected to the survey once. To count the VDN returns, you can set up a vector using the collect variable with scope local persistent.

The specifications that you can provide on the Variables for Vectors screen for the *collect* variable used in this example are shown in the following table.

Variable	Description	Type	Scope	Length	Start	Assignment
C	VDN return counter for the calls	collect	P	1	1	NA

After *collect* variable P is set up, administer C to be 0 using the following step in the vector assigned to VDN1:

```
set C = none ADD 0
```

Administer the vector assigned to VDN 2 to check the count for the P scope collect type variable C and disconnect the call if the count has reached 1. In the below vector, step 3 tests the value of C. If the value of C is greater than 0, which means the call had already been connected to the survey, processing for the call is branched to step 7. Step 7 disconnects the call without playing an announcement. Step 4 adds 1 to the collect variable C since the call is processed by the survey announcement in the next step.

```
1. wait-time 0 secs hearing ringback
2. goto step 4 if counted-calls to vdn active <=G
3. goto step 7 if C>0  [C was initialized to 0 by the vector assigned to VDN1]
4. set C = C ADD 1
5. step for connecting the caller to the survey
6. stop
7. disconnect after announcement none
```

Value type variable

With the *value* variable type, you can change vector applications from one operational mode to another. To implement value variables, perform the following steps:

- Administer a value variable in the variable administration table.
- You can administer an FAC and associate the FAC with a Variable Access Code (VAC) to use a dial code procedure to change a variable value assignment. VAC designations VV1 through VV9 are provided on the FAC screen.
- If you associate an administered *value* variable with a FAC, you can dial the FAC and enter a single digit (0 to 9) to change the variable assignment. If the variable is not associated with a FAC, you must change the variable assignment in the Variables for Vector screen.

Scope

The scope of *value* variables is global.

Reason to use

One of the reasons to use the *value* variable is to change vector processing through a manual operation. Without value vector variable, call centers have to use a dummy agent logged into a dummy skill to detect the status of a call center such as a disaster event or a closure. Now the value vector variable type can be used as the trigger that can be tested in vectoring using a **`goto vector`** command. The trigger can be set by dialing the FAC assigned to the value type variable and entering in the value that changes the vector processing for the calls to that vector. For example, *value* variable *V* can be set to 0 for normal operation and then to 1 to trigger the disaster operation using the FAC.

Additional information

- You can use the phone to access an FAC if you associate a *value* variable with an FAC. You can also change the assigned variable value. If you do not create an FAC to use with a *value* variable, the only way to change the assigned variable value is to change the **Assignment** field in the Variables for Vectors table.
- If you set up an FAC to change a value variable assignment, a station user must use a physical phone that has the required console permissions set to yes on the Class of Service screen.

- To reset the assigned value for a value variable to `null`, access the FAC associated with variable and enter `*` instead of a digit.

Example

The following example shows how to use the *value* variable *A* as a conditional in a vector step:

`goto vector` 34 if A = 2

where *A* is an administered *value* variable and the value that *A* is tested against is an arbitrary, single-digit number that represents an operational mode or condition for response in your call applications.

VIV interactions and considerations

Interactions and considerations	Description
Avaya CMS	Vector administration supports the vector variable command syntax on Avaya CMS Release 12 or later. However, the definition of each variable can only be administered through Communication Manager on the Variables for Vectors administration table. Also, if the CMS release is earlier than Release 12, an attempt to administer a vector that includes more than one vector variable generates an error message. **Note:** The specific commands for which variables are supported, depends on the CMS and Communication Manager release.
Variable failure conditions	When the tested variable conditional is not defined in the variables for vectors administration table, a **`goto`** test fails. The call does not branch and processing falls through to the next vector step.
Retention of vector variable values and assignments	The content of a local vector variable exists only while a call is in vector processing. Once a call exits vector processing, the value is cleared. Note that a call that experiences a **`converse-on`** vector command remains in vector processing. In addition, a **`route-to`** or **`adjunct routing link`** step that routes to a local VDN extension also remains in vector processing. Therefore, values that can be obtained by the call related local vector variables (*ani*, *asaiuui*, *collect*, *stepcnt*, *vdn*, and *vdntime*) and the value stored in “digits” can be used in a subsequent routed-to vector or vector steps for the same call. The value of a vector variable is not directly passed during an adjunct routing route request operation. The adjunct routing route request operation does pass the value of the “digits” buffer using the collected digits Information Element (IE). The vector **`set`** command can populate the “digits” buffer. The

Table continues…

Interactions and considerations	Description
	value determined by or assigned by using vector variables can be written to the “digits” buffer and become available to an adjunct. The **`set`** command can also be used to assign a value to the ASAI UUI string using the *asaiuui* vector variable type.
Vector Variable Usage Data	Enter the `list measurements summary` command to view the vector variable usage data on page 4 of the Measurement Summary report.

Related links

Variables in Vectors on page 80

VIV administration

Note:

For most variable types, administration is done solely in the variables administration table. However, a Feature Access Code (FAC) administration step is also required to use an FAC to change assignments for value variables.

Example Variables for Vectors screen

Use the following screen to administer vector variables.

```
change variables Page 1 of x
                                   Variables For Vectors
 Var  Description                   Type    Scope Length Start Assignment       VAC
 A    ____________________________ _______ _     __     __    _________________ ___
 B    ____________________________ _______ _     __     __    _________________ ___
 C    ____________________________ _______ _     __     __    _________________ ___
 D    ____________________________ _______ _     __     __    _________________ ___
 E    ____________________________ _______ _     __     __    _________________ ___
 F    ____________________________ _______ _     __     __    _________________ ___
 G    ____________________________ _______ _     __     __    _________________ ___
 H    ____________________________ _______ _     __     __    _________________ ___
 I    ____________________________ _______ _     __     __    _________________ ___
 J    ____________________________ _______ _     __     __    _________________ ___
 K    ____________________________ _______ _     __     __    _________________ ___
 L    ____________________________ _______ _     __     __    _________________ ___
 M    ____________________________ _______ _     __     __    _________________ ___
 N    ____________________________ _______ _     __     __    _________________ ___
 O    ____________________________ _______ _     __     __    _________________ ___
```

Required variable administration entries

The following table summarizes the information required in the various fields of the Variables for Vectors screen for the different types of variables.

Variable type	Scope	Length	Start	Assignment	VAC
ani	Local only (L)	1 to 16 digits (required)	start position from 1 to 16 (required)	—	—
asaiuui	Local only (L)	1 to 16 digits (required)	start position from 1 to 96 (required)	—	—
collect	Local, Local Persistent, or Global (L, P, or G, required)	1 to 16 digits (required)	start position from 1 to 16 (required)	Local and Local Persistent- not applicable Global - 1 to 16 digits	—
dow	Global only (G)	—	—	—	—
doy	Global only (G)	—	—	—	—
stepcnt	Local only (L)	—	—	—	-
tod	Global only (G)	—	—	—	—
value	Global only (G)	1	—	1 digit (0 to 9, optional). If you do not assign a value in this field, a null value is specified. However, if you administer an FAC to set the variable assignment, any value that you assign by dial code procedure is subsequently displayed in this field.	VVX (optional). You must enter a VAC value to use an FAC to change the variable assignment. The format for the VAC value is VVX, where *X* is a single digit that ranges from 0 to 9. The VVX value that you list in this field, must be obtained from the FAC screen after you set up the FAC. In the FAC screen, the VVX value is displayed on the same line as the FAC code. If you do not specify a VVX value when you administer the variable, you will receive an intercept tone when you attempt to dial the FAC.
vdn	Local only (L)	—	—	active or latest	—
vdntime	Local only (L)	—	—	—	—

Performing optional FAC administration for value variables

About this task

This section describes the administration steps to use value variables in vectors and to use an FAC to change the variable assignments.

Use the following screen to administer an FAC.

```
change feature-access-codes                                Page x of x

                       FEATURE ACCESS CODE (FAC)

                     Call Vectoring/Call Prompting Features

   Converse Data Return Code: ____

Vector Variable 1 (VV1) Code: ____
Vector Variable 2 (VV2) Code: ____
Vector Variable 3 (VV3) Code: ____
Vector Variable 4 (VV4) Code: ____
Vector Variable 5 (VV5) Code: ____
Vector Variable 6 (VV6) Code: ____
Vector Variable 7 (VV7) Code: ____
Vector Variable 8 (VV8) Code: ____
Vector Variable 9 (VV9) Code: ____
```

To administer an FAC, perform the following steps:

Procedure

1. On the Call Vector/Call Prompting Features page of the Feature Access Code screen, enter an FAC code in the field next to one of the Vector Access Code (VAC) entries.

   The FAC code must be a 1 to 4 digit string, but either a pound (“#”) or an asterisk (“*”) can be substituted for a numeral at the first digit position.

2. Note the VVX value associated with the new FAC code.

   Possible VAC entries range from VV1 to VV9. You must enter this value in the **VAC** field on the Variables for Vectors screen when you administer the value variable to be associated with the FAC.

VIV job aid

The following table summarizes the basic functions and characteristics of the different VIV types.

Variable type	Description	Scope	Specification	Max digit length	Assigns
ani	Holds the phone number of the caller	L	Start digit position and Length	16	Incoming call data
asaiuui	Holds call-specific user data associated with the caller	L	Start digit position and Length	16 out of a total of 96	Incoming call or ASAI application data
collect	Holds user-defined digits associated with the call for control, routing or special treatment that can be assigned a value by the Variables for Vectors table, collect digits steps or the **`set`** vector command.	L P G	Start digit position and Length	16	The for parameter of the **`collect digits`** command or assignment in the variables table
tod	Holds the current time of day in 24-hour format for processing	G	None	Always 4	The main server system clock - for example, 02:19 = 02:19 am
dow	Holds the current day of week for processing	G	None	1	The main server system clock (1-7) - for example, 1 = Sunday
doy	Holds the current day of year for processing	G	None	Always 3	The main server system clock (1-365 or 1 -366 in a leap year)
stepcnt	Provides the count of vector steps executed for the call, including the current step	L	None	4	The vector processing step counter
value	Holds a single numerical digit (0-9) for user-defined processing	G	None	1	A user-defined value entered using the VAC FAC procedure or assignment in the variables table
vdn	Holds the VDN extension number of the call for processing	L	Active or Latest VDN	13	Routing for a call
vdntime	Provides the time taken, in seconds, by a call center to process a call.	L	None	Always 4	Time in vector processing including prior processing for a call routed by BSR/LAI

Related links

Variables in Vectors on page 80

VIV vector examples

This section includes simple examples that show how vector variables can be used to help improve call processing operations.

Example application using time and day variables

- You can use *tod* and *dow* variables to create flexible vectors that evaluate factors, such as hours and days of week, so an appropriate call treatment is delivered to customers.
- You can use global *collect* variables to define call center start and close times for different days of the week. The *collect* variables provide threshold values that are tested against *tod* and *dow* values to determine appropriate call treatments.
- You can set up special VDNs to reassign variable values for the opening and closing time. For instance, when a change in daylight saving time occurs. The new variable values are instantly propagated to any number of vectors in which the variables are used.

Scenario details

The example call center has the following daily hours of operation that must be specified in the 24-hour clock time:

Day of week	Opening time	Closing time
Monday to Thursday	0700	2300
Friday	0700	2100
Saturday and Sunday	0700	1600

How to administer the variables

The specifications that you can provide on the Variables for Vectors screen for the variables used in this example are shown in the following table.

Variable	Description	Type	Scope	Length	Start	Assignment
tod or *dow* variables						
T	Current time of day	*tod*	G			Obtains the current time of day from the system clock in 0000 - 2359 format
D	Current day of the week	*dow*	G			Obtains the current day of week in 1- 7 format (1=Sunday)
Start time or close time variables						

Table continues…

Variable	Description	Type	Scope	Length	Start	Assignment
O	Opening time, all days of week	*collect*	G	4	1	0700
L (In the current example, Monday through Thursday closing time defines an upper bound on the latest possible closing time for any day of the week. Therefore, variable designation L is used to signify the latest possible closing time.)	Closing time, Monday through Thursday	*collect*	G	4	1	2300
F	Closing time, Friday	*collect*	G	4	1	2100
W	Closing time, Saturday and Sunday	*collect*	G	4	1	1600

How to create a vector to use the TOD and DOW variables

The following vector example explains how the Time of Day (TOD) and Day of the Week (DOW) variables can be used for call center business hours to control call processing in a desired manner.

```
 1. goto step 30 if T < O                     [if tod is earlier than 0700 hours, go to
the out of hours treatment]
 2. goto step  8 if T < W                     [if tod is earlier than 1600 (earliest
possible closing time), working hours apply.
                                                        Continue with step 8]
 3. goto step 30 if D = 1                      [if dow is Sunday, go to the out of hours
treatment]
 4. goto step 30 if D = 7                     [if dow is Saturday, go to the out of hours
treatment]
 5. goto step  8 if T < F                     [if tod is earlier than 2100 (Friday close
time), working hours apply]
 6. goto step 30 if D = 6                     [if tod is later than 2100 (as determined
by the preceding step), and dow is Friday,
                                                        go to the out of hours treatment]
 7. goto step 30 if T > L                     [if tod is later than 2300, go to the out
of hours treatment]
 8. goto step 31 if holiday in table 8  [based on the outcome of all preceding steps,
working hours apply unless today is a
                                                        holiday]
 9.  announcement 16549 [Please wait for the next available agent]
 10. consider skill 80 pri m adjust by 0
 11. consider location 16 adjust by 10
 12. queue-to best
 13. goto step 30 if staffed agents in skill 80 = 0
 14. wait-time 2 secs hearing silence
```

```
....
....
30. announcement 18465 [Please call again during regular business hours]
31. closed for holiday treatment
```

In the vector example, *tod*, *dow* and global *collect* variables control the call process flow by testing the call time and day values against a series of time windows that represent possible ranges of operational hours for the call center.

Steps 1 and 2 determine whether the time is within the minimum window of operational hours common to all work days, which is currently defined as 0700 to 1600 hours.

Step 1 tests whether the time is earlier than the 0700 opening time that is common to every day of the week (T < O). If the time is earlier than 0700, vector processing branches to the out of hours treatment at step 30. Otherwise, control passes to step 2.

Step 2 tests whether the time is earlier than the earliest possible closing time for any day of the week, which is 1600 on weekend days (T < W). If so, the call time is within the range of work hours that are common to all days of the week, and processing branches to step 8, which checks for a holiday before processing goes through the series of consider and queue-to best steps that are included in steps 9 through 12. Otherwise, vector processing goes to step 3 for further assessment.

Steps 3 and 4 then test whether the current day is Saturday (dow = 7) or Sunday (dow = 1). When either case is true, call control passes to the out of hours treatment provided at step 30. Otherwise, the call control passes to step 5 for further assessment.

Step 5 tests whether the time is earlier than the Friday closing time (T < F). If so, the current time is within the normal range of operating hours for Monday through Friday, and call processing branches to steps 8 through 12 for in-hours treatment. Otherwise, call vectoring goes to step 6 for further assessment.

Step 6 tests whether the day is Friday (dow = 6). If so, processing goes to out of hours treatment at step 30. Otherwise, call vectoring continues at step 7.

Step 7 completes the assessment of possible time windows by testing whether the tod is later than the latest possible closing time of 2300 hours on Monday through Thursday (T < L). If so, the call is directed the out of hours treatment provided at step 30. Otherwise, the time falls within normal work hours for Monday through Thursday and processing goes to steps 8 through 12 for in-hours treatment.

How to create a vector to reassign the hours of operation

tod and *dow* variables can be tested against *collect* variables that specify call center opening and closing time for different days of the week. Because global *collect* variables are used to specify the hours of operation, you can create a simple vector that allows the hours of operation to be changed very quickly and which is instantaneously propagated to multiple vectors.

The following example shows a vector that allows the call center opening time, which is specified by variable O in the current example, to be quickly changed by dialing a VDN dedicated for that purpose.

Note:

You must create other vectors like this one for each of the global *collect* variables that you use to set call center opening and closing time.

```
VDN 1
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 17000 for none  [Please enter your security code.]
3. goto step 6 if digits <> 12345
4. collect 4 digits after announcement 17001 for O     [Please enter your daily opening
time.]
5. disconnect after announcement 17006                 [Your change is accepted.]
6. disconnect after announcement 17010                 [You have entered an invalid
security code.]
```

Example application using a value variable

The *value* variable always has a global scope and is designed to work with FACs so that the variable assignments can be quickly changed. One application of the *value* variables is to allow multiple call applications to be quickly switched from one operational mode to another. Such a rapid switch over capability can be useful for businesses whose operations are impacted by unpredictable events. For example, a public utility can desire a switch over capacity to respond to widespread power outages associated with severe weather events.

To set up a value variable to use in multiple vectors to meet such a special switch over need, you can administer both a value variable and an associated FAC, as described in the related topics.

How to administer an FAC to use with a value variable

For this example, the FAC code is accessed when you dial `*23`. The following administration screen shows how to enable an FAC.

Note:

When you administer the FAC for a variable, note the VVX number associated with the new FAC. The VVX value must be provided in the **VAC** field on the Variables for Vectors screen, as described in *administering the value variable*.

```
change feature-access-codes                              Page x of x

                       FEATURE ACCESS CODE (FAC)

                     Call Vectoring/Call Prompting Features

   Converse Data Return Code: ____

Vector Variable 1 (VV1) Code: _*23
Vector Variable 2 (VV2) Code: ____
Vector Variable 3 (VV3) Code: ____
Vector Variable 4 (VV4) Code: ____
Vector Variable 5 (VV5) Code: ____
Vector Variable 6 (VV6) Code: ____
Vector Variable 7 (VV7) Code: ____
Vector Variable 8 (VV8) Code: ____
Vector Variable 9 (VV9) Code: ____
```

How to administer the value variable

After you set up an FAC to use with the *Value* variable, you must administer the Variables in Vectors screen to set up the value variable associated with the FAC.

Variable	Description	Type	Scope	Length	Start	Assignment	VAC
S	Switch over for blizzard	value	G	1		1	VV1

In the variable administration table, verify that the VAC code has the same value that appears with the code number on the FAC screen. If a VAC entry is not provided, you will receive an intercept tone when you dial the FAC.

How to use the value variable in multiple vectors

After you complete the required administration for the *value* variable and its associated FAC, you can use it to redirect calls from vectors used for normal operational treatments to special treatment vectors that address the switch over conditions.

The following vector step can be used in multiple vectors to implement the change in operational mode:

```
goto vector 123 @step 1 if S = 2
```

In this example, the default value for the switch over variable is administered with a value assignment of “1”, to denote normal operational modes. When a switch over due to blizzard conditions is required, the call center administrator dials *23 to access the FAC and enters the digit “2” to indicate that switch over conditions are now in effect.

Example applications using global collect variables

This section presents VIV examples using a global collect variable type instead of the single digit value variable type. The value variable allows an FAC assignment so that the “value” of the variable can easily be changed by a dialing sequence. The global collect variable can be used in the same way, that is, by dialing a VDN instead of an FAC.

Global collect variables offer many advantages over the value variable, including:

- Global collect variables are not limited to nine FACs.
- The ability to add a security code to prevent malicious or accidental changes to the call flows.
- Creation of a VDN/vector menu that prompts for the variable to change (multiple variables via one VDN).
- Call flow routing can be changed remotely by calling the VDN rather than through remote access to dial an FAC.
- Additional features such as feedback announcements and confirmation.

How to verify a password and change a value

In the example, the global collect variable *A* is used in call center vectors to control the flow that a call experiences. The variable is assigned using the `change variable` command and is given a type *collect* and a scope of “G” for global. The other settings depend on how the variable is used. The length is 1 to 16 with a start position of “1”.

The vector prompts the user for a 16–digit password. The password is compared to the expected value contained on the active VDN as VDN variable “V2”. Step 8 prompts the user to enter new value for variable *A*. The announcement is recorded to list the expected values along with the use of each value.

The following example illustrates one approach to accomplish the tasks. The vector flow can also be written using vector subroutines for entry confirmation.

```
01 wait-time 2 secs hearing ringback
02 # Entry and Validation of Security Code
03 collect 16 digits after announcement V1 for none
04 goto step 6 if digits = V2
05 disconnect after announcement none
06 # This vector modifies the value of Variable A for global call flow
07 # control. Enter 111-Normal Ops, 222-Evacuation, 333-Severe Impairment
08 collect 3 digits after announcement V3 for A
09 # Goodbye
10 stop
```

How to add change confirmation

This example adds the following to the previous example:

- Announcement of the entered value.
- Change verification.
- Retry loop.

For this example, variable A is defined as a global collect variable of length 3 and a start of 1, and variable B (used as a temporary variable) is defined as a local *collect* variable of length 3 and a start of 1.

```
01 wait-time 2 secs hearing ringback
02 # Entry and Validation of Security Code
03 collect 16 digits after announcement V1 for none
04 goto step 6 if digits = V2
05 disconnect after announcement none
06 # This vector modifies the value of Variable A for global call flow
07 # control. Enter 111-Normal Ops, 222-Evacuation, 333-Severe Impairment
08 # or Enter 000 to exit
09 collect 3 digits after announcement V3 for B
10 goto step 32 if B = 000
11 # Play announcement to inform user what value they entered.
12 goto step 15 if B <> 111
13 announcement 61111
14 goto step 24 if unconditionally
15 goto step 18 if B <> 222
16 announcement 61112
17 goto step 24 if unconditionally
18 goto step 21 if B <> 333
19 announcement 61113
20 goto step 24 if unconditionally
21 # Non-valid digit string was entered announcement, please try again
```

```
22 announcement 61114
23 goto step 9 if unconditionally
24 # Please confirm that this is the desired value 0-no, 1-yes
25 collect 1 digits after announcement 61115 for none
26 goto step 30 if digits <> 1
27 set A = B ADD none
28 # Play announcement that value was changed and then disconnect.
29 disconnect after announcement 61116
30 # Value was not confirmed or incorrect - try again
31 goto step 9 if unconditionally
32 disconnect after announcement none
```

Example applications using vdn type variables

Use the *vdn* variable type to reduce the number of vectors required to provide differential treatment to specific service VDNs. The following examples show different ways to use *vdn* type variables to create a single vector that can be used by multiple VDNs, even as you maintain the ability to provide differential call treatment based on VDN identity.

The following table shows the specifications that you must enter on the Variables for Vectors screen for a *vdn* type variable that are used in the vector examples in this section.

Variable	Description	Type	Scope	Length	Start	Assignment
Y	VDN for DNIS testing	vdn	L			active

This first example shows how the administered *vdn* type variable *Y* can be used in a single vector to provide multiple announcement treatments based on call identity. Vector processing for the call proceeds through a series of paired **`goto`** and **`announcement`** steps that attempt to match the call VDN with an appropriate announcement.

```
1. goto step 3 if Y <> 1001
2. announcement 2001
3. goto step 5 if Y <> 1002
4. announcement 2002
5. goto step 7 if Y <> 1003
6. announcement 2003
7. goto step 9 if Y <> 1004
8. announcement 2004
9. queue-to skill 50
```

In step 1, the call-specific value for the *vdn* type variable *Y* is compared to one of several possible administered VDN values (Y <> 1001). If the value for *Y* matches the specified VDN value, an announcement treatment specific to that VDN is provided in step 2. Otherwise, vector processing branches from step 1 to the next test or announcement pair and proceeds until the caller receives an appropriate announcement treatment.

The following example shows another way that the *vdn* type variable can be applied in a vector to implement selective call treatment. In this example, the *vdn* type variable assigned to the call is tested against a VDN to distinguish local and non-local callers.

```
1. wait 0 secs hearing ringback
2. goto step 4 if Y = 4561  [VDN for 800 number callers]
3. announcement 2700  [Our store is located at 1300 West 120th Avenue.]
4. queue-to skill 30 pri l
5. wait-time 5 secs hearing ringback
```

```
6. announcement 1002  [All agents are serving other customers, please wait.]
7. wait-time 60 secs hearing music
8. goto step 6 if unconditionally
```

As shown above, step 2 tests whether the value assigned to the *vdn* type variable is equal to the VDN associated with 800-number callers (Y = 4561). If so, call control branches to step 4. Otherwise, call control passes to step 3, which provides an announcement intended specifically for local callers.

Example application using a vector variable in other commands

A vector variable can be used in the **route-to** number command to route a call to a destination provided indirectly through user input (collect type), a vdn type (active or latest VDN extension for the call) or from ASAI UUI (user data) associated with the call. This example uses a destination address provided remotely by ASAI UUI included with the call. The variable R defines the portion of the ASAI UUI digit string to use as the **route-to** number.

Variable	Description	Type	Scope	Length	Start
R	Alternate route to destination	asaiuui	L	5	3

You can use the *asaiuui* type variable R in a vector to route the call to the destination defined by a remote location if the number of staffed agents is less than a certain number. If the number of staffed agents is less than 100, the call is routed to the 5-digit destination indicated in the ASAI UUI, forwarded with the call from the remote location. Otherwise, the call must be put in queue for handling at the current location.

```
1. wait-time 0 secs hearing ringback
2. goto step 8 if staffed-agents in skill 22 < 100
3. queue-to skill 22 pri l
4. wait-time 6 secs hearing ringback
5. announcement 2001
6. wait-time 60 secs hearing music
7. goto step 3 unconditionally
8. route-to number R
9. goto step 3 unconditionally
```

At step 8, the variable R is assigned 5 digits of the call's ASAI UUI data digit string starting from digit position 3. This 5-digit number is used as the destination for the **route-to** command. Step 9 provides backup in case the **route-to** number command fails due to an empty ASAI UUI digit stream or the number obtained is an invalid destination.

Example application using a vector variable in the converse-on command

Including a vector variable in the **converse-on** command as a data item to pass (outpulse) to the VRU or IVR allows forwarding of additional data that is not currently a supported data type. You can define the variable as any of the existing variable types, such as *collect*, *value*, *tod*, *doy*, *dow*, and *asaiuui*. You can use the *asaiuui* type to forward data provided by a remote site or local ASAI

interfaced application or with assignment using the **set** command. For this example, variable D forwards numerical account code data of up to 6 digits provided by an ASAI application.

Variable	Description	Type	Scope	Length	Start
D	ASAI provided data	asaiuui	L	6	1

The ASAI application uses adjunct routing to reach VDN2 that is assigned to the following vector. The data is included as ASAI UUI in the route-select message that routes the call to VDN2. The VRU interfaced through the **converse-on** command performs further interactive processing of the call based on the account code provided in the ASAI UUI and indicates where to next route the call.

```
1. wait-time 0 secs hearing music
2. converse-on skill 30 pri l passing vdn and D
3. collect 5 digits after announcement none for
4. route-to digits with coverage y
```

The **collect** command at step 3 collects the 5-digit destination provided by the VRU using the data return function. Step 4 routes the call to that destination.

Chapter 7: VDN variables

With VDN variables, you can:

- Assign up to nine variable fields, V1 through V9, on the VDN screen.
- Use the VDN variables in all vector commands that support variables except as a for parameter with the **`collect-digits`** command.
- Use as an operand to the **`set`** command.
- Use up to 16 digits to assign a number to the VDN variable and use up to 15 characters to describe the VDN variable.
- Use VDN variables as indirect references to announcement extensions and other numerical values in vector commands.
- The VDN variables assigned to the active VDN for the call are used in processing the vector.

You can create general purpose vectors that support multiple applications with call wait treatments customized for your application.

Call centers have many vectors that use the same basic call flow, but are unique because each requires unique announcements, route-to destinations, holiday tables, Vector Routing Table (VRT) indices, and conditional limits. With VDN variables, you can create a generic call flow vector. The unique items are designated on the VDN screen using VDN variables. VDN variables reduce the number of vectors, ensure common flows and ease of administration when the flows need to change due to unforeseen events such as problems with trunking, staffing, or messaging.

VDN variable fields

Each VDN variable field has a maximum 15-character description and 16-digit assignment as described in the following table.

Variable	Description	Assignment
V1	ABCDEFGHIJKLMNO	1234567890123456
V2... V9	ABCDEFGHIJKLMNO	1234567890123456

Use the **Description** field to describe the VDN variable using up to 15 characters.

Use the **Assignment** field to assign up to 16 digits to the VDN variable. Each digit entry can be:

- `0 – 9`

- `blank`

Using VDN variables with vector commands

You can use the VDN variables in all vector commands that support vector variables except as a for parameter with the **`collect digits`** command.

Announcement command

You can enter a VDN variable between V1 - V9 as an announcement extension in all commands that use an announcement in the extension field.

The following syntax rules apply when VDN variables are used with the **`announcement`** command.

```
announcement [V1-V9]
collect [1-16] digits after announcement [V1-V9]
for [none, A-Z, AA-ZZ]
disconnect after announcement [V1-V9]
wait-time [0-999 secs, 0-480 mins, 0-8 hrs] hearing [V1-V9] then [music, ringback,
silence, continue]
```

Requirements and considerations

- You can use a VDN variable or a vector variable, but not both.
- When the command is executed, the assignment entry for that variable is taken from the VDN screen for the active VDN of the call and used as the announcement extension number.
- The number must be a valid announcement extension assigned on the Audio/Announcement screen.

Converse-on command

The following syntax rules apply when VDN variables are used with the **`converse-on`** command.

```
converse-on skill [hunt group, 1st, 2nd, 3rd]
pri [l, m, h, t] passing [data1] and [data2]
converse-on split [hunt group] pri [l, m, h, t] passing [V1-V9] and [V1-V9]
```

Note:

You can use a VDN variable in data1, data2, or in both.

Disconnect command

You can use VDN variables with the **`disconnect`** command after an announcement extension.

The following syntax rules apply when using VDN variables with the **`disconnect`** command.

```
disconnect after announcement [V1-V9]
```

Goto commands

The following syntax rules apply when using VDN variables with the **goto** command.

goto step 1-99 if **or** **goto vector 1-2000 @ step1-99 if**						
A-Z, AA-ZZ	>,<, =,<>, >=, <=	*V1-V9*				
	in table	*V1-V9*				
	not in table					
ani	>, >=, <>, =, <, <=	*V1-V9*				
	in table	*V1-V9*				
	not in table					
available-agents	in skill	hunt group, skills for VDN: 1st, 2nd, 3rd	<, >=, <>, =, <, <=	*V1-V9*		
calls-queued	in skill	hunt group, skills for VDN: 1st, 2nd, 3rd	pri	priorities: l = low, m = medium, h = high, t = top	>, >=, <>, =, <, <=	*V1-V9*
counted-calls	to vdn	vdn extension, latest, active	>, >=,<>, =, <, <=		*V1-V9*	
digits	>, >=,<>, =, <, <=	*V1-V9*				
	in table	*V1-V9*				
	not in table					
expected-wait for	best	>, >=, <>, =, <, <=	*V1-V9*			
	for call					
	for split	hunt group	pri	priorities: l = low, m = medium, h = high, t = top	>, >=, <., =, <,<=	*V1-V9*
	for skill	hunt group, skills for VDN: 1st, 2nd, 3rd				
holiday	in table	*V1-V9*				
	not in table					
ii-digits	>, >=, <>, =, <, ,=	*V1-V9*				

Table continues…

`goto` step 1-99 if **or** **`goto` vector 1-2000 @ step1-99 if**						
	in table	*V1-V9*				
	not in table					
interflow-qpos	>, >=, <>, =, <, <=	*V1-V9*				
oldest-call-wait	in skill	hunt group, skills for VDN: 1st, 2nd, 3rd	pri	priorities: l = low, m = medium, h = high, t = top	>, >=, <>, =, <=	*V1-V9*
rolling-asa	for skill	hunt group, skills for VDN: 1st, 2nd, 3rd	>, >=, <>, =, <,<=	*V1-V9*		
staffed-agents	in skill	hunt group, skills for VDN: 1st, 2nd, 3rd	>, >=, <>, =, <, <=	*V1-V9*		
V1-V9	>, <, =, <., >=, <=	*V1-V9*				
	in table	*V1-V9*				
	not in table					
wait-improved for	best	>, >=, <>, =, <, <=	V1-V9			
wait- improved for	best	>, >=, <>, =, <, <=				*V1-V9*
	skill	hunt group, skills for VDN: 1st, 2nd, 3rd	pri	priorities: l = low, m = medium, h = high, t = top	>, >=, <>, =, <=	
	split	hunt group				

Route-to command with VDN variables

The following syntax rules apply when VDN variables are used with **`route-to`** number commands.

`route-to`	number	*V1-V9* ~r [*V1-V9*] (~r indicates that Network Call Redirection is attempted.)	with cov	`y` `n`	if	digit	>, >=, <>, =, <=	0-9, #
						interflow-qpos	<, =, <=	1-9
						unconditionally		

Requirements and considerations

- You can use a variable in the number field as the destination address for the **route-to** command. When the route-to number [*V1-V9*] step is executed, the current numerical value of up to 16 digits is used as the destination address.
- If the variable is not defined, the route-to step fails and a “variable not defined” vector event 38 is logged. Vector processing continues at the next vector step. The destination number retrieved from the string of digits of the current value of the variable must be a valid destination as defined by the Communication Manager dial plan. Otherwise, the **route-to** command fails to log the appropriate vector event and vector processing continues at the next step.

Set command with VDN variables

The following syntax rules apply when using VDN variables with the **set** command.

```
set [variables, digits] = [operand1] [operator] [operand2]
```

You can use VDN variables in operand 1 and 2.

Wait command with VDN variables

The following syntax rules apply when VDN variables are used with the **wait** command.

```
wait-time [0-999 secs, 0-480 mins, 0-8 hrs]
hearing [V1-V9] then [music, ringback, silence, continue]
```

Case studies

Using one vector for different announcements

In this case study, agents working for the Alpha service bureau handle calls for three different companies - ABC, XYZ , and JYK. The processing for all three companies is the same, but the announcements are different.

Since the processing is identical, Alpha decides to use the same vector for all three call types. VDN variables make this possible because Alpha can use a VDN variable to define the different announcement extensions. Each call type is routed to three different VDNs - one for each company. In this example, the V1 VDN variable defines the different announcement extensions used for the initial announcement in the vector. All three VDNs are assigned to vector 5.

VDN	VDN description	V1 VDN variable assignment	Announcement
1000	ABC Company	3000	*You have reached the ABC Company …*
1001	XYZ Company	3001	*You have reached the XYZ Company …*
1002	JYK Company	3002	*You have reached the JYK Company …*

Vector 5

```
1. wait-time 0 secs hearing ringback
2. queue-to skill 10 pri l
3. announcement V1
4. wait-time 60 secs hearing music
5. announcement 3010 [All our agents are still busy.]
6. …
```

Business case

In the case study, the XYZ company has a separate vector for each application. Using VDN variables, the company can consolidate similar vectors that are each reached by a different VDN, into one vector. The company plans to use the newly-freed vectors for other applications. The problem is that the number of different parameters or values required to be assigned to the VDNs as VDN variables exceeds the limit of five variables.

The case study shows a method for combining parameter values into digit strings of up to 16 digits. Each digit string can be assigned to the VDN variables, separated into the component parts and assigned to vector variables in the common vector for each of the vector commands.

Current configuration

Before vector consolidation, all vectors had the same basic structure as shown in vector 1 for calls to VDN 1. In spite of this similarity, each vector has the following differences:

- Three different extension numbers for the announcements
- Two different Vector Routing Tables for digit checking
- Three different route-to number destinations
- A different messaging skill mailbox extension
- A different skill for queuing the call and for the messaging skill. These can be assigned using the skill preferences fields on the VDN screen.

```
Vector 1
1. wait-time 0 secs hearing ringback
2. collect 4 digits after announcement 1001 for none
3. goto vector 300 @step 1 if digits in table 11
4. goto vector 301 @step 1 if ani in table 12
5. goto step 13 if expected-wait for skill 100 pri l > 600
6. queue-to skill 100 pri l
7. announcement 1002
8. wait-time 120 secs hearing 1003 then music
```

```
9. route-to number 2001  [LAI looking for an available agent at location 1]
10. route-to number 2002 [LAI looking for an available agent at location 2]
11. route-to number 2003 [LAI looking for an available agent at location 3]
12. goto step 7 unconditionally
13. messaging skill 210 for extension 5001
```

How to assign parameters

Following are the parameters assigned for three VDNs. The parameters appear in the vector in the same order as described in the following table.

Parameter	VDN 1	VDN 2	VDN 3
announcement extension 1 for collect step	1001	1010	1100
VR table 1 for digits	11	21	31
VR table 2 for ani	12	22	32
queuing skill (first)	100	200	300
announcement 2	1002	1012	1102
audio source 3 for wait command	1003	1013	1103
route-to destination 1	2001	3001	4001
route-to destination 2	2002	3002	4002
route-to destination 3	2003	3003	4003
messaging skill hunt group (second)	210	310	410
messaging mailbox extension	5001	5002	5003

How to group parameters

One way to combine parameters is to group the parameters by function. For example, combine all announcements into one VDN variable. The following table describes this approach.

VDN variable	Parameter	VDN 1	VDN 2	VDN 3
V1	announcement extension 1 for collect step	1001	1010	1100
	announcement 2	1002	1012	1102
	audio source 3 for wait command	1003	1013	1103
V2	VR table 1 for digits	11	21	31
	VR table 2 for ani	12	22	32
V3	route-to destination 1	2001	3001	4001
	route-to destination 2	2002	3002	4002
	route-to destination 3	2003	3003	4003
V4	messaging mailbox extension	5001	5002	5003

Table continues…

VDN variable	Parameter	VDN 1	VDN 2	VDN 3
Skill Preferences	queuing skill (first)	100	200	300
	messaging skill hunt group (2nd)	210	310	410

How to assign digit strings

The string of digits to be assigned to each VDN variable on the VDN screen is described in the following table. The order is based on how the subroutine is written to separate the components. The capital letters A through H reference the vector variables that are used in the common processing vector.

VDN variable	Description	VDN 1	VDN 2	VDN 3
V1	Three announcements: A, B, C	100110021003	101010121013	110011021103
V2	Two table values: D, E	1112	2122	3132
V3	Three route-to destinations: F, G, H	200120022003	300130023003	400140024003
V4	mailbox	5001	5002	5003
Skill Preferences	1st	100	200	300
	2nd	210	310	410

Note that VDN variables V5 through V9 are not used in this example.

How to separate and assign parameters to vector variables

Vector 1 is the common vector for incoming calls that go to VDN 1, VDN 2, and VDN 3. Vector 1 is modified to include a subroutine call to vector 2 that separates the combined parameters assigned to each VDN variable and assigns the parameters to the correct vector variables in vector 1.

```
Vector 1 - revised to serve as the common vector for calls to VDN1, VDN2 and VDN3
1. wait-time 0 secs hearing ringback
2. goto vector 2 @step 1 if unconditionally
3. collect 4 digits after announcement A for none
4. goto vector 300 @step 1 if digits in table D
5. goto vector 301 @step 1 if ani in table E
6. goto step 14 if expected-wait for skill 1st pri l > 600
7. queue-to skill 1st pri l
8. announcement B
9. wait-time 120 secs hearing C then music
10. route-to number F  [LAI looking for an available agent at location 1]
11. route-to number G [LAI looking for an available agent at location 2]
12. route-to number H [LAI looking for an available agent at location 3]
13. goto step 7 if unconditionally
14. messaging skill 2nd for extension V4
```

How to define vector variables

The A through H vector variables need to be defined on the Variables for Vectors screen as the *collect* type with local scope as described in the following table. The **Assignment** and **VAC** fields are left blank.

Var	Description	Type	Scope	Length	Start
A	announcement 1	collect	local	4	1
B	announcement 2	collect	local	4	1
C	announcement 3	collect	local	4	1
D	table 1 (digits)	collect	local	2	1
E	table 2 (ani)	collect	local	2	1
F	route to 1	collect	local	4	1
G	route to 2	collect	local	4	1
H	route to 3	collect	local	4	1

How to separate each VDN variable

Vector 2 is the subroutine vector and separates each VDN variable into component parts.

```
Vector 2
1. set A = V1 SEL 12 [A = 1001 when V1 = 100110021003 since A being of length 4 is
assigned only the leftmost 4 digits]
2. set B = V1 SEL 8 [B = 1002 since SEL selects 10021003 and B being of length 4 is
assigned only the leftmost 4 digits]
3. set C = V1 SEL 4 [C = 1003 since SEL selects the rightmost 4 digits]
4. set D = V2 SEL 4 [D = 11 when V2 = 1112 since D being of length 2 is assigned only the
leftmost 2 digits]
5. set E = V2 SEL 2 [E = 12 since SEL selects the rightmost 2 digits]
6. set F = V3 SEL 12 [this step and following functions the same as for A, B, and C]
7. set G = V3 SEL 8
8. set H = V3 SEL 4
```

Summary

The case study described how to use the VDN variables to support eight parameters. The case study also described how to use variable V4 for another parameter that required to be passed with the active VDN for the call. The approach supported nine parameters with four VDN variables while keeping V5 as a spare. Since *A*-*H* vector variables are local variables, you can reuse the variables in other vector applications with similar string lengths.

Chapter 8: Vector subroutines

Subroutines use common vector programs that can be used by different vectors without duplicating the same sequence in each vector. Subroutines can significantly decrease the number of steps and vectors required.

The goto step is used for vector subroutines. The goto step uses:

- The @ step parameter to branch to a specific step in the vector
- The **`return`** command to return from a subroutine

The maximum simultaneous active subroutine calls allowed are:

- 8000 for Avaya S8500, S87XX, S88XX, and Avaya Common Server platforms
- 400 for Avaya S8300

With vector subroutines, you can reuse common sets of vector commands. For example, you can use a single subroutine for all vectors to determine if a call has arrived within business hours. Without a subroutine, each vector repeats the step.

Following are some of the advantages of using vector subroutines:

- More steps per vector by removing duplication.
- Unused steps at the end of vectors can be used for subroutines, expanding vector capacity.
- Ease of administration. You can change just one vector subroutine that is referenced by many vectors, such as changing office hours or wait treatment.

The goto command and subroutines

Use the **`goto vector`** command to branch vector processing to a subroutine or to a specific step in the vector. The **`goto vector`** command works with the **`return`** command to return vector processing to the calling vector. When a **`goto vector`** command is executed, the vector and the subsequent step number for the command are stored with the call. This is the return destination that is used with subroutines.

When the **`goto vector`** command branches to the specified vector, any data associated with the call remains with the call. Examples of call-associated data are collected digits and dial-ahead digits. Changes made stay with the call when the call returns to the calling vector.

The @ step parameter

Use the @ step parameter with the **goto vector** command conditionals to branch to a specific step in a vector. For example:

- **goto vector xxx @ step yy if <conditional> [comparator] <threshold>**
- **goto vector xxx @ step yy if unconditional**

The requirements for the @ step parameter are as follows:

- The default step number is 1. The step number remains at 1 until you change it to a step number between 2 and 99.
- When the step number is set between 1 and 99, the **goto vector** command saves the returned data when subroutines are active. Vector processing starts again at the branched-to vector at the specified step.
- If the specified step in the branched-to vector is blank, vector processing skips to the next step in the vector. If it is the last step, it is treated as a stop step.

Note:

When upgrading to Communication Manager 3.0 or later, all existing vectors with goto vector steps are converted to the goto vector xxx @ step 1 syntax.

Example 1: Test for working hours

The call center of the XYZ retail stores has a large number of vectors to check if calls arrive during working hours or not. Before the availability of vector subroutines, each vector required the same series of steps to test for working hours. With vector subroutines, only one vector is required for the series of steps that check for working hours. Each vector that requires the check uses a **goto vector** step to run the tests. Vector processing returns to the step following the calling **goto vector** step if the test passes. Otherwise, the *out-of-working* hours treatment is given by the subroutine.

The call center edits just one vector and the change is reflected in other vectors that reference this vector.

Incoming call processing vector example

```
1. wait 0 secs hearing ringback
2. goto vector 20 @step 1 if unconditionally
3. queue-to skill 100 pri l [subroutine returns here if call is during working hours]
4. announcement 1000 [
Thank you for calling XYZ Retail Stores, your call is important to us]
5. …
```

Checking working hours vector subroutine example

```
Vector 20
 1. goto step 9 if time-of-day is all 23:00 to all 07:00
 2. goto step 9 if time-of-day is Friday 21:00 to sat 07:00
 3. goto step 9 if time-of-day is sat 16:00 to sun 07:00
 4. goto step 9 if time-of-day is sun 16:00 to mon 07:00
 5. goto step 7 if holiday in table 5
 6. return [call is during working hours]
 7. announcement 2001 [The XYZ Stores are closed on holidays.]
 8. goto to step 10 if unconditionally
 9. announcement 2001 [You have called after the XYZ Stores are closed.]
10. disconnect after announcement 2001 [Please call back during normal business hours: 7
am to 11 pm on
Monday through Thursday, 7 am to 9 pm on Friday and 7 am to 4 pm on Saturday and Sunday.]
```

Chapter 9: ANI/II-digits routing and CINFO

Use the Automatic Number Identification (ANI) and Information Indicator Digits (II-digits) Call Vectoring features to make vector routing decisions based on the caller identity and the originating line. Use Caller Information Forwarding (CINFO) to collect Caller Entered Digits (CED) and Customer Database Provided Digits (CDPD) for a call from the network.

Avaya uses the term ANI to denote both ANI and Calling Line Identification (CLID) which can be used interchangeably within Vectoring. ANI and II-digits, when provided with an incoming call to a VDN, are sent to Call Management System (CMS) when vector processing starts. ANI, II-digits, and CINFO are forwarded with interflowed calls. ANI and II-digits are also passed over the Adjunct Switch Application Interface (ASAI) in event reports.

Related links

CINFO command set on page 129

CINFO command set

Command category	Action	Command
Branching/ Programming	Go to a vector step (ANI, II-digits) Go to a vector step that is based on ced or cdpd (CINFO digits)	`goto step`
	Go to another vector (ANI, II-digits) Go to another vector based on ced or cdpd. (CINFO digits)	`goto vector`
Information Collection	Pass ANI to a Voice Response Unit (VRU) Pass ced and cdpd to a VRU (CINFO)	`converse-on`
	Collect ced and cdpd from a network ISDN SETUP message.	`collect digits`
Routing	Route the call to a number that is programmed in the vector, based on ced or cdpd	`route-to number`
	Route the call to digits supplied by the network	`route-to digits`
	Request routing information from an ASAI that is based on ced or cdpd	`adjunct-routing`

Related links

ANI/II-digits routing and CINFO on page 129

ANI routing

ANI provides information on the caller's identity that can be used to improve call routing decisions. For example, calls from a specified customer can receive unique routing, local calls can be routed differently from long distance calls, or calls from different geographical areas can receive different routing. ANI can be compared against entries in a Vector Routing Table (VRT), and is supported with ISDN or SIP trunks.

ANI basics

Calling Party Number and Billing Number

ANI is based on the Calling Party Number (CPN), which is not always identical to the Billing Number. For example, if the call is placed by a user from Communication Manager, the CPN can be either the Communication Manager-based billing number or the station identification number.

String length

The ANI routing digit string can contain up to 16 digits. This supports international applications. However, ANI information in those countries that use the North American dial plan contains only 10 digits.

Call types that use ANI

The following call types have associated ANI values:

- Incoming ISDN (including PRI, BRI, and H.323) calls that send ANI.
- Incoming SIP calls that send SIP contact headers.
- Incoming R2-MFC signaling calls that send ANI.
- Distributed Communications Services (DCS) calls.
- Internal calls.

Note:

If ANI is not provided by the network for an incoming call, ANI is not available for vector processing on the call.

Use of wildcards

The **`goto...if ANI`** vector uses wildcards for either a direct comparison (‘if ani = 123+’) or comparison in a vector routing table (‘if ani in table 12’). You can use either ‘+’ or ‘?’ wildcards.

- ‘+’: Only one trailing, or starting ‘+’ is allowed per digit string and it matches zero or more digits, or a single ‘#’. The wildcard ‘+’ is not the same as the leading ‘+’ that SIP stations can send on an ANI. The leading ‘+’ makes **`goto...if ANI`** comparison fail.

- ‘?’: As many ‘?’ as possible digits placed anywhere in the digit string are allowed. A ‘?’ matches exactly one digit in the location of the ‘?’.

Use with vector routing tables

ANI data can be tested against ANI numbers provided in vector routing tables.

EAS agent calls

When an EAS agent makes a call to a VDN, the agent login ID is used as the ANI in place of the physical terminal number.

Internal transfer to VDN

When a call is transferred internally to a VDN, the following outcomes can occur:

- If the transfer completes before the call reaches the ANI conditional, the ANI value of the originator of the call is used.
- If the transfer completes after the call reaches the ANI conditional, the ANI value of the terminal that executes the transfer is used.

Tip:

To ensure that the ANI of the originator is preserved during a transfer, add a filler step such as wait with silence, to the beginning of the vector. In this way, a transfer can be completed before the ANI conditional is encountered.

Use of ANI with Vector Routing Tables

You can test ANI against entries in a Vector Routing Table (VRT). VRT contain lists of numbers that can be used to test a **`goto...if ani`** command. ANI can be tested to see if it is either in or not-in the specified table. Entries in the tables can also use the “+” and “?” wildcards.

The following sample VRT includes various area codes for the state of California.

```
                    VECTOR ROUTING TABLE
  Number: 6               Name: California           Sort? n

1:  714+                                  17: _______
2:  805+                                  18: _______
3:  619+                                  19: _______
4:  707+                                  20: _______
5:  209+                                  21: _______
6:  310+                                  22: _______
7:  213+                                  23: _______
8:  408+                                  24: _______
9:  510+                                  25: _______
10: 818+                                  26: _______
11: 909+                                  27: _______
12: 916+                                  28: _______
13: 415+                                  29: _______
```

The following vector example shows how calls can be routed based on information provided in the sample VRT.

```
1. announcement 45673
2. goto step 9 if ani = none
3. goto vector 8 if ani in table 6
```

```
4. queue-to split 5 pri l
5. wait-time 10 seconds hearing ringback
6. announcement 2771
7. wait-time 10 seconds hearing music
8. goto step 6 if unconditionally
9. route-to number 0 with cov y if unconditionally
```

In the example vector, the call is routed to an operator if no ANI is available for the call. If the first three numbers match an area code from table 6, the call is routed to vector 8. All other calls are queued.

Use of ANI without Vector Routing Tables

```
1. wait-time 4 secs hearing silence
2. goto step 13 if ani = none
3. goto step 12 if ani = 3035367326
4. goto vector 74920 if ani <= 9999999
5. goto vector 43902 if ani = 212+
6. goto vector 43902 if ani = 202+
7. wait-time 0 seconds hearing ringback
8. queue-to split 16 pri m
9. wait-time 120 seconds hearing 32567 then continue
10. announcement 32456
11. goto step 9 if unconditionally
12. route-to number 34527 with cov y if unconditionally
13. route-to number 0 with cov n if unconditionally
14. busy
```

In step 2, calls that do not have an associated ANI are routed to an operator. In step 3, calls are routed from a phone to a specified extension. In step 4, operator or attendant local calls are routed, which are calls with less than seven digits, to a different vector. In steps 5 and 6, calls are routed from area codes, that is, Numbering Plan Areas (NPAs) 212 and 202 to a different vector. Calls that are not rerouted by the previous vector steps are put in a queue.

II-digits routing

II-digits provide information on the originating line for a call. You can use the information for some of following purposes:

- Detect fraudulent orders for catalog sales, travel reservations, money transfers, and traveler's checks.
- Assign priority or special treatment to calls that are placed from pay phones, cellular phones, and motel phones. For example, an automobile emergency road service can prioritize calls placed from pay phones.
- Detect calls placed from pay phones when the caller intentionally tries to prevent being tracked by collection agencies or dispatching services.
- Convey the type of originating line on the agent display by routing different type calls to different VDNs.

II-digits basics

String description	II-digits is a 2-digit string that ISDN PRI provides for an incoming call. II-digits delivery is a widely available ISDN PRI AT&T Network service. This service is bundled with ANI delivery and tariffed under the MEGACOM 800® and MultiQuest 800® INFO-2 features to provide information on call origination. R2-MFC Call Category digits, when available, are treated as II-digits for routing. Leading zeros are significant. For example, the II-digits value 02 that is associated with a call does not match the digit string 2 in a vector step.
Use with a VRT	As is true for ANI routing and collected-digit routing, II-routing digits can be compared against entries in a Vector Routing Table.
Use of wildcards	The II-digits string used in a vector step or a vector routing table can contain either the + or ? wildcard.
VDN Return Destination preservation	When a call is returned to vector processing as a result of the VDN Return Destination feature, the II-digits are preserved.
Call types associated with II-digits	The following calls have associated II-digits values: • Incoming ISDN PRI calls that include II-digits. • Incoming ISDN PRI Tie Trunk DCS or non-DCS calls that include II-digits. **Note:** Since tandeming of II-digits is only supported if the trunk facilities used are ISDN PRI, traditional DCS does not support II-digits transport but DCS Plus, that is, DCS over PRI, supports II-digits transport.
Internal transfer to a VDN	When a call with II-digits is transferred internally to a VDN, the following outcomes can occur: • If the transfer completes before the call reaches the II-digits conditional, the II-digits value of the originator of the call is used. • If the transfer completes after the call reaches the II-digits conditional, the II-digits value of the terminal executing the transfer is used. Under normal circumstances, there are no II-digits for a terminal that executes a transfer. **Tip:** To ensure that the originator II-digits is preserved, add a filler step such as **`wait with silence`** to the beginning of the vector. In this way, a transfer can be completed before the II-digits conditional is encountered.

II-digits codes

Note:

The North American Numbering Plan Administration (NANPA) maintains the II-digit assignments. Visit http://www.nanpa.com/number_resource_info/ani_ii_assignments.html to view the latest II-digit assignments and descriptions.

II-digits assignments	
II-digits	**Description**
00	Plain Old Telephone Service (POTS) - non-coin service requiring no special treatment
01	Multiparty line (more than 2) - ANI cannot be provided on 4 or 8 party lines. The presence of this 01 code causes an Operator Number Identification (ONI) function to be performed at the distant location. The ONI feature routes the call to a CAMA operator or to an Operator Services System (OSS) for determination of the calling number.
02	ANI Failure - the originating switching system indicates (by the 02 code), to the receiving office that the calling station has not been identified. If the receiving switching system routes the call to a CAMA or OSS, the calling number can be verbally obtained and manually recorded. If manual operator identification is not available, the receiving switching system, for example, an interLATA carrier without operator capabilities, can reject the call.
03-05	Unassigned
06	Station Level Rating - The 06 digit pair is used when the customer has subscribed to a COS in order to be provided with real time billing information. For example, hotel or motels, served by PBXs, receive detailed billing information, including the calling party's room number. When the originating switching system does not receive the detailed billing information, for example, room number, this 06 code allows the call to be routed to an operator or operator services system to obtain complete billing information. The rating or billing information is then provided to the service subscriber. This code is used only when the Directory Number (DN) is not accompanied by an automatic room or account identification.
07	Special Operator Handling Required - calls generated from stations that require further operator or OSS screening are accompanied by the 07 code. The code is used to route the call to an operator or OSS for further screening and to determine if the station has a denied-originating COS or special routing or billing procedures. If the call is unauthorized, the calling party is routed to a standard intercept message.
08-09	Unassigned
10	Not assignable - conflict with 10X test code
11	Unassigned
12-19	Not assignable - conflict with international outpulsing code
20	Automatic Identified Outward Dialing (AIOD) - without AIOD, the billing number for a PBX is the same as the PBX Directory Number (DN). With the AIOD feature, the originating line number within the PBX is provided for charging purposes. If the AIOD number is available when ANI is transmitted, code 00 is sent. If not, the PBX DN is sent with ANI code 20. In either case, the AIOD number is included in the AMA record.

Table continues…

II-digits assignments	
II-digits	**Description**
21-22	Unassigned
23	Coin or Non-Coin - on calls using database access, for example 800, ANI II 23 is used to indicate that the coin or non-coin status of the originating line cannot be positively distinguished for ANI purposes by the SSP. The ANI II pair 23 is substituted for the II pairs which otherwise indicates that the non-coin status is known, that is 00, or when there is ANI failure. ANI II 23 can be substituted for a valid 2-digit ANI pair on 0-800 calls. In all other cases, ANI II 23 must not be substituted for a valid 2-digit ANI II pair which is forward to an SSP from an EAEO. Some of the situations in which the ANI II 23 can be sent: • Calls from non-conforming end offices (CAMA or LAMA types) with combined coin/non-coin trunk groups. • 0-800 Calls • Type 1 Cellular Calls • Calls from PBX Trunks • Calls from Centrex Tie Lines
24	Code 24 identifies a toll free service call that has been translated to a POTS routable number using the toll free database that originated for any non-pay station. If the received toll free number is not converted to a POTS number, the database returns the received ANI code along with the received toll free number. Code 24 indicates a toll free service call since that fact can no longer be recognized simply by examining the called address.
25	Code 25 identifies a toll free service call that has been translated to a POTS routable number using the toll free database that originated from any pay station, including inmate telephone service. Specifically, ANI II digits 27, 29, and 70 are replaced with Code 25 under the stated condition.
26	Unassigned
27	Code 27 identifies a line connected to a pay station which uses network provided coin control signaling. II 27 is used to identify this type of pay station line irrespective of whether the pay station is provided by a LEC or a non-LEC. II 27 is transmitted from the originating end office on all calls made from these lines.
28	Unassigned
29	Prison or Inmate Service - the ANI II digit pair 29 is used to designate lines within a confinement or detention facility that are intended for inmate or detainee use and require outward call screening and restriction, for example, 0+ collect only service. A confinement or detention facility can be defined as including, but not limited to, Federal, State or Local prisons, juvenile facilities, immigration and naturalization confinement or detention facilities, which are under the administration of Federal, State, City, County, or other Governmental agencies. Prison or Inmate Service lines are identified by the customer requesting such call screening and restriction. In those cases where private pay stations are located in confinement or detention facilities and the same call restrictions applicable to Prison or Inmate Service required, the ANI II digit for Prison or Inmate Service applies if the line is identified for Prison or Inmate Service by the customer.

Table continues…

II-digits assignments	
II-digits	**Description**
30-32	Intercept - where the capability is provide to route intercept calls, either directly or after an announcement recycle, to an access tandem with an associated Telco Operator Services System, the following ANI codes must be used: • 30 - Intercept (blank) - for calls to unassigned DN. • 31 - Intercept (trouble) - for calls to DNs that have been manually placed in trouble-busy state by Telco personnel. • 32 - Intercept (regular) - for calls to recently changed or disconnected numbers.
33	Unassigned
34	Telco Operator Handled Call - after the Telco Operator Services System has handled a call for an IC, it can change the standard ANI digits to 34, before outpulsing the sequence to the IC, when the Telco performs all call handling functions, for example, billing. The code tells the IC that the BOC has performed billing on the call and the IC only has to complete the call.
35-39	Unassigned
40-49	Unrestricted Use - locally determined by carrier
50-51	Unassigned
52	Outward Wide Area Telecommunications Service (OUTWATS) - this service allows customers to make calls to a certain zone(s) or band(s) on a direct dialed basis for a flat monthly charge or for a charge based on accumulated usage. OUTWATS lines can dial station-to-station calls directly to points within the selected band(s) or zone(s). The LEC performs a screening function to determine the correct charging and routing for OUTWATS calls based on the customer's COS and the service area of the call party. When these calls are routed to the inter-exchange carrier using a combined WATS-POTS trunk group, identify the WATS calls with the ANI code 52.
53-59	Unassigned
60	TRS - ANI II digit pair 60 indicates that the associated call is a TRS call delivered to a transport carrier from a TRS provider and that the call originated from an unrestricted line, that is, a line for which there are no billing restrictions. Accordingly, if no request for alternate billing is made, the call is billed to the calling line.
61	Cellular or Wireless PCS (Type 1) - The 61 digit pair is to be forwarded to the inter-exchange carrier by the local exchange carrier for traffic originating from a cellular or wireless PCS carrier over type 1 trunks. Note: ANI information accompanying digit pair 61 identifies only the originating cellular or wireless PCS system, not the mobile directory placing the call.
62	Cellular or Wireless PCS (Type 2) - The 62 digit pair is to be forwarded to the inter-exchange carrier by the cellular or wireless PCS carrier when routing traffic over type 2 trunks through the local exchange carrier access tandem for delivery to the inter-exchange carrier. Note: ANI information accompanying digit pair 62 identifies the mobile DN placing the call but does not necessarily identify the true call point of origin.
63	Cellular or Wireless PCS (Roaming) - The 63 digit pair is to be forwarded to the inter-exchange carrier by the cellular or wireless PCS subscriber roaming in another cellular or wireless PCS network, over type 2 trunks through the local exchange carrier access tandem for delivery to the inter-exchange carrier. Note: Use of 63 signifies that the called number is

Table continues…

II-digits assignments	
II-digits	**Description**
	used only for network routing and must not be disclosed to the cellular or wireless PCS subscriber. Also, ANI information accompanying digit pair 63 identifies the mobile DN forwarding the call but does not necessarily identify the true forwarded-call point of origin.
64-65	Unassigned
66	TRS - ANI II digit pair 66 indicates that the associated call is a TRS call delivered to a transport carrier from a TRS provider and that the call originates from a hotel or motel. The transport carrier can use this indication, along with other information, for example, whether the call was dialed “1+” or “0+”, to determine the appropriate billing arrangement.
67	TRS - ANI II digit pair 67 indicates that the associated call is a TRS call delivered to a transport carrier from a TRS provider and that the call originated from a restricted line. Accordingly, sent paid calls must not be allowed and additional screening, if available, must be performed to determine the specific restrictions and type of alternate billing permitted.
68-69	Unassigned
70	Code 70 identifies a line connected to a pay station, including both coin and coin-less stations, which does not use network provided coin control signaling. II 70 is used to identify this type pay station line irrespective of whether the pay station is provided by a LEC or a non-LEC. II 70 is transmitted from the originating end office on all calls made from the lines.
71-79	Unassigned
80-89	Reserved for future expansion to a 3-digit code
90-92	Unassigned
93	Access for private virtual network types of service: the ANI code 93 indicates, to the IC, that the originating call is a private virtual network type of service call.
94	Unassigned
95	Unassigned - conflict with test codes 958 and 959
96-99	Unassigned

II-digits routing example

The following vector example illustrates the branching of calls that use II-digits to route to different VDNs. In this example, VDN Override is set to `yes`. In this way, the VDN name or VDN of Origin Announcement (VOA) can be used to convey to the agent the type of II-digits that are associated with the call.

```
1. goto step 9 if ii-digits = none
2. goto step 10 if ii-digits = 00
3. goto step 11 if ii-digits = 01
4. goto step 12 if ii-digits = 06
5. goto step 13 if ii-digits = 07
6. goto step 13 if ii-digits = 29
7. goto step 14 if ii-digits = 27
8. goto step 15 if ii-digits = 61
9. route-to number 1232 with cov n if unconditionally
10. route-to number 1246 with cov n if unconditionally
11. route-to number 1267 with cov n if unconditionally
12. route-to number 1298 with cov n if unconditionally
```

```
13. route-to number 1255 with cov n if unconditionally
14. route-to number 1298 with cov n if unconditionally
15. route-to number 1254 with cov n if unconditionally
```

In the example, if the call has no II-digits, step 1 branches to step 9, which routes the call to extension 1232. If the call has II-digits, steps 2 through 8 are used to route calls with different II-digits to various extensions.

CINFO

Use Caller Information Forwarding (CINFO) to associate ceds and cdpds with several vector commands to improve call processing.

The network-provided ISDN PRI SETUP message for a call includes ced and cdpd data when both of the following conditions are met:

- The incoming trunk is enabled for ISDN-PRI.
- The network uses AT&T Network Intelligent Call Processing (ICP) service.

CINFO basics

UEC IE storage

When an ISDN call is received from either the AT&T network or a tandemed PRI call, Communication Manager stores the codeset 6 User Entered Code Information Element (UEC IE) when it contains the ced or cdpd. If more than one ced UEC IE is received, only the first one is stored or tandemed with the call. If more than one cdpd UEC IE is received, only the first one is stored or tandemed with the call.

Use with collect digits commands

When a collect ced digits or collect cdpd digits step is processed, Communication Manager retrieves the ced or cdpd and places the digits in the collected digits buffer. Any digits that were previously in the collected digits buffer, such as dial-ahead digits, are erased. If a TTR was connected to the call from a previous collect digits step, the TTR is disconnected.

Valid digits are 0 through 9, *, and #. If the ced or cdpd contain invalid digits, Communication Manager does not store the UEC IE. When the collect digits step is reached, the collected digits buffer is still cleared and if a TTR is attached, it is still disconnected. A vector event is generated to indicate that no digits were collected.

If no ced or cdpd are received from the network when a collect digits step is processed, the step is not processed. However, the collected digits buffer is still cleared and if a TTR is attached, it is still disconnected.

Use of wildcards

If an asterisk (*) is included in the collected digits, it is treated as a delete character. Only the digits to the right of the asterisk are collected. If a pound sign (#) is included in the collected digits it is treated as a terminating character. Only the pound sign and the digits to the left of it are collected. If a single pound sign is sent, it is placed in the collected digits buffer.

String length

The number of ced or cdpd to collect cannot be specified in the collect digits step. Although ced and cdpb can each contain as much as 30 digits, only 16 digits can be collected and stored. If there are more than 16 digits, a vector event is generated.

Vector commands that use ced and cdpd

The following vector steps can access CINFO ced and cdpd in the collected digits buffer:

- adjunct routing link (digits passed in an event report as collected digits)
- converse-on...passing digits
- goto...if digits...
- goto...if digits in table...
- route-to digits
- route-to number...if digit...

Tip:

You can use the **callr-info** button on the phone to view the ced and cdpd information just like the other collected digits.

Internal transfer to a VDN

When a call is transferred internally to a VDN, the following outcomes can occur:

- If the transfer completes before the call reaches the CINFO conditional, the CINFO value of the originator of the call is used.
- If the transfer completes after the call reaches the CINFO conditional, the CINFO value of the terminal that executes the transfer is used.

Tip:

To ensure that the originator's CINFO is preserved during a transfer, add a filler step such as **`wait with silence`** to the beginning of the vector. In this way, a transfer can be completed before the CINFO conditional is encountered.

Buffer storage considerations

To retrieve both the ced and cdpd for a call, you must use two **`collect digits`** steps. Because the **`collect digits`** command for ced or cdpd clears the collected digits buffer, the ced or cdpd that is collected first must be used before the second set is requested.

CINFO vector example

The following vector example involves a scenario in which an incoming call enters a network enabled for the ICP service. The network Communication Manager requests information from the caller (ced) and from the call center database (cdpd). The digits are conveyed in the call ISDN message to Communication Manager and then made available to **`collect digits`** vector steps. ced and cdpd are both used to determine routing for the call.

```
 1. wait-time 2 secs hearing silence
 2. collect ced digits
 3. goto step 7 if digits = 1
 4. goto step 11 if digits = 2
 5. route-to number 0 with cov n if unconditionally
 6. stop
 7. collect cdpd digits
 8. route-to digits with coverage n
 9. route-to number 0 with cov n if unconditionally
10. stop
11. queue-to split 6 pri m
12. wait-time 10 secs hearing ringback
13. announcement 2564
14. wait-time 20 secs hearing music
15. goto step 13 if unconditionally
16. route-to number 0 with cov n if unconditionally
```

In this vector, step 1 provides a wait-time step if calls are transferred to this vector. Step 2 collects the ced. Steps 3 and 4 branch the call to a different vector step depending on the ced digit. If no ced were received, or if the digit received was not 1 or 2, step 5 routes the call to the attendant. If the ced digit collected was 1, the call routes to a second collect step where cdpd are collected. The vector then routes the call to the cdpd. If the ced digit collected was 2, the call queues to split 6.

CINFO interactions

Adjunct Switch Application Interface (ASAI)

Both CED and CDPD can be passed to an ASAI adjunct as collected digits with the **`adjunct routing link`** command and other event reports. ASAI passes a maximum of 16 digits.

If a TouchTone Receiver (TTR) is connected to a call as a result of ASAI-requested digit collection and the call encounters a collect ced or cdpd step, the TTR is disconnected from the call. In addition, any ASAI-requested digits stored in the collected digit buffer are discarded and no entered digits event report is sent.

ASAI does not distinguish between CINFO digits and user-entered digits that are collected as a result of a **`collect digits`** step. CINFO digits are provided to an ASAI adjunct in the same manner as any other collected digits from a vector.

The call offered to VDN domain event report contains the digits from the most recent collect ced or collect cdpd vector step.

Best Service Routing (BSR)

BSR digits are included with the call if a multisite BSR application routes the call to another Communication Manager.

Call Management System (CMS)

You do not have to enable the **Vectoring (CINFO)** field on the System-Parameters Customer-Options screen for ced or cdpd to be passed to CMS. Any version of the CMS accepts ced or cdpd.

Conference

When a conference is established, CINFO digits are merged into the call record of the conference. However, there is no indication of the party to which the digits were originally associated. For security reasons, the CINFO digits are erased when the first ISDN call drops out of the conference.

Look-Ahead Interflow (LAI)

CINFO digits are included with the call if LAI routes the call to another Communication Manager.

Transfer

If a call is transferred from Communication Manager, CINFO digits are lost. If a call is transferred to an internal extension, CINFO digits are retained.

Important:

If a call is transferred to a VDN, the CINFO digits must not be collected until the transferring party completes the transfer. Therefore, when transfers are likely, include an appropriate **`wait-time`** step before the **`collect`** step.

Chapter 10: Multi-National Calling Party Number prefixes

The Calling Party Number (CPN) that accompanies a call can be used for a variety of purposes, including phone display, routing, billing, and screen pops using ASAI. However, using the CPN to identify the origin of the call can produce ambiguous results. For instance, the country code for Germany and the city code for Padova, Italy, are the same (49), so calls from the two locations will each begin with the same digits. The prefix in a multinational CPN helps differentiate local or national numbers from international numbers, allowing Communication Manager to identify calls as national (NTL) or international (INTL). For example, in the U.S. the national prefix is 1 and the international prefix is 011, whereas in Europe the digits are typically 0 and 00. When the values are administered in Communication Manager, the values are included as part of the CPN and can be displayed on an agent's phone or passed to ASAI so that the call can be handled appropriately.

A feature is available on the System-Parameters Feature-Related screen to optionally pass the CPN prefix to VDNs and vectors as part of the CPN. You can use this option on a system-wide basis, or on an individual VDN screen. Accordingly, international call handling can be improved by prioritizing and routing more intelligently such as by automatically routing calls to agents who speak the language. For example, if a customer from Germany contacts to a call center in Spain, based on the international code of Germany (0049 - international prefix + country code), the call can be routed to a German speaking agent. Also, an international call can be given priority over a national call to save toll charges.

Related links

CPN prefix routing example on page 142

CPN prefix routing example

In the following example Vector Routing table includes the country codes for countries in South America, including the US international CPN prefix.

```
VECTOR ROUTING TABLE
      Number: 10      Name: South America       Sort? n
           1: 01154                                17:
           2: 011591                               18:
           3: 01155                                19:
           4: 01156                                20:
           5: 01157                                21:
           6: 011593                               22:
           7: 011594                               23:
```

```
          8: 011592                                24:
          9: 011595                                25:
         10: 01151                                 26:
         11: 011598                                27:
         12: 01158                                 28:
         13: 011597                                29:
         14: 011500                                30:
         15:                                       31:
         16:                                       32:
```

The following vector example shows how calls are routed based on the administered international CPN prefix, country codes, and the information provided in the Vector Routing Table shown above.

```
1. wait-time 0 seconds hearing ringback
2. goto step 11 if ani = none
3. goto vector 1910 @step 1 if ani = 01149+
4. goto vector 1912 @step 1 if ani in table 10
5. goto vector 1915 @step 1 if ani = 01152+
6. goto vector 1920 @step 1 if ani = 011+
7. queue-to split 16 pri m
8. wait-time 60 seconds hearing 32567 then music
9. announcement 32456
10. goto step 8 if unconditionally
11. route-to number 0 with cov n if unconditionally
12. busy
```

In the above example:

- Step 2 checks for calls that do not have an associated ANI, and routes such calls to an operator.
- Step 3 routes calls from Germany to a vector 1910.
- Step 4 routes calls from South American countries, where the country codes are listed in the Vector Routing Table 10, to vector 1912.
- Step 5 routes calls from Mexico to vector 1915.
- Step 6 routes all other international calls to vector 1920. Calls that are not rerouted by the previous steps are then queued.

Related links

Multi-National Calling Party Number prefixes on page 142

Chapter 11: How to create and edit call vectors

You can create vectors by any of the following means:

- Basic screen administration on the System Administration Terminal (SAT), Avaya Site Administration (ASA) or Integrated System Management
- Avaya Call Management System (CMS) using the ASCII administration interface
- Avaya Visual Vectors

For more information about how to create vectors with Visual Vector, see *Avaya Visual Vectors User Guide*. No instructions are available for using the Avaya CMS ASCII administration interface.

Related links

Call Vector screen basic administration on page 144

Call Vector screen basic administration

A vector is entered online using the basic screen administration on the Call Vector screen. The following is an example of the first page of the screen.

Call Vector screen (Page 1 of 3)

```
change vector 20                                                    Page 1 of 3
                            CALL VECTOR
 Number: 20                 Name:_______________________
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? n   ANI/II-Digits? n   ASAI Routing? n
 Prompting? n   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? y      Holidays? y
 01 ________________
 02 ________________
 03 ________________
 04 ________________
 05 ________________
 06 ________________
 07 ________________
 08 ________________
 09 ________________
 10 ________________
 11 ________________
```

The following procedure summarizes how to enter a vector online using basic screen administration:

Procedure

1. At the command prompt, type `change vector xxx`, where *xxx* is the number of the vector.

   Use the `change vector` command to edit an existing vector or to create a new vector. Use the `list vector` command to view all the vectors that are administered for your system.

2. In the **Name** field, assign a name of up to 27 alphanumeric characters.

   The system automatically assigns the vector number, which appears next to the **Number** field.

3. In the **Multimedia** field, determine whether the vector must receive early answer treatment for multimedia calls. This field is applicable if you use Multimedia Call Handling. If you select **y**, calls are answered at the beginning of vector processing and billing for calls start at the same time.

4. In the **Attendant Vectoring** field, select **y** to use the vector as an attendant vector. This field is applicable if the field option in the **Attendant Vectoring** field on the System-Parameter Customer-Options screen is **y**.

5. In the **Meet-me Conf** field, select **y** to use the vector for the Meet-me Conference feature. This field is applicable if the field option in the **Meet-me Conference** on the System-Parameter Customer-Options screen is **y**.

   Note:

   Do not administer **Attendant Vectoring** and **Meet-me Conference** at the same time for a vector.

6. In the **Lock** field, select **y** if you want to allow the vector to be edited only from the Communication Manager SAT or a terminal emulator.

   If you administer the **Attendant Vectoring** field as **y**, you cannot change the default setting of the **Lock** field. The default is **y**.

   Note:

   Always lock vectors that contain secure information, such as access codes.

7. Look at the other fields and determine which fields to administer as **y**.

   The fields indicate the Call Vectoring features and corresponding commands that you can use. If a field setting is **n**, you cannot use the corresponding feature.

   Note:

   Administer the Call Vectoring features on the System-Parameters Customer-Options screen.

Feature	Description
Basic	You can use the Basic Call Vectoring commands.
EAS	Expert Agent Selection is administered as **y**.

Table continues…

Feature	Description
G3V4 Enhanced	You can use the G3V4 Enhanced Vector Routing commands and features.
ANI/II-Digits	You can use the ANI and II-Digits Vector Routing commands. You must administer the G3V4 Enhanced Vector Routing features before you use ANI/II-Digits Vector Routing commands.
ASAI Routing	You can use the Adjunct Routing command.
Prompting	You can use the Call Prompting commands.
LAI	Look-Ahead Interflow is administered as **y**.
G3V4 Adv Route	You can use the G3V4 Advanced Vector Routing commands.
CINFO	You can collect ced and cdpd digits with the collect digits step.
Best Service Routing	BSR is administered as **y**. The available commands vary depending on whether you use singlesite or multisite BSR.
Holidays	You can create tables for special days such as holidays and promotional days. You can use VDN Time Zone Offset to allow the same tables to apply for different time zones based on the VDN the call is routed to. For information about creating holiday tables and defining holiday vectors, see *Avaya Aura® Call Center Elite Feature Reference*.

8. Enter a maximum of 99 vector commands in the blanks next to the step numbers.

   Note:

   You do not have to type every letter of each command that you enter. If you type just the first few letters of a command and press **Enter** or the **Tab** key, Communication Manager populates the entire command.

9. Press **Enter** to save the vector.

Result

After editing a vector, verify that the vector works as intended. This step is particularly important if you delete a step that is the target of a `goto` step.

Related links

How to create and edit call vectors on page 144

How to view vector variable information

Viewing vector variable information

Procedure

While using the `change vector` command, press **Esc F6**.

Result

After the Edit (i/d/v/c/u) prompt, enter a “v” and the variable to be viewed.

Enter an A-Z or AA-ZZ value.

Example: **v G**

1. Press **Enter**.

   Result: The variable information displays at the bottom of the screen.

```
change vector 1                                                   Page   1 of   3

                                  CALL VECTOR

    Number: 1                Name: ---------------
                                          Meet-me Conf? n           Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing? y
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? y   Holidays? y
 Variables? y   3.0 Enhanced? y
01 goto step   1           if rolling-asa      for skill 1st      =  999
02 goto step   2           if rolling-asa      for skill 1st      =  999
03 goto vector 2  @step 1  if rolling-asa      for vdn   active   =  999
04 goto vector 3  @step 1  if rolling-asa      for vdn   latest   =  999
05 goto step   3           if time-of-day      is mon 09:00 to fri 17:00
06 set          A      = V2     MUL   V5
07 set          digits = V4     DIV   A
08 set          digits = none   ADD   none
09 set          U      = digits CATL  none
10 set          digits = digits CATR  none
11 set          digits = none   SEL   digits

                       Press 'Esc F6' for vector editing
Var G: value type VALUE G L=1 ASGN=[5] VAC=VV1
```

Related links

How to create and edit call vectors on page 144

Variable display fields

The following line is displayed at the bottom of the Call Vector screen after you use the commands to view the vector variable information.

Var *letter*: description type scope [L=length S=start ASGN=current_value VAC=fac]

Name	Description
The following four fields are always displayed.	
Var letter	Displays the A through Z vector variable letter you requested.
Description	Displays the name of the variable. For example, *value* type variable.
Type	Displays the vector variable type. For example, value.

Table continues…

Name	Description
Scope	Displays the defined scope as local L or global G.
The following items are displayed only if applicable for the vector variable type.	
L=length	If length is allowed for this variable type, the field displays the defined maximum digit length for the variable.
S=start	If start is allowed for this variable type, the field displays the defined start digit position for the variable. This field does not display for the value type variable.
ASGN=current_value	If the variable is not local and the assignment is determined during call processing, this field displays the current active or latest assignment for the variable. If there is no current value, ASGN=0 is displayed.
VAC=fac	If the value type variable is defined, this field displays the Variable Access Code, VV1 through VV9.

Related links

How to create and edit call vectors on page 144

Variable display examples

Refer to the values assigned in Table A hen looking at the example outputs in Table B.

Table A							
Variable	**Description**	**Type**	**Scope**	**Length**	**Start**	**Assignment**	**VAC**
A	testing for processing time	vdntime	L				
B	digits for ani testing	collect	G	16	1	123456789012 3456	
C	ASAI announce definition	asaiuui	L	1	3		
D	test with null value	collect	G	1	4		
E	total executed vector steps	stepcnt	L				
G	value type	value	G	1		5	VV1
T	time of day, military time	tod	G			1708	

Table continues…

Table A							
Variable	**Description**	**Type**	**Scope**	**Length**	**Start**	**Assignment**	**VAC**
V	set to active VDN for call	vdn	L			active	
W	day of week, 1=Sunday	dow	G			3	
X	caller ID	ani	L	16	1		
Y	day of year	doy	G			102	
Z	temporary value	collect	L	4	1		

Table B	
For	**Based on the values in Table A, the following text is displayed**
`Edit (i/d/v/c/u): v A`	Var A: testing for processing time VDNTIME L
`Edit (i/d/v/c/u): v B`	Var B: digits for ani testing COLLECT G L=16 S=1 ASGN=[1234567890123456]
`Edit (i/d/v/c/u): v C`	Var C: ASAI announce Definition ASAIUUI L L=1 S=3
`Edit (i/d/v/c/u): v D`	Var D: test with null value COLLECT G L=1 S=4 ASGN=[]
`Edit (i/d/v/c/u): v E`	Var E: total executed vector steps STEPCNT L
`Edit (i/d/v/c/u): v G`	Var G: value type VALUE G L=1 ASGN=[5] VAC=VV1
`Edit (i/d/v/c/u): v T`	Var T: time of day, military time TOD G ASGN=[1708]
`Edit (i/d/v/c/u): v V`	Var V: set to active VDN for call VDN L ASGN=ACTIVE
`Edit (i/d/v/c/u): v W`	Var W: day of week, 1=Sunday DOW G L= S= ASGN=[3]
`Edit (i/d/v/c/u): v X`	Var X: caller ID ANI L L=16 S=1
`Edit (i/d/v/c/u): v Y`	Var Y: day of year DOY G ASGN=[102]
`Edit (i/d/v/c/u): v Z`	Var Z: temporary value COLLECT L L=4 S=1

Related links

How to create and edit call vectors on page 144

Inserting a vector step

Procedure

1. After entering the **`change vector`** command, press **Esc F6**
2. At the command line, type `i` followed by a space and the number of the step that to be added, and press **Enter**.

   For example, to insert a new vector step 3, type `i 3` and press **Enter**. You cannot add a range of vector steps.
3. Type the new vector step.

Result

When a new vector step is inserted, the system automatically renumbers all succeeding steps and renumbers the **goto** step references. Under certain conditions, attempts to renumber **goto** step references results in an ambiguous renumbering situation. In such a case, the step reference is replaced by an asterisk (*). A warning prompts you to resolve the ambiguous references and the cursor automatically moves to the first reference to be resolved. You cannot save a vector with unresolved goto references.

You cannot insert a new vector step if 99 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 99.

Related links

How to create and edit call vectors on page 144

Deleting a vector step

Procedure

1. After entering the **change vector** command, press **Esc F6**
2. At the command line, type `d` followed by a space and the number of the step to be deleted, and press `Enter`.

   You can delete a range of vector steps. For example, to delete steps 2 through 5, type `d 2-5` and press **Enter**.

Result

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

For example, if you delete step 7 when you have a **goto** step 7 reference, 7 is replaced by an asterisk (*).

A warning prompts you to resolve the ambiguous references and the cursor automatically moves to the first reference to be resolved. You cannot save a vector with unresolved goto references.

Related links

How to create and edit call vectors on page 144

Entering a comment out indication to an existing vector step

About this task

The vector editing function “c” capability inserts a pound (#) at the beginning of existing vector commands to comment out the vector steps. Once commented out, vector steps are skipped during normal vector processing.

Blank steps and comment out steps cannot be commented out. The comment out capability has no limitations other than the number of vector steps. That is, you can comment out 99 steps in all 8000 vectors or 198,000 steps.

To comment out an existing step:

Procedure

1. After entering the `change vector` command, press **Esc F6**
2. At the command line, type `c` followed by a space and the number of the step to comment out and press **Enter**.

   You can comment out a range of vector steps. For example, to comment out steps 2 through 5, type **c 2-5** and press **Enter**.

Result

Note:

The comment out indication functions differently from the **#** command. See # command on page 171 for details.

Related links

How to create and edit call vectors on page 144

Removing a comment out indication

About this task

Once the comment out indication is removed from a vector step, the vector step executes during vector processing. Removing a comment out indication from a vector step that contains no comments does not affect vector processing.

To remove a comment out indication:

Procedure

1. After entering the `change vector` command, press **Esc F6**.
2. At the command line, type `u` followed by a space and the number of the step to remove a comment out indication and press **Enter**.

   You can remove comments for a range of vector steps. For example, to remove the comment out indication for steps 2 through 5, type `u 2-5` and press **Enter**.

Related links

How to create and edit call vectors on page 144

How to create and construct a vector

You can begin with a single vector that consists of one step and build on this vector to create new vectors for additional functions. At each step, you will learn a few vector commands and approaches to vector processing. The commands in this tutorial will give you a good idea of how to use Call Vectoring.

Step 1: Queuing a call to the main split

If a call is not immediately answered by an agent or an operator, the call is usually queued until an agent becomes available. A call can be connected to an available agent or queued using the vector shown in the following example. In this example, calls are queued to Split 5.

Queuing call to main split

```
                                                          Page 1 of 1
                            CALL VECTOR
 Number: 27            Name: base                Multimedia? n    Lock? n
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? n   ANI/II-Digits? n   ASAI Routing? n
 Prompting? n   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? y      Holidays? y

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

Agent availability

If an agent is available, the **`queue-to`** split command 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. Once the call is sent to the main split queue, the call remains there until an agent receives the call.

Call priority levels

Each call queued to a split occupies one queue slot in the same split. Calls are queued sequentially according to the assignment of the priority level. In the example vector, note that the assigned priority level is low. The priority level establishes the order of selection for each call that is queued. A call can be assigned one of the following four priority levels: top, high, medium, or low.

Within a given split, calls are sequentially delivered to the agent according to the assigned priority level. Calls with the assigned priority as “top” are delivered to an agent first and calls with the assigned priority as “high” are delivered second.

Step 2: Providing feedback and delay announcement

A call remains in queue until an agent becomes available. In the meantime, the caller hears some feedback indicating that the call is being processed.

The vector in the following example provides a feedback solution. In this example, Announcement 2771 contains the following message: “We are sorry. All our operators are busy at the moment. Please hold.”

Providing feedback and delay announcement

```
                                                          Page 1 of 3
                            CALL VECTOR
   Number: 27                   Name: base       Multimedia? n           Lock? n
Multimedia? n      Attendant Vectoring? n     Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? n    ANI/II-Digits? n    ASAI Routing? n
 Prompting? n   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? y        Holidays? y

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

Using the wait-time command

The **wait-time** command in step 2 provides a maximum delay of 8 hours before vector processing proceeds to the next vector step. You can assign the time parameter can be assigned as follows:

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

In the example vector, the wait time is 10 seconds.

In addition to the delay period, the **wait-time** command provides the caller with a feedback. In the example vector, a ringback is provided. You can provide the following types of feedback with the **wait-time** command: silence, system music, an alternate music or other audio source.

The **wait-time** command in the example vector provides the caller with a maximum of 10 seconds of ringback. The **wait-time** command terminates if an agent answers the call before vector processing completes the command. The delay period ends and the accompanying feedback stops. In the example, if the call is delivered to an agent after four seconds, the caller does not hear the remaining six seconds of ringback.

If the call is not answered by the time the **wait-time** command is completed, vector processing continues.

The **announcement** command consists of a recorded message that is often used to encourage the caller to stay on the phone or to provide information to the caller. If the call is delivered to an agent during the **announcement** command, the announcement is interrupted.

Multiple callers can be connected to an announcement at any time. For more information on announcements, see *Avaya Aura® Communication Manager Feature Description and Implementation*.

Step 3: Repeating delay announcement and feedback

The announcement vector provides feedback to the caller after the call is queued. However, if the announcement is played and the agent does not answer the call soon after the announcement, further feedback or treatment is necessary. One solution is provided in the following Call Vector example:

Repeating delay announcement and feedback

```
                                                          Page 1 of 1
                            CALL VECTOR
 Number: 27            Name: base                Multimedia? n    Lock? n
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? n   ANI/II-Digits? n   ASAI Routing? n
 Prompting? n   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? y      Holidays? y

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

The **wait-time** command in step 4 provides additional feedback in the form of music. If the agent does not answer the call before step 4 completes, the **goto step** command in step 5 is processed.

Conditional branching

The **goto step** command is an example of a Call Vectoring command that passes the vector processing control from the current vector step to either a preceding or succeeding vector step.

With the **goto step** command in vector step 5, you can establish an announcement-wait loop that continues until an agent answers the call. The command makes an unconditional branch to the **announcement** command in step 3. If the agent does not answer the call before the announcement in step 3 completes, control is passed to the **wait-time** command in step 4. If the call is still not answered, control is passed to step 5 where the unconditional branch is made once again to step 3. As a result of the established loop, the caller receives constant feedback.

Step 4: Queuing a call to a backup split

With Call Vectoring, a call can be queued simultaneously to a maximum of three splits improving the overall call response time. Multiple split queuing is especially useful during periods of heavy call traffic.

Queuing call to backup split

```
                                                          Page 1 of 1
                              CALL VECTOR
 Number: 27             Name: base                Multimedia? n    Lock? n
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? n   ANI/II-Digits? n   ASAI Routing? n
 Prompting? n   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? y      Holidays? y

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

The **queue-to** split command in step 1 queues the call to the main split. If the agent does not answer the call before the **wait-time** command in step 4 completes, the **check** split command in step 5 attempts to queue the call to a backup split 7 at a medium priority. The condition expressed in the command (**if calls-queued < 5**) determines whether or not the call queues to the backup split. 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.

Conditions used with the check split command

The “calls-queued” condition is one of several conditions that can be included in the **check** split command. Following are the other conditions: unconditionally, average speed of answer (rolling-asa), available agents, staffed agents, expected wait time and oldest call waiting. As is true for the **queue-to** split command, the **check** split command can queue a call at one of the following four priorities: low, medium, high, or top.

Elevating call priority

If the call is queued to split 7, the call priority changes from low to medium, provided 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 as preference is given to callers who have been holding the line for a long time.

Step 5: Limiting the queue capacity

Starting with Communication Manager 2.1, hunt group queue slots are allocated dynamically. Therefore, there is no need to include vector steps to ensure that pre-allocated queue slots in a hunt group have not been exhausted. However, the same approach you used to determine queue slot exhaustion in releases previous to 2.1 can be used to limit the number of calls that are put into queue. The existing vector steps that checked for exhaustion also serve as queue-limiting vectors as is, or modified to limit a different number of calls. The following vector example describes provisions for limiting the number of calls that queue to a split/skill or a hunt group.

Limiting number of queued calls

```
                                                          Page 1 of 1
                          CALL VECTOR
 Number: 27            Name: base                Multimedia? n    Lock? n
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? n   ANI/II-Digits? n   ASAI Routing? n
 Prompting? n   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? y      Holidays? y

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

A check of split 5 is implemented by the **`goto step`** command in step 1. The **`goto step`** command tests if the split contains more than 20 calls using the condition **`if calls-queued in split 5 pri l > 20`**. If this test is successful, control is passed to the **`busy`** command in vector step 10. The **`busy`** command provides a busy signal to the caller and eventually causes the call to drop.

Alternately, if 20 or less medium priority calls are already 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.

Redirecting calls to a backup split

You can queue calls to a backup split instead of playing a busy tone if the queue-to split step cannot queue the call. To queue the call to another split, change the step parameter for the **`goto step`** command from 10 to 6 so that the command reads **`goto step 6.....`**. In this case, control is passed from step 1 to the **`check`** split step 6. Because the queuing step is included within a continuous loop of steps (steps 6 through 9), attempts are made to queue the call.

Step 6: Checking for non business hours

If a caller contacts the call center during non business hours, you can play an announcement requesting the caller to call during working hours. The following example illustrates how to address the situation. The vector is used for a company that is open 7 days a week, from 8:00 a.m. to 5:00 p.m.

Checking for non business hours

```
                                                        Page 1 of 2
                            CALL VECTOR
 Number: 27            Name: base                Multimedia? n    Lock? n
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? n           Lock? y
     Basic? y   EAS? n   G3V4 Enhanced? n   ANI/II-Digits? n   ASAI Routing? n
 Prompting? n   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? y      Holidays? y

 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 seconds hearing ringback
 05 announcement 2771
 06 wait-time 10 seconds hearing music
 07 check split 7 pri m if calls-queued < 5
 08 wait-time 60 seconds hearing music
 09 announcement 2881
 10 goto step 6 if unconditionally
 11 busy
 12 disconnect after announcement 3222
```

The **`goto step`** command in step 1 checks if the call arrives during non business hours. 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 an announcement that requests the caller to call back during working hours. The command then disconnects the call.

If the call does not arrive during the non business hours, control is passed to step 2 and vector processing continues. On step 2, split 5 is checked for calls waiting at all priority levels.

Note:

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.

About duplicate VDNs

You can use the **`duplicate vdn`** command to create duplicate VDNs from an existing VDN. With this functionality, you can configure one VDN as a template that can be reused when creating similar VDNs.

Creating duplicate VDNs

Procedure

1. Enter the following command: `duplicate vdn master_ext [start nnnn] [count xx]`.

   For example, `duplicate vdn 2010 start 3001 count 8`. [start nnnn] is an optional parameter that you can use to specify the number from which to start the number sequence for the duplicate VDN. If you do not specify a start number, the first available VDN after the master VDN is selected automatically. Only one VDN is selected. [count xx] is also an optional parameter. If you do not specify a start number, the first available VDN after the master VDN is selected automatically. Only one VDN is selected.

2. Enter a new name for each duplicate VDN on the Vector Directory Number screen.
3. Edit the duplicate VDNs.

About duplicate vectors

You can use the **`duplicate vector`** command to create duplicate vectors from an existing vector and edit the duplicate vectors to create vectors that are similar to the existing vector. You can use this functionality to configure one vector as a template that can be reused when creating similar vectors.

Creating duplicate vectors

Procedure

1. Enter the following command: `duplicate vector master_vector [start nnnn] [count xx]`.

   [start nnnn] is an optional parameter that you can use to specify the number from which to start the number sequence for the duplicate vector. If you do not specify a start number, the first available vector after the master vector number is selected. Only one vector is selected. [count xx] is also an optional parameter. If you do not specify a start number, the first available vector after the master vector is selected automatically. Only one vector is selected.

2. Enter a new name to each duplicate vector on the Duplicate Vector screen.
3. Edit the duplicate vectors.

Result

The reporting adjunct can access the vectors as normal.

Removing calls from queues

Procedure

Use the **`route-to number xxx`** command, where *xxx* is a VDN extension, as the last step in a call vector.

When you use that command as the last step, Communication Manager removes the call from the queue before call processing continues to subsequent vectors through a route-to VDN command.

When Communication Manager routes the call to the another VDN, Communication Manager:

a. Ends vector processing of the previous vector.

b. Removes the call from any split queue or skill queue that the previous vector processing step establishes for the call.

c. Continues call processing in the vector assigned to the routed-to VDN.

Chapter 12: Vector management

3.0 Enhanced Vectoring requirements

Screen title	Hardware
System-Parameters Customer-Options	No special hardware is required for 3.0 Enhanced Vectoring.

Adjunct Routing requirements

Screen title	Hardware

• Hunt Group
• Class of Restriction (for Direct Agent calls)
• Call Vector
• Station
• Station (ISDN-BRI-ASAI)
	`ISDN-BRI Connection`
	A TN556 ISDN-BRI circuit pack and a TN778 packet control must be in place. The TN778 packet provides packet bus control. An adjunct or a host processor must be in place to receive the request and to select the route. You can use a TN2198 two-wire BRI port circuit pack in place of the TN556 circuit pack. In this case, an NT1 is also required.
	`Packet Bus`
	You must set **Packet Bus** to `y`, which is G3r only, on the Maintenance-Related System Parameters screen before you administer the associated ISDN-BRI screens and fields.

Advanced Vector Routing requirements

Screen title	Hardware
• Vector Directory Number • Hunt Group • Call Vector	Requires no hardware other than that required for Basic Call Vectoring.

ANI/II-Digits requirements

Screen title	Hardware
• Vector Directory Number • Hunt Group • Call Vector • Trunk Group • Vector Routing Tables	Requires no hardware other than that required for Basic Call Vectoring.

Basic Call Vectoring requirements

Screen title	Hardware
• Vector Directory Number • Hunt Group • Call Vector • Feature-Related System Parameters	Announcement capabilities require one of the following: • TN2501AP Integrated Announcement circuit packs. • Avaya Aura® Media Server hosted announcement. • H.248 Media Gateway VAL source for announcements. • External announcement facility, that is, analog announcements. Each analog announcement requires a port on an analog line circuit pack or on an auxiliary trunk circuit pack. For more information, see *Avaya Aura® Communication Manager Hardware Description and Reference*.

Call Prompting requirements

Screen title	Hardware
• Vector Directory Number • Hunt Group • Call Vector	Announcement capabilities require one of the following: • TN2501AP Integrated Announcement circuit packs. • Avaya Aura® Media Server hosted announcement. • H.248 Media Gateway VAL source for announcements. • External announcement facility, that is, analog announcements. Also, each analog announcement requires a port on an analog line circuit pack. For more information, see *Avaya Aura® Communication Manager Hardware Description and Reference*.

CINFO requirements

Screen title	Hardware
• Vector Directory Number • Hunt Group • Call Vector • Trunk Group • Vector Routing Tables	Requires no hardware other than that required for Basic Call Vectoring.

G3V4 Enhanced Vectoring requirements

Screen title	Hardware
• Vector Directory Number • Hunt Group • Call Vector	Requires no hardware other than that required for Basic Call Vectoring.

Holiday Vectoring requirements

Screen title	Hardware
• Holiday Table • Call Vector • Vector Directory Number	No special hardware required for Holiday Vectoring.

Look-Ahead Interflow requirements

Screen title	Hardware
• Trunk Group • CPN Prefix Table	Existing hardware can be used for Look-Ahead Interflow (LAI) connectivity to the receiving Communication Manager. Interconnecting facilities must be ISDN-PRI or SIP with no interworking, that is, call connections that use both ISDN-PRI and non-ISDN-PRI facilities to complete, for the full capabilities of the feature to be operational. LAI calls that interwork can interflow successfully

Screen title	Hardware
	LAI calls can connect ISDN-PRI switch-to-switch using private, public, or ISDN facilities.

Network Call Redirection requirements

Screen title	Hardware
• BSR • Trunk Group • Signaling • DS1	No special hardware is required for Network Call Redirection (NCR).

Variables in Vectors requirements

Screen title	Hardware
• Vector Directory Number • Variables for Vectors Table	No special hardware is required for Variables in Vectors (VIVs).

VDN variables requirements

Screen title	Hardware
Vector Directory Number	No special hardware is required for VDN variables.

Vectoring (Best Service Routing) requirements

Screen title	Hardware
Singlesite Best Service Routing (BSR)	
• Vector Directory Number • Call Vector	No special hardware required for singlesite BSR.
Multisite BSR	

Table continues…

Screen title	Hardware
• Best Service Routing Application Plan • Vector Directory Number • Call Vector • Trunk	Multisite BSR requires no special hardware other than ISDN BRI/PRI or SIP connectivity between switches.

Administering Vector Disconnect Timer

Call Vectoring provides a vector disconnect timer, which can be set for a time between 1 and 240 minutes, inclusive. Set **Vector Disconnect Timer** on the Feature-Related System-Parameters screen to `y`. The timer starts when vector processing starts. Once the timer runs out, the call is dropped. The timer stops when vector processing terminates.

When you set the field to `y`, queued calls that have are unanswered within the administered time are dropped.

Changing and testing a vector

About this task

Do not change the vectors that are currently being used to process calls as changes can have immediate and uncertain effects on call treatment. Instead, write a new vector.

When testing vectors, do not test the entire vector at once, but first divide the vector into portions and test each portion until the entire vector is tested.

After you test the new vector, change the VDN to point to the new vector.

The following guidelines serve as a general procedure to change and test vectors:

Procedure

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.

   Add and test vector commands one command at a time, starting with the first command. Ensure that each line is correct before proceeding to the next one.
4. Test the new vector with the new VDN.

   This ensures that the new vector functions 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 being currently processed by the old vector 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.

Identifying links to a vector

More than one VDN always point to a vector. In addition, some vectors are linked to other vectors by the **`goto vector`** command or by the **`route-to`** command that points to a VDN. Before you delete or change a vector, identify all the VDNs and vectors that will be affected.

The `list usage vector xxx` command displays all the VDNs and vectors that send calls to vector "xxx", where "xxx " is the assigned vector number.

For example, to delete vector 3, enter `list usage vector 3` and press **Enter** to determine what other elements of the system send calls to vector 3.

The List Usage Report screen appears as follows:

```
list usage vector 3                                                Page 1

                         LIST USAGE REPORT

Used By
Vector                Vector Number         1                 Step 3
VDN                   VDN Number            58883
```

VDN 58883 points to vector 3. In addition, step 3 in vector 1 sends calls to vector 3. When you delete vector 3, change the vector and VDN to point to a different vector, or delete the vector and VDN.

Finding all occurrences of a digit string

About this task

A single extension or an external phone number can be used for several elements in a complex vectoring system. When you modify VDNs or vectors, or when you change the phone numbers used in the system elements using commands, such as **`route-to`** or **`best service routing`**, Communication Manager allows you to find a specific digit string.

Procedure

1. The **`list usage digit-string (1-16 digits)`** command finds the specified digit string in vectors, vector routing tables, and BSR plans.

The digit string can contain numerals from 0-9 and characters, such as *, #, ~, p, w, W, m, and s.

For example, to find the system elements that route calls to VDN 53338:

2. Type `list usage digit-string 53338` and press `Enter`.

   The system displays the List Usage Report screen.

```
list usage digit-string 53338                                    Page 1

                        LIST USAGE REPORT

Used By
Vector               Vector Number         1                 Step  3
Vector               Vector Number         5                 Step  8
Vector               Vector Number         18                Step  4
Vector               Vector Number         37                Step 10
Best Service Routing Plan Number           1             Location  1
Best Service Routing Plan Number           2             Location  3
Best Service Routing Plan Number           5             Location  1
```

   Three BSR plans and steps in four different vectors route calls to this VDN. If you delete this VDN or assign a different extension, you will have to update the extension used by these system elements.

Chapter 13: Call Vectoring commands

About Avaya Call Center packages

The features required to use vector commands are included in the following Call Center packages:

- Call Center Introductory: Provides the Automatic Call Distribution (ACD) capabilities for a small, non skill-based call center with up to 50 agents.
- Call Center Elite: Includes Expert Agent Selection (EAS) and advanced Call Vectoring capabilities.
- Avaya Aura® Call Center Elite: Includes Business Advocate as an entitlement.

Note:

Call Center Deluxe was available for software releases prior to Communication Manager 2.0. An Introductory package, identical to Call Center Deluxe without Best Service Routing (BSR), was also available. The Introductory package is now included in the Communication Manager 6.0 or later.

Most features required to enable vector commands are included in the basic Communication Manager package. To use the skill options associated with some vector commands, you must activate the EAS feature. EAS is included in the Call Center Elite package.

Additionally, other vector commands require Avaya Virtual Routing, the marketing name for what is referred to as MultiSite Best Service Routing (BSR), which activates the Look-Ahead Interflow (LAI) feature. Commands, such as Auto Attendant which activates Call Prompting, are available with non-Call Center Right-To-Use (RTU) offerings.

Communication Manager options required to enable vector commands

The following table lists the options required to enable various vector commands, options, and parameters. (“Basic” refers to the call vectoring basic options)

Command	Basic	Prompting	Attendant	Other required options
`adjunct routing link`	x			ASAI

Table continues…

Command	Basic	Prompting	Attendant	Other required options
announcement	x	x		
busy	x			
check best	x			ACD, G3V4 Advanced Routing, Best Service Routing (BSR)
check split/skill if <condition>	x			ACD
check split/skill if rolling-asa	x			ACD, G3V4 Enhanced, G3V4 Advanced Routing
check split/skill if expected-wait	x			ACD, G3V4 Enhanced, G3V4 Advanced Routing
check best if expected-wait	x			ACD, G3V4 Enhanced, G3V4 Advanced Routing; BSR
check split/skill if oldest-call-wait pri	x			ACD, G3V4 Enhanced
check split/skill/ best if wait-improved	x			ACD, G3V4 Advanced Routing, Best Service Routing
check skill if available-agents all-levels	x			EAS active
check skill if available-agents pref-level	x			EAS active
check skill if available-agents pref-range	x			EAS active
collect digits		x		
collect ced/cdpd digits		x		Vectoring (CINFO)
consider location	x			ACD, G3V4 Advanced Routing, Best Service Routing, Look-Ahead Interflow LAI is provided with virtual routing Right To Use (RTU)
consider split/skill	x			ACD, G3V4 Advanced Routing, Best Service Routing
converse-on split/ skill	x			
converse-on split/ skill passing wait	x			ACD, G3V4 Enhanced, G3V4 Advanced Routing
disconnect	x		x	

Table continues…

Command	Basic	Prompting	Attendant	Other required options
disconnect after announcement <extension>	x		x	
goto step/vector if unconditionally	x	x		
goto step/vector if <condition> in split/skill	x			ACD
goto step/vector if digits		x		
goto step/vector if time-of-day	x			
goto step/vector if oldest-call-wait pri	x			ACD, G3V4 Enhanced
goto step/vector if rolling-asa	x			ACD, G3V4 Enhanced, G3V4 Advanced Routing
goto step/vector if expected-wait	x			ACD, G3V4 Enhanced, G3V4 Advanced Routing
goto step/vector if expected-wait for best	x			ACD, G3V4 Enhanced, G3V4 Advanced Routing, Best Service Routing
goto step/vector if counted-calls	x			G3V4 Enhanced, G3V4 Advanced Routing
goto step/vector if ani	x			G3V4 Enhanced, G3V4 ANI/II-Digits Routing
goto step/vector if ii-digits	x			G3V4 Enhanced, G3V4 ANI/II-Digits Routing
goto step/vector if wait-improved	x			ACD, G3V4 Advanced Routing, BSR
goto step/vector if interflow-qpos	x			ACD; Look-Ahead Interflow (LAI)
goto step/vector if queue fail			x	
goto step/vector if holiday in/not-in table	x		x	Holiday Vectoring
messaging split/skill	x	x		
messaging split/skill active/latest If G3V4 software has not been purchased, these	x	x		

Table continues…

Command	Basic	Prompting	Attendant	Other required options
commands require the G3V4 maintenance load				
queue-to best	x			ACD, G3V4 Advanced Routing, Best Service Routing
queue-to split/skill	x			ACD
queue-to attd-group				Attendant Vectoring
queue-to attendant				Attendant Vectoring
queue-to hunt group				Attendant Vectoring
reply-best	x			ACD, G3V4 Advanced Routing, Best Service Routing, Look-Ahead Interflow
return	x			Vectoring (3.0 Enhanced)
route-to number	x			
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		
route-to number if interflow-qpos	x			ACD, Look-Ahead Interflow
set	x			Vectoring (3.0 Enhanced)
stop	x	x		
wait-time <time>	x	x	x	
wait-time <time> hearing <treatment>	x	x	x	
wait-time <time> hearing <extn> then <treatment2>	x	x	x	

Vector command description

The following table provides a brief description of each Communication Manager call vectoring command.

Command	Description
#	To add comments to vectors.
adjunct routing link	To request an adjunct to route a call.
announcement	To connect the caller to a recording.
busy	To connect the caller to a busy tone.
check	To connect or queue calls based on conditions.
collect digits	To prompt the caller for digits.
consider	To retrieve the BSR status data from a local split, skill, or a remote location.
converse-on	To deliver a call to a converse split or skill and to activate a Voice Response Unit (VRU).
disconnect	To force the call to disconnect after playing an announcement.
goto step and goto vector	To cause an unconditional or a conditional branch to another step in the vector.
messaging	To prompt the caller to leave a message at a specified extension.
queue-to	To connect or queue a call to: • The primary split or skill. • The attendant, the attendant group, or the hunt group with Attendant Vectoring. • The best resource found by the **consider** command series.
reply-best	To send the BSR status data to the primary vector. You can use the **reply-best** command only in a multisite application.
return	To return vector processing to the step following the **goto** command after a subroutine call processing.
route-to	To connect a call to the entered destination using the **collect digits** command, to an internal destination, or to an external destination.
set	To perform arithmetic and string operations during vector processing and assign values to a vector variable or to the digits buffer.
stop	To stop vector processing.
wait-time	To initiate feedback to the caller and to delay processing of the next vector step.

command

Purpose

The comment (**#**) command adds comment steps to vectors. The steps are skipped without being analyzed and vector processing continues with the next step. You can include comments within vectors for ease of maintenance and troubleshooting.

Note:

The comment (#) command functions differently from the comment out option of the Esc F6 vector editing function.

Syntax and valid entries

#	A comment command that adds a note with up to 71 characters.
	A comment out command that tells a vector step to ignore processing. Use the edit function, <esc> f6 to insert the command.

Considerations

- Each # command line uses a vector step.
- You can enter up to 71 characters of text after the # character.
- You can have as many blank # commands as the number of available vector steps.
- There is no limit on the number of # command comment steps in a vector. However, the total number of non-blank comment steps allowed per system is limited as follows:
  - Avaya S8300 or Avaya S8400 server: 1,280 steps in 256 vectors.
  - Avaya S8800 or Avaya Common server: 40,000 steps in 8,000 vectors.

  For more information, see *Avaya Aura® Communication Manager System Capacities Table*.
- A single comment (#) command is counted as one step.
- More than two consecutive comment (#) commands are counted as one step.
- The comment (#) command steps are not counted toward the 10,000 executed step limit or by the *stepcnt* vector variable.
- Call Management System R14 or later supports the comment out capability, that is, adding a comment (#) after the step number. Attempts to use the comment out capability with prior releases of Call Management System result in the `unsupported step` type error.

The following sample System Capacities screen shows the used, available, and system limit values for non-blank comment (#) commands.

Sample System Capacities screen

```
display capacity                                                  Page   3 of  13
                                  SYSTEM CAPACITY
                                                                        System
                                                      Used Available  Limit
                                                     -----------------------
                   CALL COVERAGE
                           Coverage Answer Groups:      0    1000     1000
                                   Coverage Paths:      5    9994     9999
                               Call Pickup Groups:      1    4999     5000
                                     Call Records:      -       -    15424

          CALL VECTORING/CALL PROMPTING
                   Total Vector Directory Numbers:     81   19919    20000
               Meet-me Conference VDNs per system:      1    1799     1800
   Maximum Number of Expanded Meet-me Conf. Ports:      0     300      300
                         Total Vectors Per System:     90    1910     2000
            Meet-me Conference vectors per system:      1     999      999
        BSR Application-Location Pairs Per System:      8    2552     2560
                         Background BSR Poll VDNs:      0       5        5
```

```
                Vector Comment Steps (non-blank):     77    9923    10000
                           Policy Routing Tables:      1    1999     2000
                           Policy Routing Points:      1    5999     6000
```

Operation

You can create the comment (**#**) command vector steps by typing a pound (#) character in the command field of a blank vector step. You can enter up to 71 characters as the text parameter to the comment (**#**) command. Any combination of alphanumeric visible ASCII characters, including blanks, is valid.

adjunct routing link command

Purpose

The **`adjunct routing link`** command causes a message to be sent to an adjunct requesting routing instructions.

Syntax and valid entries

`adjunct routing link`	1-64 - CTI Link ID [A-Z, AA-ZZ] V1-V9
Link capacity is based on your release and configuration. For more information, see *Avaya Aura® Communication Manager System Capacities Table*.	

Requirements

The following are the requirements for using the **`adjunct routing link`** command:

- You must install the Adjunct Switch Application Interface (ASAI) software.
- You must have an Application Enablement Services (AES) port and connect the port to an ASAI host.
- You must use a valid assigned link number. If the value determined during call processing is not valid, the adjunct route step is skipped, and a vector event is logged.

Note:

Do not reassign or change the link number administration assignments during system operation.

The adjunct routing link process

The **`adjunct routing link`** command provides a means for an adjunct ASAI processor to specify the destination of a call. Communication Manager provides information in an ASAI route request message. The ASAI adjunct uses the message to access a database and determines a route for the call. In a typical application, the ASAI adjunct can use the dialed number, CPN/BN, or digits collected using Call Prompting or CINFO to access customer information. The adjunct then

determines the call route. A maximum of 16 digits collected from the last **`collect digits`** command can be passed.

An adjunct specified in an **`adjunct routing link`** command can route a call to an internal number, an external number, a split, a VDN, an announcement extension, or a specific agent. An adjunct can also provide priority ringing, priority queuing and specify that a route to an agent be done as a DAC.

When a call encounters an **`adjunct routing link`** command, Communication Manager sends an ASAI message to the specified adjunct requesting a call route. The following list identifies the contents of the message:

- Calling number information: The ISDN-PRI or R2-MFC signaling facilities provide the CPN/BN. If the call originates from a local Communication Manager extension, the extension is the calling number.
- Originating line information (II-digits): The ISDN-PRI facilities provide a two-digit code to indicate the type of originating line.
- Called number: Is the originally called extension if a call is forwarded to a VDN, or the first VDN through which the call is routed if the call is not forwarded to the VDN.
- Routing VDN: Last VDN that routed the call to the vector that contains the **`adjunct routing link`** command.
- Call identifier: An ASAI identifier that permits the ASAI adjunct to track multiple calls using either event notification or third party call control.
- Look-Ahead Interflow (LAI) information (if any): Includes the original VDN display information, the priority level of the call at the originating Communication Manager, and the time that the call entered vector processing.
- Digits collected using Call Prompting (if any): Digits are collected by the most recent **`collect digits`** command. The digits can be CINFO digits, but if so, ASAI does not indicate the type of digits collected.
- User-to-User Information (if any): Is the ASAI user-provided data associated with the call. If provided by ASAI, the data is provided in a 3rd-Party-Make-Call, Auto-Dial, or Route-Select message. If provided over ISDN, the data is in the SETUP message that delivers the call to Communication Manager.

The **`wait-time hearing i-silent`** command is used to allow the adjunct to decide whether to accept an incoming ISDN-PRI call. When this step is encountered after an **`adjunct routing link`** step, Communication Manager does not return an ISDN PROGress message to the originating Communication Manager. This is particularly important for Network ISDN features and for the LAI feature.

If the call is queued, the **`adjunct routing link`** step is ignored and vector processing continues with the next vector step. If the ASAI link specified in the **`adjunct routing link`** step is down, the step is skipped.

An ASAI link failure can change the manner in which the subsequent treatment, that is, announcement or wait-time steps (if any) in the vector, are processed. In some instances, such processing is influenced by the position that the treatment steps occupy in the vector. In other

instances, the positioning of the commands along with their relationship to specific **goto** commands come to effect. For example, any **announcement** or **wait-time** step that immediately follows an **adjunct routing link** step, with its ASAI link down, is skipped.

The second step after the **adjunct routing link** step is often implemented as a default treatment, for example, a route-to an attendant. If the ASAI link is down, the default step executes immediately. Otherwise, the step executes only if the application does not respond with a route within the time period specified by the **wait-time** step.

On the other hand, if a **goto** step follows an **adjunct routing link** step, Communication Manager executes the **goto** step and skips various treatment steps according to their position in the vector and the conditional determination of the **goto** step. Specifically, if the **goto** step succeeds and the branch is taken, Communication Manager skips the **announcement** or **wait-time** step, which is the first non-goto step branched by the **goto** step.

Note:

The **goto** step is designed to branch to a non treatment step, which contains a command other than a **wait-time** or an **announcement** command.

Alternately, if the **goto** step fails and the branch is not taken, Communication Manager skips the **announcement** or **wait-time** step that immediately follows the **goto** step, if the application is down.

Note:

The **goto** step that fails can be at the end of a sequence of **goto** steps that branch to each other.

After Communication Manager sends a route request to the ASAI adjunct, vector processing continues with the vector steps that follow.

The step that follows the **adjunct routing link** step, in effect, determines the maximum length of time Communication Manager waits for the ASAI adjunct to reply with a call route. Accordingly, you must always include either a **wait-time** or an **announcement** step immediately after an **adjunct routing link** step. Moreover, Communication Manager cancels the route request if vector processing encounters a step containing any of the following commands:

- **busy**
- **check split**
- **collect digits**
- **converse-on split**
- **disconnect**
- **messaging split**
- **queue-to split**
- **route-to**

Multiple adjunct routing steps can follow each other in sequence. Each step activates a separate adjunct route request. Any intervening vector command (or blank step) between two **`adjunct routing link`** commands cancels any previous route-to requests.

If the server receives a valid call route before one of the vector commands in the previous list is executed, the server uses a route-select message to route the call to the destination specified by the adjunct route. If the call route received in invalid, the route request is terminated without affecting vector processing.

The adjunct can also decide to not route a call by rejecting the route request sent by the server, or the link/application can go down. Upon receiving a route request rejection, or detection of a link/application failure, the server terminates the **`announcement`** or **`wait-time`** step that is being executed for the call and continues with the next vector step.

When the server receives a call route from the ASAI adjunct, the server first validates the route as follows:

1. The server verifies if the VDN's Class of Restriction (COR) permits the call to be terminated at the adjunct-supplied destination.
2. The server verifies if the adjunct-supplied information, that is, the destination number, ACD split, and Trunk Access Code (TAC) / Alternate Automatic Routing (AAR) / Automatic Route Selection (ARS) access code, for the route is valid. This includes checking if the destination is compatible with the dial plan and that the options specified by the adjunct are correct.
3. If the ASAI adjunct specifies the Direct Agent Call (DAC) option, the destination number, that is, the agent must be logged in to the adjunct-specified ACD split.
4. If the destination for the call is external, the server verifies if the trunk is available for the call.

If any of the conditions are not met, the route validation fails and the server performs the following actions:

1. Discards the route.
2. Notifies the ASAI adjunct that the route is invalid.
3. Continues with vector processing.

If the route is valid, the server performs the following actions:

1. Terminates vector processing immediately.
2. Notifies the ASAI adjunct that the route is accepted.
3. Routes the call to the destination specified by the ASAI adjunct.

When the call is routed, the caller hears the normal call progress tones and feedback. However, if the call is routed to an extension with no available call appearances and no coverage path, the caller hears a busy tone. Any other features such as Send-All-Calls or Call Forwarding, that can be in effect at the adjunct-supplied destination interact with the routed call.

Note:

The operation described is similar to that for the **`route-to with coverage`** set `y` commands.

The **`adjunct routing link`** command has no interaction with answer supervision.

If adjunct routing is used with ISDN-PRI, an **adjunct routing link** command followed by a wait-time hearing silence step signals the originating server that the receiving server has accepted the call, even though answer supervision has not been provided. To prevent this from occurring, use the **wait-time hearing i-silent** command after the **adjunct routing link** step.

adjunct routing link command feature interactions

For a call made directly to a VDN, the command is treated like a **route-to** command that has the with coverage parameter set to y.

Note:

If the Display VDN for Route-to DAC field is enabled for the VDN, the name of the VDN is displayed at the agent station for a call that is routed through an adjunct.

For a call that is covered to a VDN, the command is treated like a **route-to with coverage = n** command. A covered call that is routed by an **adjunct routing link** command to a destination that has the Call Forwarding feature activated is not further redirected, since the call has already been redirected by coverage.

For LAI or Network ISDN features, the **adjunct routing link** command is treated as a neutral vector command in all cases. However, the command is usually followed by an **announcement** or **wait-time** command, each of which is a call acceptance command. The G3V4 **wait-time hearing i-silent** command can be used when a neutral **wait-time** command is required to allow the adjunct to accept or reject the call.

If an **announcement** command follows a failed **adjunct routing link** command, the announcement is interrupted. If the **adjunct routing link** command succeeds, that is, the server receives a destination from the ASAI adjunct, the announcement terminates immediately.

If an ASAI adjunct has supplied dial-ahead digits for a **collect digits** step and the vector processes a **collect ced digits** or **collect cdpd digits** step, the ASAI supplied dial-ahead digits are discarded with no notification to the adjunct.

If a Touch-Tone Receiver (TTR) is connected to a call because an ASAI adjunct has requested digit collection and the vector processes a **collect ced digits** or **collect cdpd digits** step, the TTR is disconnected from the call.

adjunct routing link command interactions with CMS

Adjunct routing attempts are stored in the ADJATTEMPTS database item and are reported as *Adjunct Routing Attempts* in the standard reports. If the call is queued to a split/skill when the **adjunct routing link** command is encountered, the step is skipped and no messages are sent to Call Management System (CMS). The adjunct routing attempts are not reported for such calls.

When the server successfully executes a routing response from the adjunct, the success action is tracked in the ADJROUTED and ADJROUTTIME database items and is shown as *Adjunct Routing Completions* in the standard reports.

Additional tracking of the **`adjunct routing link`** command is based on the destination successfully routed-to and varies as follows.

Routed to station or attendant		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Vector flow out	
INTIME	Average time in vector	
CONNECTCALLS/ CONNECTTIME	Other calls connect	Answered calls on R5

Routed to trunk		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Vector flow out VDN flow out	
INTERFLOWCALLS/ INTERFLOWTIME	VDN flow-interflow	
INTIME	Average time in vector	

Routed to Vector Directory Number (VDN)		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Vector flow out VDN flow out	
INTIME	Average time in vector	
INFLOWCALLS	Vector flow in VDN flow in	New vector new VDN

Routed to split or hunt group		
Database item	**Report heading**	**Notes**
CALLSOFFERRED		New split
LOWCALLS/MEDCALLS		No priority/priority

The standard reports display information about split/skill calls based on the final disposition of the call. A calls that is processed by vectors is vector-directed when the **`adjunct routing link`** command is in a vector. When such a call is answered by an agent, the call is tracked as ACDCALLS/ANSTIME and is reported as ACD calls, split/skill ACD calls, and the Average Speed of Answer (ASA). A call that is abandoned after the command routes the call to a station or an attendant is tracked in the VDN tables as ABNCALLS/ABNTIME.

adjunct routing link command interactions with BCMS

If the command advances a call to another position, that is, the ASAI routing is successful, the call is tracked as an OUTFLOW in the VDN report.

announcement command

Purpose

Use the command to play a recorded announcement for the caller.

Syntax and valid entries

announcement	extension number A-Z, AA-ZZ V1-V9

Requirements

- You must install an integrated board, an aux trunk, or an analog (Tip and Ring or Lineside DS1) announcement equipment.
- You must administer and record an announcement.

Basic operation for the announcement command

The announcement is played from the beginning to the 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 that is provided before an announcement, for example, a wait with music or ringback, continues until the announcement is played.

If the announcement's queue is full, the call retries the announcement step for an indefinite period of time before processing any new vector steps.

An announcement are interrupted, if an **announcement** command follows a failed **adjunct routing link** command. If the **adjunct routing link** command succeeds, that is, the server receives a destination from the ASAI adjunct, the announcement terminates immediately.

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 aux trunk or analog equipment is not attached

For a complete description of the types and operation of announcements, see *Avaya Aura® Communication Manager Feature Description and Implementation*.

announcement command considerations

- After an announcement is provided, reattach an audible feedback such as music.
- Depending on the type of announcement equipment and how the equipment is administered, callers can listen to the entire announcement or can interrupt an announcement. When a call is connected to an announcement, any previous treatment is discontinued.
- When an announcement starts, the caller waits in an announcement queue if the announcement is not ready to play. Callers hear the previously established call treatment until the announcement starts. If the announcement queue is full, vector processing retries the **`announcement`** command indefinitely.

  Important:

  If an integrated announcement board is in use and the requested announcement is not administered or recorded, vector processing skips the **`announcement`** command and continues with the next vector command.
- If the call is in a split or skill queue, the call remains in queue while the announcement plays. If the call is still in queue after the announcement ends, the caller hears silence until another **`announcement`** command, a **`wait hearing ringback`** command, or a **`wait hearing music`** command is processed. If the call connects to a station while the announcement is playing, the announcement stops 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 station.

Delay announcements

The follow example shows a vector step that uses a delay announcement:

Delay announcement

```
announcement 2556 [All our agents are busy. Please hold.]
```

If the caller remains on hold, a supplementary delay announcement similar to the following example can be used.

Follow-up delay announcement

```
announcement 2557 [Thanks for holding. All our agents are still busy. Please hold.]
```

Tip:

A delay announcement is usually coupled with a delay step that is provided by the **`wait-time`** command.

Forced announcements

When heavy call traffic is expected due to a major event, such as a widespread service problem being addressed, a call center can provide a forced announcement. Forced announcements are followed by a **disconnect** command.

The following example shows a forced announcement that can be inserted into a vector to address such situations.

Forced announcement example

```
disconnect after 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.”]
```

Information announcements

In some cases, callers can be provided with an information announcement that addresses the queries without further interaction. An example information announcement is shown below.

Information announcement example

```
disconnect after announcement 2918 [“Today has been declared a snow day. Please report
for work tomorrow at 8 A.M.”]
```

Recording announcements

To create an integrated announcement or music recording that resides in Voice Announcement over LAN (VAL) boards, virtual VAL sources in media gateways, or Avaya Aura® Media Server, use a system telephone, create .wav files using a local computer, or an announcement recorded at a professional recording studio.

Recording announcements using .wav files

Using .wav files for recording provides the best quality, flexibility and reliability.

- Use a computer recording application, such as Microsoft Sound Recorder to create a CCITT m-Law (for U.S.) or A-Law, 8 KHz, 8-bit mono format .wav file.
- Use a file name with up to 27 characters without blanks.
- Transfer the file to the VAL announcement source using FTP. You can also use System Manager. If you are using Avaya Aura® Media Server-based announcement sources, you must use the System Manager announcement manager interface to transfer the files.
- Administer the .wav file name to an announcement extension on the Announcement/Audio Sources screen.

Recording announcements using a telephone

You can record announcements already defined on the Announcement/Audio Sources screen directly to the VAL source assigned to the announcement extension.

- Using a Communication Manager system telephone with a console Class of Service (COS), dial the assigned announcement access Facility Access Codes (FACs).
- For the best quality and functionality, use a DCP or IP phone.
- With a DCP or IP phone, use the “#” button to stop the recording without introducing a click and dropping the recording session. With an analog phone, softly depress the switch hook to end the recording.

  Note:

  You cannot use a telephone to record an announcement with an audio group assignment. For VAL announcements use FTP and for Avaya Aura® Media Server-based announcements use System Manager to move each prerecorded file to each of the sources defined for the audio group.
- Get the best audio quality by using a DCP phone directly connected to the same gateway that contains the VAL source or in the same port network multi-connect grouping.
- Do not use remote or branch phone connections that route over Inter-Gateway Alternate Routing (IGAR)-supported facilities because the beginning portion of the announcement can get clipped and not recorded.

Announcement recording tips for high traffic volume applications

When setting announcement recordings for high traffic volume applications:

- Ensure that the announcement extensions are defined with `queuing` enabled. Set the **Q** field on the Announcements/Audio Sources screen to `y`.
- Use the integrated announcement boards for better performance. The TN2501 Voice Announcement over Local Area Network (VAL) boards have the highest capacity. These boards consist of 31 play ports and 60 minutes of storage using up to 256 announcement files.
- Use short announcements. Long announcements delay further processing and hold up resources for a long time.
- When recording announcements for collecting digits, play the longer part of the announcement using an announcement command. Define the specific instructions for dialing in the announcement for the collect digits command. Minimize the use of variable-digit collection and intermediary announcements. These tips reduce holding up the digit-collection resources.
- Spread heavy use announcements over multiple boards. If announcements for different applications are mixed on the same board, do not mix announcements for applications that have coincident peak periods.
- Use locally-sourced music and announcements to reduce the use of bandwidth and VoIP resources in IP-connected configurations. This feature also provides backup for announcement source failures in all configurations.

  For more information, see *Administering Avaya Aura® Call Center Elite*.

- Use repeating announcements only with the expanded **`wait-time xx secs hearing ann_extn then`** [**`music`**, **`ringback`**, **`silence`** or **`continue`**] command. The ann_extn for the wait step timing puts a limit on how long the call is processed by the vector step.

  Note:

  The use of repeating announcements, such as integ-rep and integ-mus, in the announcement steps or collect digits steps can halt processing of the subsequent vector steps as repeating announcements are non-ending announcements.

Considerations for DTMF transfer and connect applications

- Record the announcement using an analog telephone or a good quality DTMF touch-tone keypad that has a direct electrical connection.
- Do not exceed 6.25 digits per second when generating the DTMF digits that are recorded.
- For DTMF signaling to the network, the Call Center/PBX DTMF generation must send DTMF tones with a minimum of 80 milliseconds (ms) of digit duration and 80 ms of inter-digit silence, and include a pause of a minimum of 350 ms between the *8 and the redirection number. The lower and upper bounds are 300 ms - 11 seconds.
- Minimize the recording of noise or periods of silence.

You cannot record DTMF tones using IP phones or record H.248 Media Gateway Virtual Voice Announcement over LAN (VVAL) announcement sources using any type of telephone. Instead, you must choose any of the following:

- Record DTMF to port network TN2501 VAL boards using an analog telephone connected to the same port network. You can transfer the created wave files to H.248 Media Gateway VVAL or Avaya Aura® Media Server sources.
- Record the DTMF tones using a commercially-available personal computer software tool such as Vox Studio. You can transfer the saved wave file to the desired Media Gateways.

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).

announcement command feature interactions

For Look-Ahead Interflow, the **`announcement`** command is treated as a call acceptance vector command or a neutral vector command.

The command is treated as a call acceptance vector command when one of the following is true:

- The announcement is available.
- The call is queued for an announcement.

- The announcement is retried.

The command is treated as a neutral vector command when the announcement is unavailable.

Announcement command interactions with CMS/BCMS

CMS and BCMS do not track announcement commands.

busy command

Purpose

Use the **busy** command to terminate vector processing after playing a busy signal.

Syntax

busy	Terminates vector processing after playing a busy signal

Operation

The caller hears a busy tone and vector processing terminates. An exception to this operation occurs on Central Office (CO) trunks where answer supervision has not been sent. Callers on such trunks do not hear the busy tone from Communication Manager. Instead, the 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.

Use a busy tone to process a call that arrives at a time when there are a large number of calls queued in the main split or when the call center is out of service or closed.

busy command example

```
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 seconds hearing music
5. stop
6. busy
```

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

Answer supervision considerations for the busy command

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 a 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.

busy command feature interactions

For LAI or BSR, the **busy** command is treated as a call denial vector command in all cases.

busy command interactions with CMS

busy command	
Database item	**Report heading**
BUSYCALLS/BUSYTIME	Calls Forced Busy Calls Busy/Disc
OTHERCALLS/OTHERTIME	Inbound Other Calls
INTIME	Avg Time In Vector

BUSYTIME, OTHERTIME, and INTIME for splits and vectors are tracked according to when the busy tone starts. BUSYTIME, OTHERTIME and INTIME for VDNs are tracked according to when the trunk idles.

busy command interactions with BCMS

BCMS tracks calls that are busied out due to the **busy** command as OTHER in the VDN report.

check command

Purpose

Checks the status of a split or skill to deliver the call to the split or skill.

Syntax and valid entries

Note:

The maximum limit is less on some platforms. Use the help key for your switch administration software to determine the applicable limit for your system.

check	best	if	expected wait	< 1-9999 seconds		
			unconditionally			
			wait improved	> 0-9999 seconds		
	skill	hunt group skills for VDN:1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	if	available-agents > 0-1499	all-levels
						pref-level skill level 1 [Skill levels are 1-16

Table continues…

						(1 is best, 16 is lowest). Skill level 2 must be greater than or equal to skill level 1.]
						pref-range skill level 1 to skill level 2
	skill	hunt group skills for VDN: 1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	if	calls-queued < 1-999 expected-wait < 1-9999 seconds oldest-call-wait < 1-999 seconds rolling-asa <1-999 seconds staffed-agents > 0-1499 wait-improved > 0-9999 seconds unconditionally	
	split	hunt group				
	split	hunt group	pri: l=low, m=medium, h=high, or t=top	if	available-agents > 0-1499	

Operation

Use the **check** command to check the status of a split or skill against the conditions that you specify in the command. If the conditions specified in the command are met, the call is delivered to the split or skill. If the conditions are met, but no agents are available, the call is queued to the split or skill waiting for an agent to become available.

You can use each **check** command with one of the following parameters:

- split
- skill
- best

To use the **check split or skill** command, you must specify the split or skill that you want to check. The **check best** command checks the status of the best split or skill identified by the immediately preceding series of **consider** steps and delivers or queues the call to the split or skill. You do not have to specify the split or skill in **check best** commands as Communication Manager compares many skills and identifies the best in the preceding series of **consider** steps.

You can customize the command to check for or respond to specific conditions. For example, you can specify the **check** command to put calls in a queue or deliver calls unconditionally. You can use the command to put calls in a queue or deliver calls 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 a queue at the specified priority level or higher is waiting less than the threshold value, which is expressed in seconds.

- Rolling Average Speed of Answer (ASA), which is expressed in seconds, is less than the threshold value.
- Expected Wait Time (EWT), which is expressed in seconds, is less than the threshold value.
- EWT is improved by more than the threshold value by queuing the call to the specified split or skill. EWT in the specified split or skill is compared to the current EWT of the call . The EWT of a call is infinite if the call is not in a queue.

A call can be queued simultaneously to up to three splits or skills. A call remains queued until vector processing terminates, using a successful **`disconnect`**, **`busy`**, or **`route-to`** command, or using 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 a split or skill to which the call is queued, the following actions take place:

1. Call begins to ring.
2. Communication Manager removes the call from the queue when an agent answers the call.
3. Vector processing terminates.

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

- Command conditions are not met.
- Queue of the split or skill is full.
- Desired split or skill has no queue and no available agents.
- Desired split or skill is not vector-controlled.
- Call is already queued to the split or skill at the specified priority level.
- Call has been previously queued to three different splits or skills.

A **`route-to`** command to another VDN can be used to remove a queued call from the splits or skills. The steps in the routed-to vector can then be used to queue the call to other splits or skills.

The **`check`** skill command can further have a skill level preference parameter. The skill level preference parameter can be a skill level or a range of skill levels. With the option, you can choose to route calls to an available agent with the specified skill level. Skill level preference is applied under the following conditions:

- Agents with the specified skill must be available, otherwise, the step is skipped and vector processing continues at the next step.
- The administered skill must be of type `EAD-MIA` or `EAD-LOA`.
- If agents with the specified skill level are unavailable, the call is routed to an available agent with the specified skill.

check split command

This 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.

The **`check split`** command is almost identical to the **`queue-to split`** command.

check skill for available agents with level preference

This form of the **`check`** command checks a skill for available agents with a preferred skill level or preferred skill level range for possible termination of the call to the particular skill when there is more than one available agent to choose from. The command attempts to route the call to an agent with the specified skill level in the skill.

Under agent surplus conditions, the skill level preference parameter on the **`check skill vectoring`** command allows you, for instance, to indicate a preference to route high-value and critical calls to the best agents or a preference to route low value calls to trainees or novice agents.

Following is the syntax for this command:

`check skill`	hunt group: 1st, 2nd, or 3rd	pri l = low, m = medium, h = high, or t = top	if	available-agents > `0-1499`	all-levels			
					pref-level	skill level 1		
					pref-range	skill level 1	to	skill level 2

If you select the pref-level parameter, the system displays only the **Skill Level 1** field, in which you can enter a skill value between 1 and 16. If you select pref-range, the system displays two fields, **Skill Level 1** and **Skill Level 2**. Using these two fields, you can enter a range of preference levels, such as 5 (Skill Level 1) to 13 (Skill Level 2). The values in both these fields need to be between 1 and 16. The number in **Skill Level 2** field needs to be greater than or equal to the number you enter in the **Skill Level 1** field.

Valid entries	Usage
skill	Skill level preference options are only available for the check skill version of the command, not for check split or check best.
if available-agents > 0 or greater	This option ensures that the skill level preference is applied only in agent surplus conditions.
all-levels	The system ignores the skill level of the agent. This is the default value.
pref-level	The system displays the **Skill Level 1** field in which you can enter a skill level value for the agent from 1 to 16. Preference level of 1 is the best skill while 16 is the least.
pref-range	The system displays two fields, **Skill Level 1** and **Skill Level 2**. Using these two fields, you can enter a range of preference levels, such as 5 (Skill Level 1) to 13 (Skill Level 2). The values in both these fields need to be between 1 and 16. The number in **Skill Level 2** field needs to be equal to or greater than the number you enter in the **Skill Level 1** field.

The following sample **`check skill vector`** command illustrates the use of skill level preference:

```
check skill 5 pri h if available-agents > 0 pref-range 1 to 3
queue-to skill 17 pri t
```

You must always have a **queue** command following the **check** command for the case where the available-agents conditional fails. This way the call will queue to the skill for service when the agent becomes available.

In this example, the vector checks for an available agent with skill 5 and one of the preferred skill levels, 1, 2, or 3. If there is one, the call is routed to that agent. If there is an available agent with skill 5 but not with one of the preferred skill levels, the call is routed to that agent. Otherwise, the next vector step executes and queues the call to skill 17.

Answer supervision considerations for the check command

Since the **check** command merely checks the status of a split/skill, no answer supervision is returned to the serving Central Office (CO).

check command feature interactions

You can use the **check** command to access a messaging-system, a message center, or a server split or skill in cases where a VDN is assigned as a coverage point. To enable this function, you must assign the split or skill as a vector-controlled hunt group.

For BSR and LAI, the **check** command can be treated either as a call acceptance vector command or a neutral vector command.

The **check** command is treated as a call acceptance vector command if one of the following conditions is true:

- Call terminates to an agent.
- Call queues to a split or skill.
- BSR interflowed call is accepted at remote interflow vector.

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

No Class of Restriction (COR) checking is carried out when a **check** step places a call to a split or skill.

The **oldest-call-waiting** condition checks the priority level “l” (low).

check command interactions with CMS

Calls answered using the **check** command are indicated as answered by backup in CMS.

Calls queued using a **check split/skill** command are tracked as CALLSOFFERRED and LOWCALLS/MEDCALLS/HIGHCALLS/TOPCALLS.

The presence of the command in a vector enables the calls serviced by the vector to be vector-directed. When such a call is answered by an agent, the call is tracked as ACDCALLS/ANSTIME,

and it is reported as ACD Calls, Split/Skill ACD Calls, and Avg Speed Ans. If the call is also queued to other splits/skills, OUTFLOWCALLS/OUTFLOWTIME is tracked in the first split/skill to which the call queues, and Flow Out is reported (unless the split/skill turns out to be the answering split/skill). DEQUECALLS/DEQUETIME is tracked in the second and third splits/skills if these splits/skills are not the answering split/skill, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time. However, if the second or third split/skill is the answering split/skill, INFLOWCALLS is tracked in the split/skill, and the call is reported as Flow In.

Whenever the call is answered in a split/skill accessed by the **`check split/skill`** command, the BACKUPCALLS data base item is incremented, and the call is reported as Calls Ans in Backup and Calls Handled/Backup. The Calls Ans in Main report item is calculated by using the algorithm ACDCALLS - BACKUPCALLS.

If the call abandons after the command queues the call to a split/skill, ABNCALLS/ABNTIME is tracked for the vector, the VDN, and the first split/skill to which the call is queued. The call is reported as Aban Call and Avg Aban Time. If the call is also queued to other splits/skills, DEQUECALLS/DEQUETIME is tracked in these splits/skills, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time.

BSR status poll calls are not counted as interflows. BSR interflows are now tracked as network interflowed calls (NETCALLS) by the CMS at the receiving switch. The CMS tracks a call's accumulated time-in-VDN as NETINTIME (that is, the NET_TIME value on the CMS at switch C combines the time a call has spent in VDNs at any previous locations, as communicated by information forwarding. The NETINTIME can be added to the time spent in the local switch to provide reports that include the total time the call has spent in the call center network (e.g., total ASA).

For more information about CMS database items and reports, see *Avaya Call Management System Database Items and Calculations* and *Avaya Call Management System Supervisor Reports*.

check command interactions with BCMS

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

There is no added tracking for calls interflowed by BSR. BCMS tracks the calls as outflow in the VDN report.

collect digits command

Purpose

Use the **collect digits** command so that the caller can enter up to 16 digits from a touchtone or an internal rotary phone. Use the command also so that the vector can retrieve Caller Information Forwarding (CINFO) digits from the network.

Syntax and valid entries

collect digits	ced	for none or [A-Z, AA-ZZ]		
	cdpd			
	1-16	after announcement	extension number. none, A-Z, AA-ZZ, or V1-V9	for none or [A-Z, AA-ZZ]

Requirements

The Avaya Call Center Deluxe package or Avaya Call Center Elite package must be installed. The command is also available with the Automated Attendant Right To Use (RTU).

You must use a minimum of one TN744 Call Classifier circuit pack or TN2182 Tone Clock circuit pack to collect digits from a caller. You must use these packs unless you use the command to collect digits returned by a Voice Response Unit (VRU) or the digits sent by the network.

Use the Vectoring (CINFO) feature to collect Caller-Entered Digits (CED) or Customer Database Provided Digits (CDPD) from the ISDN or AT&T network Intelligent Call Processing (ICP) service or equivalent.

Operation

The **collect digits** command has two modes of operation:

- Collecting digits
- Collecting CINFO digits

Collecting digits

You can use the **collect digits** command so that a caller can enter digits from a touchtone or an internal rotary phone. You can also use an announcement to prompt the caller to enter the digits. If the caller enters incorrect data, you can administer an announcement to prompt the caller to enter an asterisk (*). When the caller enters an asterisk, the system deletes the digits collected for the current **collect digits** command and restarts the digit collection.

Note:

You can set the **Reverse Star/Pound Digit For Collect Step** field to y on the Parameters page of the Feature-Related System Parameters screen. You can set this field to y to reverse the usual handling of the asterisk (*) and pound (#) digits by the **collect digits** vector command. When you set the field to y, the system interprets the asterisk (*) as a caller end-of-dialing indicator. The system also interprets the pound (#) sign as an indiction to clear all previously entered digits for the current collect vector step.

Specify the maximum number of digits that the system can accept from a caller. If the caller enters fewer digits than the administered limit, use an announcement command to prompt the caller to enter a pound (#) sign as an end-of-dialing indicator. If the caller fails to enter the pound (#) sign, an

interdigit time out occurs. The time out ends the **`collect digits`** command and the digits collected before the time out are available for subsequent vector processing. If all the digits strings for all the variations of a specific **`collect digits`** command are ended with the pound (#) sign, the system counts the pound (#) sign as a digit. Therefore, the number of digits collected must include any # to be collected. Otherwise, the terminating # is kept as a dial-ahead digit and is processed by a subsequent **`collect digits`** command.

Processing of the command requires that a Touch-Tone Receiver (TTR) be connected. If the call originates from an internal rotary phone, no TTR is required. TTRs accept the touchtone digits that the caller enters. The system automatically connects the TTRs.

The connection of the announcement prompt is skipped and digit collection begins when 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 to the administered announcement. If the announcement to be connected is busy and if the queue for the announcement is full, or if no queue is available, the calling party continues to hear the current feedback. The system waits 5 seconds and tries to reconnect 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 a 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 can enter digits at anytime. This causes the announcement to be disconnected or removed from the queue and the digit collection phase to begin. If the caller does not enter any digits during the announcement, digit collection begins when the announcement completes.

The interdigit timer starts with digit collection unless the TTR is already in the timing mode, that is, the dial-ahead capability is active and the TTR is not disconnected.

Digits are collected as dialed digits for the current **`collect digits`** command or as dial-ahead digits dialed because of a previous **`collect digits`** command for the next **`collect digits`** command. Digit collection continues for the current command until one of the following conditions exists:

- Maximum specified digits are collected.
- Pound (#) sign is collected.
- Inter-digit timer expires.

The interdigit timer is used for the digit entry time out and is set to 10 seconds by default. However, the timer can be set to a value between 2 to 10 seconds using the **Prompting Timeout** field on the Feature-Related System Parameters screen.

Caution:

Avaya recommends that you do not set the timeout to less than 4 seconds except in special cases. If the timeout is set to less than 4 seconds the short timeout can cause the caller to miss

entering the next digit in a sequence. The caller can miss entering the next digit if unaware that the caller must enter the digits quickly. The setting of this timer is systemwide and affects digit entry for ALL collect digits steps in all vectors.

During digit collection, if the system encounters an asterisk (*), all digits collected for the current collect digits step are discarded. If additional dial-ahead digits occur after the asterisk (*), the digits continue to be processed. If no such digits exist and if no TTR is connected, vectoring continues at the next vector step. If a TTR 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 TTR is connected for a touchtone phone, the interdigit timer is restarted to detect a time out for releasing the TTR. Vector processing then continues at the next vector step. However, Communication Manager continues to collect any subsequent dialed digits, including the pound (#) sign and asterisk (*) digits, to allow the dial-ahead capability. The additional dial-ahead digits are saved for subsequent **`collect digits`** commands providing the caller with a means to bypass subsequent unwanted announcement prompts. A single pound (#) sign can be collected and tested by subsequent **`route-to...if digits`** or **`goto...if digits`** commands. Alternately, any collected digits from callers or CINFO can be passed to a host with an ASAI or forwarded to another site with the Information Forwarding feature enabled. Collection of dial-ahead digits continues until one of the following occurs:

- Vector processing stops or is ended.
- The sum of the digits collected for the current **`collect digits`** command and the dial-ahead digits exceeds the Communication Manager 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 count towards the 24-digit limit, as do any dial-ahead digits entered after the asterisk or pound sign digit.
- The TTR required by the touchtone phone user to collect digits is disconnected. This process 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 **`adjunct routing link`** step is encountered during vector processing.
  - Successful or unsuccessful **`converse-on`** step is encountered during vector processing.
  - 10 second time out occurs, during which time the caller does not dial any digits, asterisks (*) or pound signs (#)
  - A **`collect ced/cdpd digits`** step is processed.

When the TTR is disconnected due to a **`route-to number`**, **`route-to digits`**, **`converse-on`**, or an **`adjunct routing link`** step, all dial-ahead digits are discarded, that is, following a failed

route-to, **converse-on** or **adjunct routing link** step, a subsequent **collect digits** step always requires the caller to enter digits.

Dial-ahead digits are available for use only by subsequent **collect digits** commands. The digits are not used by other vector commands that operate on digits, for example, **route-to digits**, or **goto...if digits**. In addition, the digits are not displayed as part of the **callr-info** button operation until the digits are collected with a **collect digits** command.

Collecting CINFO digits:

With **collect digits** step, you can collect the CINFO digits from the network. When a **collect ced or cdpd digits** step is processed, the system retrieves the first 16 ced or cdpd digits from the ISDN User Entered CODE (UEC) Information Element (IE) that is associated with the call. The command puts the digits in the collected digits buffer. All existing digits in the collected digits buffer are erased. If a TTR is connected to the call from a previous **collect digits** step, the TTR is disconnected.

If the ced or cdpd digits contain invalid digits, that is not *0-9*, *, or #, the digits are not put in the collected digits buffer. However, the collected digits buffer is still cleared and if a TTR is attached, the TTR is disconnected.

If no ced or cdpd digits are received from the network when the collect ced digits or collect cdpd digits step is reached, the step is skipped. However, the collected digits buffer is still cleared and if a TTR is attached, the TTR is disconnected.

An asterisk (*) in the collected digits is treated as a delete character. Only the digits to the right of the * are collected. A # is treated as a terminating character. Only the # and the digits to the left of the # are collected. A single # is put in the collected digits buffer.

The number of ced or cdpd digits to collect cannot be specified in the **collect digits** step. If less than digits are present, all digits are collected. If more than 16 digits are present, the first 16 digits are collected and a vector event is generated. The CINFO ced and cdpd digits can be used with any vector step that uses the digits in the collected digits buffer. When ced or cdpd digits are collected, the digits can be viewed either on a two-line display or by using **callr-info**.

Answer supervision considerations with the collect digits command

Answer supervision is provided as soon as a TTR is connected. The **collect digits** 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.

collect digits command feature interactions

For BSR and LAI, the **collect digits** command is treated as a call acceptance vector command except for the **collect ced digits** and the **collect cdpd digits** commands which are neutral.

collect digits command interactions with CMS/BCMS

Processing of the **collect digits** step causes the collected digits to be passed to the Call Management System (CMS). Digits are not passed to the Basic Call Management System (BCMS).

consider command

Purpose

The **consider** command defines a resource, skill, or location, that is checked as part of a Best Service Routing (BSR) consider series. You can use the command to retrieve the BSR status data for resource comparison. After executing the consider series, a **queue-to best** or **check best** command queues the call to the best resource.

If the **consider** command is in a status poll vector, a **reply-best** step returns the BSR status data to the primary vector on the origin Communication Manager.

Syntax and valid entries

consider	location (multisite BSR only)	1-255, A-Z, AA-ZZ, V1-V9	adjust by 0-100 percent, A-Z, AA-ZZ, or V1-V9
	(Skill) hunt group skills for VDN: 1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	
	(Split) hunt group		

Requirements

For Communication Manager requirements, see *Avaya Aura® Call Center Elite Feature Reference*.

Operation

In order to deliver a call to the resource that can provide the best service, the **consider** command collects and compares information. Whether you use singlesite BSR, multisite BSR, or both, consider steps work very much the same.

Each **consider** command collects status data from one split or skill. Splits or skills on the same Communication Manager are identified by number. Remote locations must be identified by a location number assigned on the BSR Application screen. For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

Consider commands are written in a series of more than two steps called a consider series. The first step in a consider series collects status data from a resource and saves the data to a buffer. The next consider step collects status data on the assigned split or skill and compares the new data to the data in the buffer. If the existing data in the buffer indicates that the first split or skill can provide better service to the call, the data for the first split or skill remains in the buffer as the best data. If the second split or skill can provide better service to the call, the status data replaces the data already in the buffer. Each subsequent step works in the same way, collecting data from one resource, comparing the data to the best data found up to that point and replacing the best data only

if the resource tested by the current step can provide better service to the caller. This series ends when a **queue-to best**, or **check-best** command delivers or queues the call, or when a **reply-best** command returns the best resource data to a primary vector on the origin Communication Manager.

The first consider step in a series shortens the Call Vectoring 15-step time out from 1.0 to 0.2 seconds. The time out is shortened for BSR vectors only, that is, vectors that use the consider series, in order to reduce real-time delays for call processing and to reduce the incidence of race conditions in multisite BSR applications.

User adjustments

In single and multisite BSR, you can use the *adjust-by* parameter of the **consider** command to program preferences into vectors.

If a resource does not have an available agent when the **consider** step tests the resource, the **consider** step collects the Expected Wait Time (EWT) if the call is to be queued to the resource. You can adjust the EWT value, for purpose of calculation only, by assigning a value of 0-100 in the user adjustment. The units of this value are supplied by Communication Manager based on the conditions whenever the **consider** step executes.

For example, in the command **consider split 1 pri h adjust-by 20**, Communication Manager interprets “adjust-by 20” to mean “add 20 percent to the EWT, but add at the minimum 20 seconds”. For EWT of 1-100 seconds, an adjustment of 20 adds 20 seconds. Above 100 seconds, the same adjustment adds 20 percent to the EWT for the split or skill specified in the **consider** step.

Important:

If the user adjustment are defined as a number of seconds, BSR is not be efficient when EWT is high. If the user adjustment is defined as a percentage, BSR is not efficient when EWT is low. Such efficiencies become critical in multisite BSR applications, which involve issues of trunk cost and capacity.

Events that clear best data

User adjustments also apply to available agent situations (with a strategy other than first found) in a manner that is similar to Expected Wait Time (EWT). For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

As the steps in a consider series execute, the status data for the best resource found is kept in a buffer. This best data is unaffected by some call processing events and vector commands, while other events and commands initialize (clear) this buffer. The following table shows you what initializes the best data buffer and what does not.

Initialization of BSR best data	
Events and vector commands that clear best data	**Events and vector commands that do not clear best data**
Execution of any **queue-to** or **check** command	**Converse** command
Vector processing terminates: • **reply-best** command executes • agent answers • successful **route-to** command • successful **adjunct routing link** command • successful **messaging split/skill** command • vector disconnect timeout • **disconnect** command • **busy** command • vector processing reaches last step without call in queue	**Announcement** command
	Collect Digits command
	Unsuccessful execution of a **messaging split/skill** command
	Unsuccessful **adjunct routing link** command
	Goto step/vector with any conditional
	Wait command (with any feedback)
	Unsuccessful **route-to** command
	Vector processing reaches last step while call is still in queue
	Execution of a **consider** step (this will either replace the current best data with new data or leave the current data untouched)

consider command considerations

- Do not use a **consider** series in vector loops.
- Do not use any commands between the steps of a **consider** sequence that can cause a delay. For example, do not use the **announcement** and **wait** commands within a **consider** sequence. You can use the **goto** command steps.
- Arrange the **consider** steps in the order of preference.

  A **consider** step that tests the main or preferred resource must be first in the series. The second **consider** step must test the resource that is the second preference for handling the given call type. To prevent unnecessary interflows, use the **consider** steps for local resources before steps that check remote resources. Arrange the **consider** steps in the order of preference. This is especially important when the active VDN for the call uses the `1st-found` agent strategy, since Communication Manager delivers the call to the first available agent found. Arranging **consider** steps in the order of preference ensures call delivery to the best available resource.

Answer supervision considerations for the consider command

All forms of the **consider** command are ISDN neutral and do not return answer supervision.

consider command feature interactions

Splits used in the **consider** commands must be vector-controlled.

consider command interactions with CMS/BCMS

BCMS does not log Look-Ahead Interflow (LAI) attempts. Therefore, it does not log BSR status polls since these are LAI attempts.

converse-on command

Syntax and valid entries

converse-on	(Skill) hunt group skills for VDN: 1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	passing	6-digit string, *, #, ani, vdn, digits, qpos, wait, [A-Z, AA-ZZ], V1-V9	and	6-digit string, *, #, ani, vdn, digits, qpos, wait, [A-Z, AA-ZZ], V1-V9
	(Split) hunt group			none		none

Requirements

A converse split must be a vector-controlled ACD or non-ACD hunt group.

Note:

You cannot use the **converse-on** command with a non-ACD hunt group if you enable **Expert Agent Selection**.

Operation

You can use the **converse-on** command to integrate Voice Response Units (VRUs) with Communication Manager. The command effects data passing between Communication Manager and the VRU. The caller hears a voice response script housed in the VRU.

For information on call flows, data passing, collection, and return specifications involving the **converse-on** command, see Call flow and specifications for converse - VRI calls on page 200.

If the command is successful, Communication Manager delivers the call to a predetermined split or skill called the converse split or skill. The caller is connected to the VRU, which in turn executes the voice response script. If by this time the call has already queued to a non converse split or skill, the call retains the position in the non converse split or skill queue. If an agent from the non converse split or skill becomes available to answer the call while the voice response script is being executed, Communication Manager drops the line to the VRU and connects the caller to the available agent. The VRU detects disconnection and terminates the voice response script. When a voice response

script is executed, audible feedback provided by the vector is disconnected and no further vector steps are executed until the voice response script execution is complete.

If the voice response script is completed and there is no data to be returned from the VRU to Communication Manager, the VRU drops the line to Communication Manager and vector processing is reactivated on Communication Manager.

If there is data to be returned to Communication Manager, the converse data return code is outpulsed before the data to be passed is outpulsed. Once Communication Manager receives the VRU data, Communication Manager stores the data in the call prompting digits buffer as dial-ahead digits and reactivates vector processing. The caller does not hear the digits returned by the VRU.

Digits returned from the VRU can be:

- Displayed on the agent display set, automatically for 2-line display or by using the **callr-info** button for 1-line display.
- 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.
- Tandemed to an ASAI host.

Communication Manager can be set up to pass information in-band to the VRU. In such a case, the **`converse-on`** command can outpulse up to two groups of digits to the VRU. The digits can serve two major purposes: the digits can notify the VRU of the application to be executed, and the digits can share call related data such as ANI (BN) or caller digits collected by Communication Manager. In many applications, both application selection and data sharing are required. The touch-tone outpulsing rate is adjustable.

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

Sending # prevents excessive delays caused by digit time outs and other problems caused by time outs. Sending # also ensures that each data field is used to satisfy a single **`prompt`** and **`collect`** step.

Any data passed from Communication Manager to a VRU is outpulsed in-band. The user can administer two time delays on the Feature-Related System Parameters screen: converse first data delay and converse second data delay fields. The delays range is from 0 to 9 seconds with a default of zero seconds for the converse first data delay and a default of two seconds for the converse second data delay. The delay is required to allow VRU 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.

Call flow and specifications for converse - VRI calls

This section details call flow for calls involving a **converse-on** vector step and Voice Response Integration (VRI). This call flow is segmented into the following phases:

- Converse call placement
- Data passing
- VRU data collection
- Script execution
- Data return
- Script completion
- Switch data collection

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 **converse-on** step delivers the call to the converse split. The caller does not hear the ringback tone. Any audible feedback supplied by vector processing remains until the VRU answers the call and all digits have been outpulsed to the VRU. Vector processing is suspended. Callers remain in any non converse split queues and retain their position in queue while the converse session is active.

If a Call Prompting TTR is allocated to the call, the TTR is released. Any dial-ahead digits are discarded. However, any digits collected prior to the **converse-on** step are retained.

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, the 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 (RONA) feature has taken all the ports out of service.

The display is not changed by the terminating or answering of a converse call. Also, whenever a call is delivered to a display station using 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
- Agent including Auto-Available skill groups
- AUDIX hunt groups

Note:

AUDIX hunt groups are valid destinations for converse calls and do not have to be vector-controlled.

Undefined and non vector-controlled hunt group, split or skill numbers are rejected at administration time.

Any attempt to remove a hunt group, split or skill administered within a **converse-on** vector step is denied until the vector has been changed. Also, any attempt to make a hunt group, split, or skill non vector-controlled is denied if the hunt group, split, or skill is called by a **converse-on** step.

Data passing

The data passing 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 can outpulse up to two groups of digits to the VRU. The digits can serve two major purposes:

- Notify the VRU of the application to be executed.
- Share call-related data collected by the switch. This includes ANI, CINFO, or caller digits.

In many applications, both application selection and data sharing are required.

Using the pound (#) sign

Since the digit strings are of variable length, Communication Manager always appends a pound (#) sign to the end of each digit string. Administer the prompt and collect steps in the VRU script to receive 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 timeout.

How the outpulse sequence works

The complete outpulse sequence is summarized as follows:

1. The VRU answers the call.
2. Delay for the time administered in the Converse first data delay field in the System Parameters-Features screen occurs.
3. The <data_1> is outpulsed.
4. The pound sign is outpulsed.
5. Delay for the time administered in the Converse second data delay field in the System Parameters-Features screen occurs.
6. The <data_2> is outpulsed.
7. The pound sign is outpulsed.

Note:

The length of DTMF tones and the inter-digit pause between tones is administrable on the Feature-Related System Parameters screen. The optimum settings for Conversant/IR are 60 msec tones and 60 msec pauses that provide an 8.3 digits-per-second rate. These changes differ from the administration default.

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.

Values administered for <data_1> and <data_2>

You can administer the following values for <data_1> and <data_2> within the **`converse-on`** command:

Value	Description
Administered digit string	This string can contain up to six characters consisting of more than one digit (0 through 9) or asterisks (*). The pound sign (#) cannot be included in a digit string because the pound sign is reserved as the end-of-string character. However, you can administer a single pound sign.
ani	If the call is a local call or an incoming DCS call, this data type causes the extension of the calling party to be outpulsed. If the call is an incoming ISDN PRI call with ANI (BN) provided to the switch, the calling party number/billing number (CPN/BN) of the calling party is outpulsed to the voice information system. If there is no ANI (BN) to send, the end-of-string pound sign is the only character outpulsed. Any other type of incoming call results in the pound sign being outpulsed.
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.

Table continues…

Value	Description
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 non converse 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 outpulsed. **Note:** Do not use the keyword with multiple split queuing because any queue position value sent is not always meaningful. However, if the call is queued to multiple non converse splits, the value of the caller's queue position in the first non converse split is sent. This data can 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 EWT for a call in vector processing that is queued to a minimum of one split. It is a value from 0 to 9999 seconds, variable length that is not padded with zeros, always followed by a pound sign. If the call is not queued, or is queued only to splits with no working agents, only the pound sign is outpulsed.
A to Z and AA to ZZ	This data type causes the current numeric value of the vector variable to be outpulsed. If the value is undefined, a single # is outpulsed. The vector variable is defined by letters between A to Z and AA to ZZ.
V1 to V9	This data type causes the value of the VDN variable assigned to the active VDN for the call to be outpulsed. If the value is undefined, a single # is outpulsed. The VDN variables V1 through V9 are defined on the VDN screen for each VDN extension.
#	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 sign is outpulsed, and no time delays are invoked. If <data_1> is administered as none, <data_2> must also be none. The switch always outpulses a pound sign at the end of each digit string. Where the pound sign is administered, or where the digits keyword is administered and the last digit collected from the caller is the pound sign, only one pound sign is outpulsed. No pound sign is outpulsed when the keyword none is administered.

Time delay administration

Any data passed to the VRU from the switch is outpulsed in-band. Customers can administer the converse first data delay and the converse second data delay time delays on the Feature-Related System Parameters screen. The delays can range from 0 through 9 seconds, with a default of zero

seconds for the converse first data delay and a default of two seconds for the converse second data delay. The delays are required to give the VRU sufficient 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.

When the VRU hangs up during data passing

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 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.

Ensuring robust operation of VRU data passing

To ensure robust operation of the VRU data passing operation, implement the following:

- Include **`prompt`** and **`collect`** commands in the VRU script for each data field passed in the **`converse-on`** step.
- Administer each **`prompt`** and **`collect`** command to recognize the pound sign (#) as the end-of-string character.
- Ensure the number of digits expected is one greater than the number of digits passed to allow for the pound sign, which terminates every converse data field.

  Do not use an **`announcement`** command in the **`prompt`** and **`collect`** steps.
- Ensure the first digit time out in the **`prompt`** and **`collect`** steps is five 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 seconds and the converse second data delay is 4 seconds, the first digit time out for the two **`prompt`** and **`collect`** commands must be a minimum of 5 and 9 seconds, respectively.
- Ensure the inter-digit time out in the **`prompt`** and **`collect`** steps is five seconds at the minimum.
- Administer the converse first data delay so that a VRU under a heavy load has time to allocate a DTMF touch-tone receiver after answering the call.
- Administer the converse second data delay so that a VRU under a heavy load has 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 to pass data to the VRU, ensure the VRU script does not execute any commands between the time 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, 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.

If the first digit string, that is, <data_1>, passed to the VRU is for application selection, the Avaya Interactive Response Script Builder exec command invokes the appropriate script. If a second digit string, that is, <data_2>, is also used to pass an argument to this selected application, the first command in the executed script is a prompt and collect command with no announcement prompt programmed and with the pound sign (#) programmed as the end-of-string character.

“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 must ensure the administered converse first data delay and converse second data delay timers allow 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 the converse second data delay timer needs to be increased or the VRU timing values require tuning as appropriate to resolve issues.

Default and IVR converse settings

The default for the converse signaling tone and pause on the Feature-Related System Parameters screen is a 100 msec tone and 70 msec pause. This results in a 5.5 digits per second rate that provides a more conservative sending rate to support most VRUs.

For operation with Avaya Conversant or Avaya IR, change the default settings to a 60 msec tone and a 60 msec pause. This results in a more optimum rate of 8.3 digits per second supported by these products.

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 non converse 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 can 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.

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 FAC administered on the Feature-Access-Codes screen. The second sequence of digits is the sequence to be passed by the VRU. These digits are collected later during vector processing.

The Avaya Interactive Response VRU offers a built-in external function called converse_data. This function allows applications developers to perform this operation in a convenient and robust fashion.

To ensure the robust operation of the VRU data return operation, perform the following actions:

- Set the analog flash timing to 600 msecs.
- Ensure DTMF tones last a minimum of 70 msec and interdigit pauses last a minimum of 50 msec. This results in an outpulsing rate up to 8.33 digits per second.
- (Avaya Interactive Response only) Use the converse_data external function to return data to the switch.
- Hang up the line to the switch after outpulsing digits. The switch waits between 1.2 and 1.5 seconds to determine that the hang-up is a disconnect.

For applications involving VRUs other than Avaya Interactive Response VRUs, perform the following actions:

- After the flash, ensure the VRU performs dial tone detection, stutter dial tone, to ensure accurate detection (typically 0.6 to 1.0 secs) before outpulsing the converse data return FAC.
- If no dial tone is received before the time out, ensure the VRU does two more retries of the analog flash. Also, if no dial tone is detected after two retries, ensure the VRU logs an error.
- Whenever a dial tone is detected, ensure the digits of the converse data return FAC are outpulsed.
- After the converse data return FAC 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.

For an outpulse rate of 8 digits per second, that is, 0.125 seconds per digit, a 3-digit FAC and stutter dial tone detection time of 0.6 seconds, a maximum of 24 digits passed to switch must take about 6 seconds, that is, 1.2 seconds disconnect plus 8 seconds plus 0.125 seconds per digit.

The Call Classifiers required by Call Prompting are not required for returning digits in-band from the VRU to the switch. Instead, general purpose TTR boards are used. As long as dial-ahead digits are available, any **`collect digits`** steps following a **`converse-on`** step do not require a Call Classifier to be allocated to the call.

If no general purpose TTRs are immediately available, and if the call queues for a TTR, no dial tone is provided. For this scenario, the VRU does not outpulse any digits until a TTR is available and dial tone is provided.

If there are no general purpose TTRs available on the switch, and if there is no space in the TTR 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 TTRs.

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 prevent the ten second time out delay that occurs when the digits are collected. If the digit string being returned is multipart, 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 (*) can 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 an asterisk as a returned digit. If you do use an asterisk 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 must 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 are not collected as dial-ahead digits by the switch.

If an interdigit time out 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 time out 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 Communication Manager by simply hanging up the line. In cases where no data is returned to Communication Manager, control is returned by executing the **`quit`** command. In cases where data is returned, control is returned when the VRU hangs up on 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 can be displayed by an answering agent on the non converse 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 can involve either a final message to a host or a final update to a local database. For the scenario, the VRU port is still associated with the running script even though there is no longer a voice connection.

From Communication Manager point of view, the agent 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 ACW periods must be prevented, especially if such periods are so lengthy approaching the administered time out period on Communication Manager for the RONA feature. In such a case, RONA treats the VRU ports as faulty and starts taking the ports out of service.

Switch data collection

This phase is in effect only if the VRU returns information to the switch.

Once the VRU script is complete and vector processing is reactivated, the returned digits are collected and processed by vector commands. Since the digits must be collected by a **`collect digits`** command, data can be returned and processed only if you enable **Call Prompting**.

The data returned can consist of multiple parts. For example, the VRU can return a stream of seven digits in which a single digit success or fail code is followed by a six-digit account code. For this scenario, the **`converse-on`** step is followed by a sequence of vector steps including two **`collect digits`** steps. The first **`collect digits`** step can collect one digit and then check the result code. The second **`collect digits`** step can 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 TTR when dial-ahead digits are present. Since VRU-returned digits are treated as dial-ahead digits, a TTR 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.

Data 1 and Data 2 values administered within the converse-on command

The following values can be administered for <data_1> and <data_2> within the **`converse-on`** command:

- Administered digit string: Contains up to six characters consisting of more than one digit, 0 through 9, or asterisks (*). Do not include the pound sign (#) in a digit string because the pound sign is reserved as the end-of-string character. However, you can administer a single pound (#) sign.
- ani: If the call is an internal call or an incoming DCS call, this data type causes the extension of the calling party to be outpulsed. If the call is an incoming ISDN-PRI or R2-MFC Signaling call with ANI (BN) provided to the switch, the Calling Party Number/Billing Number (CPN/BN) of the calling party is outpulsed to the VRU. If there is no ANI (BN) to send, the end-of-string pound (#) sign is the only character outpulsed. Any other type of incoming call results in # being outpulsed.
- digits: Use the data type only if you have enabled **Call Prompting**. To pass CINFO digits, you must also enable **Vectoring (CINFO)**. 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: No characters are outpulsed. Also, no end-of-string pound (#) sign is outpulsed and no time delays are invoked.
- qpos: Causes the value of the queue position of a call in a non converse 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. VRUs can use the data to inform callers of the position in queue or to decide whether to execute a long or short version of a voice response script.

  Note:

  Do not use the `qpos` keyword with multiple split or skill queuing. Any queue position value that is sent is not always meaningful. If the call is queued to multiple non converse splits or skills, the value of the queue position in the first non converse split or skill is sent. Priority queuing and Dynamic Queue Position (DQP), which is available with Avaya Business Advocate, can put subsequent calls into the queue ahead of the waiting call.
- vdn: Causes the VDN extension to be outpulsed. In cases where multiple VDNs are accessed, normal VDN override rules determine which VDN extension is outpulsed.
- wait: Used the data type only if you administer the **Vectoring (G3V4 Advanced Routing)** customer option. The data type causes the expected wait time of the call in seconds to be outpulsed. For a detailed description of expected wait time, see *Avaya Aura® Call Center Elite Feature Reference*. If the call is not queued, is queued only to splits that are unstaffed, or splits where all agents are in the AUX work mode, the end-of-string pound (#) sign is the only character outpulsed. The value outpulsed is a variable number not padded with zeroes. The

value is a maximum of four digits always followed by the pound (#) sign. The range is 0# to 9999# or a single #.

- A to Z, AA to ZZ: Causes the current numeric value of the vector variable to be outpulsed. If the value is undefined, a single pound (#) sign is outpulsed. The vector variable is defined by letters between A to Z and AA to ZZ.
- V1 to V9: Causes the current value of the VDN variables assigned to the active VDN for the call to be outpulsed. If the value is undefined, a single # is outpulsed. The VDN variable is defined by the letter V followed by a number between 1 and 9.
- #: The only character that is outpulsed. Outpulsing this character causes the corresponding **`prompt`** and **`collect`** command in the voice response script to be skipped.

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

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

converse-on split command

Voice Response Integration (VRI) allows integration of Call Vectoring with the capabilities of voice response units (VRUs), particularly the Avaya Interactive Response (IR) system.

VRI capabilities

VRI can do the following:

- Execute a VRU script while retaining control of the call in vector processing.

  Note:

  If an agent becomes available to service the call, the line to the VRU is immediately dropped, and the calling party is connected to the available agent.
- Execute a VRU script while the call retains its position in the queue.
- Group VRU ports for multiple applications.
- Use a VRU as a flexible external announcement device.
- Pass data between the switch and a VRU.
- Tandem VRU data through the switch to an Adjunct Switch Application Interface (ASAI) host.

The capabilities listed above 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 integrates a VRU with the Communication Manager.

VRI benefits

Use of VRUs with vector processing offers 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.

VRI allows users to make 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, you can resolve the caller's questions while the call is in queue. This can help reduce the queuing time for all other callers during peak intervals.

When you administer **Advanced Vector Routing**, the Expected Wait Time (EWT) for a call can be passed to the VRU and conveyed to the caller.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

VRI considerations

- If you want the callers to hear the entire voice response script before speaking to an agent, queue the call only after a **`converse-on`** step executes.
- You must provide an audible feedback prior to a **`converse-on`** step whenever a large number of digits is to be outpulsed to the VRU.

Using converse-on to outpulse caller information to VRUs

You can use the **`converse-on`** command to outpulse the following types of information to a VRU:

- VDN extensions.
- Calling party extensions.
- Collected caller digits, if you enable Call Prompting.
- Expected Wait Time, if you enable Advanced Vector Routing.
- Call queue positions.
- A string of a maximum of six digits or asterisks, or a pound sign (#).
- Variables A to Z and AA to ZZ.

The following example shows a vector in which the **converse-on** command is used to outpulse VDN extensions to the VRU in a way that allows a single vector to be used by multiple VDNs.

```
VDN (extension=1040   name=``car loans''     vector=40)
VDN (extension=1041   name=``equity loans''  vector=40)
Vector 40
     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 seconds hearing music
     6. converse-on split 11 pri h passing vdn and none
     7. wait-time 20 seconds hearing music
     8. announcement 4001
     9. goto step 7 if unconditionally
    10. busy
```

In the example, a vector can be used to respond to calls that originate from VDNs that serve customer needs such as car loans and equity loans.

If vector processing proceeds to step 6, the **converse-on split** command delivers the call to the converse split.

Note:

If an agent on Communication Manager is available to respond to the call, the line to the VRU is immediately dropped and the calling party is connected to the available agent.

As shown in step 6, when the VRU port responds, vector processing outpulses the VDN associated with the call to the VRU by way of the “passing vdn” parameter. Based on the VDN number, the VRU executes the appropriate voice response script for the caller.

Before connecting to a VRU, include a vector step to test if time is available for a voice response script to be executed. In the example, step 4 includes a “calls-queued” condition used for the purpose.

Provide a feedback step prior to the **converse-on** step in case there is a delay in reaching an available converse split port. In the example, step 5 provides music for the purpose.

For more information on the call flow for converse-VRI calls, see Call flow and specifications for converse - VRI calls on page 200.

Answer supervision considerations for the converse-on split command

Call processing returns to answer supervision 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.

converse-on split command feature interactions

Interaction	Description
Abandon Call Search	If the **converse-on** step places a call to a hunt group and if the incoming call is placed using a trunk group with Abandon Call Search activated, the system checks if the calling party has abandoned the call, that is, hung up, before terminating the call to an agent.
ASAI	Since vector-controlled splits or skills cannot be ASAI-monitored domains, ASAI cannot be used to supplement the operation of the **converse-on** step. If a **converse-on** step places a call to an ASAI-monitored domain, ASAI event messages are sent over the ASAI link. Whenever a **converse-on** step places an ASAI-monitored call, the ALERTing message sent to the ASAI host includes a Cause IE, Coding Standard 3 value 23 (CS3/23). This informs the ASAI host that the call has not been de-queued from any non converse splits or skills. If a **converse-on** step is executed while an adjunct routing request is outstanding, the route request is canceled.
AAS	A **converse-on** step can place a call to AAS. In cases where the converse split or skill is ASAI-controlled, use auto-available converse splits or skills for Voice Response Integration (VRI).
Call Coverage	Call Coverage does not apply because the **converse-on** step can deliver calls only to vector-controlled splits or skills, which do not have coverage paths.
Call Detail Recording (CDR)	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 cannot be parked.
Call Pickup	Calls placed by a **converse-on** step ringing at an agent station can be picked up if that agent is part of a pickup group. Subsequent transfers are denied.
Call Prompting	To gain full VRI functionality, you must enable the **Call Prompting** field. Without Call Prompting, any data returned by the VRU cannot be collected and processed by Communication Manager. If the **converse-on** step places a call to a split or skill of live agents, any digits collected previously can be displayed by agents using the **callr-info** button.
Call Vectoring (Basic)	The **converse-on** step is an enhancement to the **Call Vectoring (Basic)** customer field. You must enable the field to invoke the VRI feature.
Class of Restriction (COR)	As is the case for the **queue-to** split or skill and **check** split or skill vector steps, no COR checking is carried out when a **converse-on** step places a call to a split or skill.
Conference	Any attempt to conference a call placed by a **converse-on** step is denied.

Table continues…

Interaction	Description
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 using a VDN in a Coverage Path (VICP), coverage callback cannot be used.
Direct Department Calling (DDC)	You can administer a converse split as a DDC split.
Distributed Communications System (DCS)	If an incoming DCS call is placed to a vector with a **converse-on** split or skill x pri y passing ani ... step, the DCS extension of the calling party is outpulsed.
Priority Levels	A call placed by a **converse-on** step can be queued at one of four priority levels: low, medium, high or top.
Hunt Groups	The **converse-on** step can deliver a call to a vector-controlled hunt group, ACD split or skill, message center or a messaging-system hunt group.
Integrated Services Digital Network (ISDN)	The **converse-on** step can be administered to outpulse to the VRU with the ANI CPN/BN of the calling party. The outpulse uses an ANI keyword.
Intercept Treatment	A caller is not 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	Communication Manager 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.
LAI	If a call placed by a **converse-on** vector step is answered by a VRU, or if such a call queues to a split or skill on the receiving Communication Manager while a LAI call attempt is outstanding, the LAI call attempt is accepted. A **converse-on** step that fails is neutral.
Message Center	The **converse-on** step can deliver calls to message hunt groups. Such calls are treated as direct calls to the message. 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.
Messaging System	If a **converse-on** step calls the messaging system, the call is treated as a direct call to the messaging system. The caller hears the welcome message and can retrieve the messages in the usual manner.

Table continues…

Interaction	Description
	If a call is forwarded to or covers to a VDN and is then delivered to a messaging-system hunt group by a **converse-on** step, the call to the messaging system is treated as a redirected call, and the caller can leave a message for the principal.
Multiple Split or Skill Queuing	A call can be queued to three different splits or skills and then to a converse split or skill 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 or Skills	A **converse-on** step does not place a call to a non vector-controlled split or skill.
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 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 non converse split or skill, the call remains in queue for the non converse split or skill. The queue priority of the call is administrable on the vector step.
Recorded Announcement	VRI can 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 (RONA)	If a **converse-on** step places a call to a hunt group with a no answer time out administered and if the call rings at an agent terminal or port for longer than the administered time out, the call is redirected and the agent or port is put into the Aux work state, or is logged out if the agent is a member of an auto-available split or skill. Thereafter, under RONA, the call is re-queued to the split or skill unless there is no room in the queue or unless this is an AAS whose agents are all logged out. If the call cannot be re-queued, 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 can 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. Do not administer a service observing warning tone since the warning tone can interfere with the interaction between the VRU and the calling party.

Table continues…

Interaction	Description
System Access Terminal (SAT)	You can administer the **converse-on** steps from the SAT terminal.
System Measurements	System measurements track converse calls to hunt groups and attendant groups.
Timed After Call Work (ACW)	Timed ACW cannot be assigned to AAS. If a call to a VDN with Timed ACW routes to a converse split, the VDN Timed ACW does not apply. If Timed ACW is assigned to a non-AAS split that is a converse split, the Timed ACW of the split does apply.
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 Communication Manager. Only after the digit passing phase from Communication Manager to the VRU is completed can the calling party enter touch-tone digits in response to a VRU prompt. Only after the VRU to Communication Manager 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 Communication Manager prompt.
Transfer	A call placed by a **converse-on** step cannot 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 VRU 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 can reconnect to the call.
Transfer out of messaging system	If a **converse-on** step delivers a call to a messaging-system hunt group and if the calling party attempts to transfer out of a messaging system, the transfer fails, and vector processing is reactivated at the next vector step.
Uniform Call Distribution (UCD)	You can administer a converse split or skill as a UCD split or skill.
VDN as a Coverage Point	Coverage attempts are denied during the following conditions: • If the **converse-on** command processes a call covering to a VDN and directs the call to a station user, that is, a member of a converse split or skill. • If the converse split or skill agent attempts to activate Consult or Coverage Leave Word Calling. The attempts are denied because the call is still in vector processing. If the converse split or skill is a messaging-system or message center split or skill, the call covered to the VDN is treated as a redirected call to the messaging system or MCS. The original principal and the reason for redirection is used in the same manner as a call forwarded call to a VDN.

Table continues…

Interaction	Description
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 the CMS and BCMS VDN reports, a **converse-on** step is treated like an **announcement** step. A call is treated as answered when the call is answered by a non converse split or skill, but not when answered by a converse split or skill.
Vector-controlled Splits or Skills	A **converse-on** step can place a call to a split or skill only if that split or skill is administered as a vector-controlled split or skill.

converse-on split command interactions with CMS

CMS tracks calls placed by a **converse-on** step to a CMS-measured split or skill. Since a **converse-on** step allows a call to be answered in more than one split or skill, trunk totals do not match split or skill totals. However, VDN totals and trunk totals do match.

For call tracking in the CMS VDN reports, a **converse-on** step is treated like an **announcement** step. A call is treated as answered when the call is answered by a non converse split or skill, but not when the call is answered by a converse split or skill.

converse-on split command interactions with BCMS

BCMS tracks calls placed by a **converse-on** step to a BCMS-measured split or skill. Since a **converse-on** step allows a call to be answered in more than one split or skill, trunk totals do not match split or skill totals. However, VDN totals and trunk totals do match.

For call tracking in BCMS VDN reports, a **converse-on** step is treated like an **announcement** step. A call is treated as answered when the call is answered by a non converse split or skill, but not when answered by a converse split or skill.

disconnect command

Purpose

Use the **disconnect** command to end call treatment and remove the call from Communication Manager. You can optionally assign an announcement that plays immediately before the execution of the **disconnect** command.

Important:

Always warn the caller before disconnecting a call.

Syntax and valid entries

disconnect	after announcement	extension number
		`none`
		`A-Z, AA-ZZ`
		`V1-V9`

Requirements

You must record and administer an appropriate announcement.

Operation

The **disconnect** command forces a call to disconnect. Any previously established call treatment ends when the **disconnect** command is executed and the call is removed from vector processing and from Communication Manager.

If the call is connected to a station while the announcement is playing, the announcement stops and the caller hears ringback. Also, because vector processing stops when the call connects to a station, the disconnect portion of the command is not processed.

When the **disconnect** command includes an announcement, Communication Manager sends answer supervision just before the announcement plays.

When the **disconnect** command does not include an announcement, Communication Manager sends answer supervision before disconnecting a call.

Note:

Answer supervision is not sent for ISDN trunks.

Call disconnect with announcement

An announcement is played before the call terminates.

```
 disconnect after announcement 2918 [Today has been declared a snow day. Please report
for work tomorrow at 8 P.M.]
```

Answer supervision considerations for disconnect command

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.

disconnect command feature interactions

For LAI, the **disconnect** command is treated either as a call acceptance vector command or a call denial vector command.

The **disconnect** command is treated as a call acceptance vector command when an announcement is included within the **disconnect** command and one of the following conditions is true:

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

The **disconnect** command is treated as a call denial vector command when one of the following conditions is true:

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

disconnect command interactions with CMS

DISCTIME, OTHERTIME, and INTIME for splits and vectors are tracked according to when the announcement starts. DISCTIME, OTHERTIME and INTIME for VDNs are tracked according to when the trunk idles.

Disconnect command	
Database item	**Report heading**
DISCCALLS/DISCTIME	Calls Forced Disc
	Calls Busy/Disc
OTHERCALLS/OTHERTIME	Inbound Other Calls
INTIME	Avg Time In Vector

disconnect command interactions with BCMS

A call that is disconnected using the **disconnect** command is tracked as OTHER in the VDN report.

goto step and goto vector command

Purpose

Use the **goto step** and **goto vector** commands for conditional or unconditional branching to a preceding or subsequent step in the vector or to branch the vector process to another vector. The goto vector step does not remove a call from queues in which the call is already placed.

All parameters, options and value limits are identical for the **goto step** and **goto vector** commands.

Syntax and valid entries

Note:

The maximum vector limit is less on some platforms. Use the help key to determine the applicable limit for your system. The maximum number of port networks and media-gateways varies with the server platform. For example, the S8710 server supports up to 64 port networks and 250 media gateways. Check capacity tables for supported limits.

`goto step 1–99` **if** `goto vector 1– 8000` **@ step** `1–99` **if**					
A-Z or AA-ZZ *V1-V9*			>, <, =, <>, >=, or <=	threshold value or digit string: 1–16, wildcards (? or +), A-Z or AA-ZZ, V1–V9	
			= or <>	none or pound (#) sign	
			in table not in table	1–999, A-Z or AA-ZZ, V1–V9	
ani			>, <, =, <>, >=, or <=	threshold value or digit string: 1–16, wildcards (? or +), A-Z or AA-ZZ, V1–V9	
			=, or <>	none or pound (#) sign	
			in table not in table	1–999, A-Z or AA-ZZ, V1–V9	
available agents	in skill	hunt group skill for VDN: 1st, 2nd, or 3rd	>, <, =, <>, >=, or <=	0-1499, 1-1500, A-Z or AA-ZZ, V1-V9	
	in split	hunt group			
calls-queued	in skill	hunt group skill for VDN: 1st, 2nd, or 3rd	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-098, 1-999, A-Z or AA-ZZ, V1-V9
	in split	hunt group			
counted-calls	to vdn	extension, latest, or active	>, <, =, <>, >=, or <=	0-098, 1-999, A-Z, AA-ZZ, V1-V9	
digits			>, <, =, <>, >=, or <=	threshold value or digit string: 1–16, wildcards (? or +), A-Z or AA-ZZ, V1–V9	
			= or <>	none	
			=	meet-me access. You can use the option only with meet-me conference vectors.	
			in table not in table	1–999, A-Z or AA-ZZ, V1–V9	

Table continues…

`goto step` 1–99 if **`goto vector` 1– 8000 @ step 1–99 if**					
expected-wait for	best		>, <, =, <>, >=, or <=	0-9999 (in seconds), A-Z or AA-ZZ, V1-V9	
	call		>, <, =, <>, >=, or <=		
	split		hunt group	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-9998 (in seconds), 1-9999 (in seconds), A-Z or AA-ZZ, V1-V9
	skill		hunt group skill for VDN: 1st, 2nd, or 3rd		
holiday			in table not in table	1-999, A-Z or AA-ZZ, V1-V9	
ii-digits			>, <, =, <>, >=, or <=	2–digit string, wildcards (? or +), A-Z or AA-ZZ, V1-V9	
			= or <>	none	
			in table not in table	1–999, A-Z or AA-ZZ, V1-V9	
interflow-qpos			>, <, =, <>, >=, or <=	1–9, A-Z or AA-ZZ, V1-V9	
media-gateway			H.248 gateway ID 1–999	= or <>	registered
			all		
			any		
`goto step` command only • meet-me full • meet-me idle The options are available only with meet-me conference vectors					
no match					
oldest-call-wait in	skill	hunt group for VDN: 1st, 2nd, or 3rd	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-998 (in seconds), 1-999 (in seconds), A-Z or AA-ZZ, V1-V9
	split	hunt group			
port network			port network ID 1– 999	= or <>	registered
			all		
			any		

Table continues…

`goto step 1-99` if **`goto vector 1- 8000` @ step `1-99` if**						
queue- fail. Is available only with the Attendant Vectoring feature						
rolling-asa for	skill		hunt group for VDN: 1st, 2nd, 3rd	>, <, =, <>, >=, or <=	0-998 (in seconds), 1-999 (in seconds), A-Z or AA-ZZ, V1-V9	
	split		hunt group			
			vdn extension, latest, or active			
server			= or <>	main, ess, or lsp		
service-hours			in table not in table	1-999 (in seconds), A-Z or AA-ZZ, V1-V9		
staffed-agents	skill	hunt group for VDN: 1st, 2nd, or 3rd	>, <, =, <>, >=, or <=	1-1499, 1-1500, A-Z or AA-ZZ, V1-V9		
	split	hunt group				
time-of-day is	mon, tue, wed, thu, fri, sat, sun, or all		hour: 00–23 minute: 00–59	to	mon, tue, wed, thu, fri, sat, sun, or all	hour: 00–23 minute: 00–59
wait-improved	best		>, <, =, <>, >=, or <=	0-9998 (in seconds), 1-9999 (in seconds), A-Z or AA-ZZ, V1-V9		
	skill		hunt group for VDN: 1st, 2nd, 3rd	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-9998 (in seconds), 1-9999 (in seconds), A-Z or AA-ZZ, V1-V9
	split		hunt group			
unconditionally						

- Wild cards: The question (?) mark matches any digit from 0 to 9 at the specified position. The plus (+) sign matches any or no characters at the specified position.
- Threshold field test: Use the word none to test for an empty digits string. Use the pound (#) sign to match a single pound (#) sign that the caller enters or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
- VDN latest and active: Latest refers to the VDN specified for the current vector and active refers to the VDN specified by the VDN Override settings.

goto step and goto vector commands operation

Basic 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 average speed of answer for the indicated split/skill or VDN meets the constraints defined by the comparator and threshold value.
- The number of available agents in the indicated split/skill meets the constraints defined by the comparator and the threshold value.
- The number of queued calls in the indicated split/skill 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 expected wait time at the specified priority level for the indicated split/skill, or for the call meets the constraints defined by the comparator and the threshold value.
- The oldest call-waiting in the indicated split/skill 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/skill meets the constraints defined by the comparator and the threshold value.
- Digits collected using the **`collect digits`** command match the criteria defined by the comparator for the specified digit string. Or, the digits are found or not found, depending upon the option chosen, in the specified Vector Routing Table. The # digit can be tested against as a single digit.
- The ani digits match the criteria defined by the comparator for the specified digit string. Or, the ani digits are found or not found, depending upon the option chosen, in the specified Vector Routing Table.
- The II-digits match the criteria defined by the comparator for the specified digit string. Or, the II-digits are found or not found, depending upon the option chosen, in the specified Vector Routing Table.
- 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.
- The Expected Wait Time (EWT) for the call is decreased by a period of time within the constraints defined by the comparator and the threshold value, which is expressed in seconds. The improvement in EWT is defined by calculating the difference between the call's current EWT and its EWT were it to be queued to the resource specified in the command.

- The call's position in the interflow-eligible portion of the queue meets the condition defined by the comparator and the threshold value (representing queue position counting backward from 1, which is the head of the eligible queue).
- For Attendant Vectoring, there is no way to check ahead of time to see if a call can queue, and there is no way to check if, after the fact, a call queued successfully. The **queue-fail** command allows you to provide additional routing if a call to an attendant vector fails. You can redirect the call to another step or to another vector if the call cannot be queued.

General considerations

When a **goto** command is used in a vector step to connect to a different VDN, the following events occur:

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.

Unconditional branching

Unconditional branching passes control from the current vector step to a preceding vector step, a subsequent vector step, or to another vector. Unconditional branching is implemented when a **goto step** or **goto vector** command is associated with an unconditionally parameter.

The following example shows a vector that uses an unconditional branching step:

Unconditional branching example

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

In the example shown above, the unconditional branch statement in step 7 establishes a loop between steps 5 through 7. Vector processing within the loop terminates when:

- An agent answers the call
- The system recognizes that the caller abandoned the call

Conditional branching

Conditional branching passes control from the current vector step to a preceding vector step, a subsequent vector step, or to another vector. Conditional branching is enabled by a **goto step** or **goto vector** command when a conditional statement is associated with the command.

The list of condition statements that can be assigned, which depends on the features enabled in your Communication Manager installation, is summarized in the following table.

Condition statement	Basic call vectoring	Advanced vector routing	ANI and II-digits routing
available-agents	x	x	x
staffed-agents	x	x	x
calls-queued	x	x	x
oldest call-waiting	x	x	x
time-of-day	x	x	x
rolling-asa		x	
counted-calls		x	
expected-wait		x	
ani			x
II-digits			x
service-hours	x		
For information about comparators that can be used with the condition statements, see goto step and goto vector command on page 219. A to Z and AA to ZZ vector variables and V1 to V9 VDN variables both need Basic Call Vectoring and Vectoring (variables). In addition, V1 to V9 VDN variables need Call Center Software 3.0 or later.			

The following vector example includes several **goto** commands that use conditional branching:

Conditional branching example

```
1. goto vector 100 if time-of-day is all 17:00 to all 8:00
2. goto vector 200 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 seconds hearing ringback
7. goto step 5 if unconditionally
8. busy
```

In the above example, conditional branch test statements are used in steps 1 through 3. If the call is placed during non business hours, the **goto vector** command in Step 1 routes the call to vector 100, but if the call is placed during business hours, control is passed to step 2.

In step 2, the **goto vector** command tests whether the call is placed during the weekend. If the test outcome is true, the call is routed to vector 200. Otherwise, control is passed to step 3.

In step 3, a **goto step** command tests for the number of calls that are queued to the main split. If the number of calls is greater than five, control is passed to busy in step 8. If the number of calls is five or less, vector processing continues at step 4, which queues the call to split 1. Finally, steps, 5 through 7 specify an announcement-wait cycle until an agent answers the call or the call is abandoned.

Time adjustments using goto conditionals

Use the following table to decide which **goto…if** conditional to use for time adjustments.

Conditional	Use to check for the following time adjustments
`goto ...if service-hours`	Uses the time adjustments from the Service Hours Table screen.
`goto ...if time-of-day`	Uses the time adjustments from the **VDN Timezone Offset** and **DST** fields on the VDN screen.
`goto ...if holiday`	Does not use time adjustments. The system time clock, as defined for the main server, is used without modification.

A time in the table from the first second of the start time, for example, 08:00:00. Also, the time is also in the table until the last second of the end time, for example, 17:00:59.

Comparing none, #, and numeric digits

How comparisons worked before vector variables

Prior to the introduction of Communication Manager 3.0, goto comparison tests using keywords such as none, or # as a threshold value were supported for only the = or <> comparators. For example:

- goto step 5 if digits = none
- goto step 5 if digits <> #

You cannot enter any other comparators with these keywords.

How comparisons work now

With vector variables and VDN variables, goto test comparisons against or containing the keywords none or # are allowed with all comparators including <, >, <=, or >=. These keywords can be compared against digit strings. When Communication Manager tests these comparisons, the keywords and digits have weighted values ordered as follows:

```
none < # < 0 < 1 to 9 < 00…
```

All comparisons are basically string comparisons, not numeric comparisons. A string comparison of 0 is less than 00, and not equal as in a numeric comparison.

With the introduction of Communication Manager 3.0, it is now possible to do less than or greater than comparisons with variables which can have a value of none (empty string) or # (invalid result or a single # digit was collected) using the ordering rules above. For example:

```
goto step 5 if digits = A

goto step 5 if digits <> A

goto step 5 if digits < A

goto step 5 if digits > A

goto step 5 if digits <= A

goto step 5 if digits => A
```

Using these properties, you can determine if a caller has entered a digit between 1 to 9 as follows:

```
1. collect 1 digit after hearing announcement x for A
2. goto step 1 if A <= 0 [will branch to step 1 if A has a value of none, # or 0]
3. [this step reached if A contains a digit between 1 to 9]
```

Comparisons still not allowed

You cannot use a comparison of the digits buffer that contains `none` or`#` against a specific numeric value that is not a variable. The goto test will always fail and fall through to the next step. For example:

```
2. goto step 1 if digits <= 0 [will branch to step 1 only if digits contains a 0]
3. …  [this step reached if digits contains none, # or a digit between 1 to 9]
```

You cannot directly enter `none` or `#` as a threshold value with comparators other than = or <>.

Media gateway, port network, and server vector conditionals

Description of conditionals

You can use any of three registered and unregistered vector conditionals with the **`goto step`** or **`goto vector`** commands to set up alternate routing or treatment of calls. These three conditionals test which type of server is processing the vector. These conditionals also test the registration status of media gateways and port networks connected with that server. The three conditionals are as follows:

- media-gateway - monitors the H.248 Media Gateway registration status
- port-network - monitors the port network gateway registration status
- server - monitors the type of server currently processing the vector step for the call

These conditionals allow alternate routing or treatment of calls based on the current status of the server processing a call, such as:

- The H.248 Media or Port Network Gateway is not registered with the Media Server processing the call
- A backup server is processing the call in *survivable* mode due to a failure of IP connectivity.

Reason to use the media gateway, port network, and server vector conditionals

These conditionals allow you to monitor the Communication Manager when it is running in a *survivable* configuration. Based on that knowledge, you can use alternative call handling or resources. For example, you can use different announcements, Interactive Voice Response systems (IVRs), or different skills to provide the best possible call handling with the available resources.

Syntax of gateway conditionals

The syntax of the gateway conditionals is as follows:

```
goto step [1-99] if media-gateway
 [1-x, all, any] [=, <>] registeredgoto step [1-99] if port-network
 [1-x, all, any] [=, <>] registeredgoto vector [1-99] @step
[1-99] if media-gateway
 [1-x, all, any] [=, <>] registeredgoto vector [1-99] @step
 [1-99] if port-network
 [1-x, all, any] [=, <>] registered
```

Parameter or condition	Description
media-gateway	Refers to a H.248 media gateway.
port-network	Refers to a port network gateway.
x	Refers to the number of gateways supported by the installed server platform.
all	Returns true if all of the equipped gateways or port networks meet the specified condition.
any	Returns true if any of the gateways or port networks meet the specified condition.
registered	Refers to the connection with the Communication Manager server currently processing the vector step for the call.
= registered	Returns true if the specified gateway is registered with the server.
<> registered	Returns true if the specified gateway is not registered with the server processing the vector step.

When gateways are not equipped

If the specified gateway number or gateway type is not administered, the test fails and logs a vector event. Vector processing continues at the next step in the vector.

Syntax of server conditionals

The syntax of the server conditionals is as follows:

```
goto step [1-99] if server
 [=, <>]
[main, ess, lsp]
goto vector [1-99] @step [1-99] if server [=, <>]
[main, ess, lsp]
```

Parameter	Description
server	The server currently processing the vector step for the call
main	The main or primary server for the network or switch configuration
ess	An Enterprise Survivable Server (ESS) as a backup server. The S8500 is an example of an ESS.
lsp	A Local Survivable Processor (LSP) that has been activated to act as a backup server for media gateway control. The S8300 is an example of an LSP.

Example 1

Use the following example to change queue-to skill from 20 to 30 if the server is the LSP.

```
1. go to step 4 if server = lsp
2. queue-to skill 20 pri 1
3. goto step 5 unconditionally
4. queue-to skill 30 pri 1
5. wait-time 10 secs hearing ringback
6. announcement 1000
7. wait-time 60 secs hearing music
8. goto step 6 unconditionally
```

Example 2

Use the following example to bypass the VRU if port network 5 is not registered. In this example, the VRU ports terminate on port network 5.

```
1. wait-time 0 secs hearing ringback
2. goto step 6 if port-network 5 <> registered
3. converse-on skill 50 pri 1 passing vdn and ani
4. collect 7 digits after announcement none
5. route-to digits
6. queue-to skill 25 pri 1
7. wait-time 10 secs hearing ringback
8. ...
```

goto step and goto vector command feature interactions

For BSR and LAI, the **goto** command is treated as a neutral vector command. When a call experiences LAI, the *ani* value is sent along with the call for ISDN PRI calls only. *ANI* is not sent for internal or DCS calls.

goto step and goto vector command interactions with CMS/BCMS

The **goto step** command is not tracked by CMS or BCMS.

The ANI and/or II-digits are passed to the CMS when the call first starts vector processing if the following conditions are true:

- Basic Call Vectoring and/or Call Prompting is optioned
- ANI is available from the network, the call is internal, or is received over DCS
- Information-Indicator Digits (II-digits) is available from the network
- The CMS is R3 (R3V5 was the first version to support II-digits) or a newer version

ANI and II-digits are not passed to BCMS.

The **goto vector** command is tracked by CMS. The following database items are created.

goto vector command		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Vector Flow Out	
GOTOCALLS/ GOTOTIME		
INTIME	Avg Time In Vector	

Table continues…

`goto vector` command		
Database item	**Report heading**	**Notes**
INFLOWCALLS	Vector Flow In	New Vector

messaging command

Purpose

With the **`messaging split or skill`** command, callers can leave a message for the specified extension, the active or latest VDN extension (default).

Syntax and valid entries

`messaging`	skill	hunt group (A valid hunt group is an ACD split or skill or a non-ACD hunt group assigned for AUDIX, remote AUDIX, MSA, or QSIG MWI on the hunt group.) VDN skill: 1st 2nd 3rd	for	extension number latest active A-Z, AA-ZZ V1–V9
	split	hunt group		

Requirements

The split or skill involved must be a messaging system split or skill, a remote messaging system split or skill.

Operation

This command causes the caller to be connected to the messaging system or message center split or skill so the caller can leave a message for the specified extension, that is, call answering service or mail.

If the split or skill number specified in the command is a valid message service split or skill such as a messaging system, 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 or skill for call answering service.

If the call is queued to the message service split or skill, or if the call terminates to an available message service agent or a messaging system 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:

- The split or skill queue is full.
- The messaging system link is down.
- All messaging system voice ports are out of service.

- The message service split or skill is DCS-AUDIX and all DCS trunks are busy.

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 or supervisor stations can be equipped with Automatic Message Waiting (AMW) lamps to accommodate the mail specified in the **`messaging split/skill`** command. The lamps can be assigned for VDNs or extensions used to access the messaging split or skill 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, the caller receives ringback until the caller disconnects.

Using a messaging step in a vector

If the extension is a VDN, and the skill group is a QSIG Message Waiting Indicator (MWI) hunt group, the messaging step in a vector is supported in Communication Manager 2.0 load 205 or later.

Example: 01 messaging skill 1 for extension 6000

In this example, skill 1 is a QSIG MWI hunt group. When a call is made to this hunt group, the call correctly routes to the mailbox of extension 6000. The SETUP message that is sent out on the QSIG trunk will correctly have 6000 as the original-called number and redirecting number.

Leaving a recorded message

The following example shows how the **`messaging split`** command allows callers to leave messages when agents are not available.

Leaving a recorded message

```
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 on 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
```

In step 1 of the example vector shown above, the **goto step** command tests whether the current time of day is outside of defined business hours. If the test outcome is true, vector processing branches to step 8.

Step 8 provides an announcement that offers callers the option to leave a recorded message, and vector processing continues with step 9, which proceeds unconditionally to step 11.

If the caller has not abandoned the call, the **messaging split** command in step 11 is executed. In this example, split 18 is an AUDIX split.

Note:

If initial vector processing went to step 2, but split 47 cannot take the call, vector processing branches to step 10, which also leads to the **messaging split** command in step 11. In this example, extension 2000 specifies the audix mailbox for split 47.

If the **messaging split** command in step 11 attempts to connect the caller to AUDIX but split queue is full or the AUDIX link is not in operation, termination to AUDIX is unsuccessful and vector processing continues with step 12, which provides an announcement for callers to try again during regular business hours.

Answer supervision considerations for the messaging command

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).

messaging command feature interactions

Interaction	Description
Messaging-system hunt group	The command can use a messaging system hunt group.
Command specifies a specific mailbox extension	If the command specifies a mailbox extension, the original principal for a call covered by a VDN is not passed to the adjunct and does not appear in the display to the answering agent. The specified extension appears in the display.
Command accessed using a direct call to the VDN	If the command is accessed using 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-system adjunct. Additionally, if the call is sent to a switch message service split or skill, the associated VDN name is sent to the messaging-system adjunct.
Command specifies active or latest as the mailbox extension	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-system adjunct and these subsequently appear in the display to the answering agent.
Mixed-length numbering plans	The messaging system does not support mixed-length numbering plans.

Table continues…

Interaction	Description
Command leaves a message for a VDN	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.
LAI	For LAI, the command can be treated as either a call acceptance vector command or a neutral vector command.
`Call Acceptance Vector` command	The command is treated as a call acceptance vector command when one of the following is true: • Call terminates to an agent or to a messaging-system port. • Call queues to a messaging split or skill.
`Neutral Vector` command	The command is treated as a neutral vector command whenever the command fails.
Messaging step in a vector	If the extension is a VDN and the skill group is a QSIG Message Waiting Indicator (MWI) hunt group, the messaging step in a vector does not work prior to Communication Manager 2.0 load 205.

messaging command interactions with CMS

When a queued call successfully goes to the messaging split, OUTFLOWCALLS/OUTFLOWTIME (1st split/skill) and DEQUECALLS/DEQUETIME (2nd/3rd splits [skills]) are tracked in the split/skill tables. The calls are reported as split/skill Flow Out, Dequeued Calls, and Dequeued Avg Queue Time.

Calls that queue using a **`messaging split/skill`** command are tracked as CALLSOFFERRED and LOWCALLS (no priority) or MEDCALLS (priority). The calls are shown in the standard reports according to the final disposition of the call.

The presence of the command in a vector enables the calls serviced by the vector to be vector-directed. When such a call is answered by an agent, the call is tracked as ACDCALLS/ANSTIME, and it is reported as ACD Calls, Split/Skill ACD Calls, and Avg Speed Ans.

Finally, if the command directs a call to a split/skill, the BACKUPCALLS database item is incremented, and the call is reported as Calls Ans in Backup and Calls Handled/Backup. The Calls Ans in Main report item is calculated by using the algorithm ACDCALLS - BACKUPCALLS.

A call abandoned after the command routes the call to a station or to an attendant is tracked as ABNCALLS/ABNTIME for the messaging split/skill and in the VDN/vector tables.

messaging command interactions with BCMS

A call advanced to another position using the **`messaging`** command is tracked as an outflow in the VDN report.

queue-to command

Purpose

The **queue-to** command queues calls to a split or skill, attendant group, attendant, or hunt group. If all agents or attendants are busy, the **queue-to** command assigns a priority level to the calls.

Syntax and valid entries

queue-to	attd-group This option is available with Attendant Vectoring.		
	attendant	`extension number`	
	best		
	hunt-group	`group number` A valid group number is a vector-controlled hunt group of any type such as Automatic Call Distribution (ACD) or Uniform Call Distribution (UCD).	priority (pri) • l=low • m=medium • h=high • t=top
	skill	Vector Directory Number (VDN) skills • **1st Skill** • **2nd Skill** • **3rd Skill**	
	split	hunt group	

Requirements

The split or skill must be vector-controlled.

Operation

A call sent with the **queue-to** command connects to an available agent or an attendant in the specified resource or enters the resource queue. When the call enters a queue, the **queue-to** command does not provide feedback to the caller. Other vector commands can provide wait treatment for calls in a queue.

With Attendant Vectoring, you can use the **wait-time 0 secs hearing ringback** step to provide immediate feedback to callers. The **queue-to** command does not provide ringback until the call rings at the attendant station. Do not use the **wait-time** step as the first vector step or as the step immediately before a **queue-to** step.

With singlesite Best Service Routing (BSR), the **queue-to best** command queues a call to a local split or local skill that the **consider** series finds as the best resource.

With multisite BSR, the best resource can be at a remote location. The **queue-to best** command then forwards the call to the interflow VDN for that location. You can administer the interflow VDN on the BSR Application Plan screen.

The system can queue a call to up to three local splits or local skills. A call remains in the queue until vector processing terminates, the caller drops or abandons the call, or the call reaches the

station of an agent. The **disconnect**, **busy**, or **route-to** command can terminate vector processing.

When an agent becomes available in a split or skill to which the system queues the call, the following occurs:

- The call rings at the agent station.
- The system removes the call from other queues.
- The system stops vector processing.

If the entered split or skill is one of the split or skill to which the call is already queued, the system queues the call again at a 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 the queue. Vector processing skips the step and moves to the next step for any of the following:

- Queue of the split or skill is full. A split or skill queue can be full if you administer a limit on the number of calls that the system can put in the queue. You can administer the **Queue Limit** field on the Hunt Group screen to a limit from `1` to `999` calls. The default option for the **Queue Limit** field is **unlimited**.
- Split or skill is not vector-controlled.
- Split or skill has no queue and no available agents. A split or skill has no queue if the **Queue** field on the Hunt Group is `n`.

  With Expert Agent Selection (EAS), all ACD groups have the field option as `y` in the following fields on the Hunt Group screen:

  - **ACD**
  - **Queue**
  - **Vector**

  Therefore, the system puts calls in the queue regardless of whether agents log in to the system.
- Call was previously queued to three splits or skills.

You can use a **route-to** command to another VDN to remove a call from a queue and to put the call in another queue.

The **queue-to best** command has operations and interactions similar to the **queue-to split/skill** command when the best resource is a local split or local skill. When the best resource is at a remote location, the **queue-to best** command functions as an unconditional **route-to** command with no coverage, performing Look-Ahead Interflow (LAI).

When vector processing executes a **queue-to best** command, the command initializes data for the best resource that the **consider** series finds for the call. If the **consider** series does not define best data, the system logs a vector event and vector processing moves to the next vector step.

A **consider** series might not produce the best data because of the following reasons:

- All resources that the **consider** series checks are unstaffed.
- No resource that the **consider** series checks has an open queue slot.

- Best data started before execution of the **reply-best** step because the status poll vector has no **consider** steps or because the vector contains a step that initializes best data.

If a queue attempt to a local resource fails, the system logs a vector event and vector processing moves to the next vector step. The command initializes the best data.

If an interflow attempt to a remote resource fails, the system logs a vector event and vector processing moves to the next vector step. If the **consider** series finds a local split or local skill as the best resource before an interflow attempt, the system queues the call to the local resource. The command initializes the best data and vector processing moves to the next step.

queue-to split command

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

queue-to split command considerations

- Keep split queues large to allow all incoming calls to be queued. If a queue is too small, a **queue-to split** or a **check split** command can fail to queue a call due to a lack of available queue slots and the call can be dropped.
- Include a vector step that tests a split queue before queuing calls and an alternate step that provides fallback treatment if the queue is full.
- When calls queue to backup splits, calls also remain in queue for any previous splits to which the calls were directed. When a split answers a call that is queued in multiple splits, the call is removed from all the other split queues.
- The **check split**, **queue-to split**, and **converse-on** commands can access only splits that are vector-controlled. A split is vector-controlled if you set the **Vector** field on the Hunt Group screen to `y`.
- When you set **EAS** to `y`, multiple *split* queuing changes to multiple *skill* queuing.

Multiple split queuing

The term “multiple split queuing” refers to the queuing of a call to more than one split at the same time. Incoming calls can be queued to a maximum of three ACD splits.

The following example vector shows this process.

Multiple split queuing example

```
1. goto step 4 if calls-queued in split 1 pri l >= 10
2. queue-to split 1 pri t
3. wait-time 12 seconds 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
```

In the example, vector step 1 tests whether the main split queue, which has 10 queue slots, is full and branches to one of the following vector steps. A low priority is specified to count calls in queue at all priority levels.

Note:

To prevent completion of vector processing without queuing the call to a split, check the split queue before queuing to the split. If the queue is full, provide an alternate treatment such as queuing to an alternate split. Use the check for available queue slots only if you have not used Dynamic Queue Slots for an entire Communication Manager instance.

If the main split queue is full, a **`goto`** command skips the main split and goes directly to step 4 to check backup splits. Otherwise, vector processing goes to step 2.

In step 2, a **`queue-to split`** command queues calls to split 1 at a top priority. Once the call is queued, vector processing continues with step 3.

Step 3 uses a **`wait-time`** command to specify 12 seconds of delay. If the call is not answered within 12 seconds, vector processing continues with step 4.

Step 4 contains a **`check split`** command that tests whether there are less than five calls queued to split 2.

- If the test outcome is true, the command attempts to connect the call to an agent in the split. If such a connection cannot be made, the command puts the call in the split queue at the specified priority level and vector processing continues with step 5.
- If the test outcome is false, the vector processing continues with step 5.

Step 5 contains another **`check split`** command that repeats the same process described for step 4, with the exception that the attempt to queue is now applied to split 3.

At this point in the vector process, if all previous attempts to direct the call to an available split do not succeed, steps 6, 7 and 8 are used to provide caller feedback and loop the call back to step 4 for additional attempts to connect to a split.

Option with VDN as the coverage point

You can use a VDN as the last point in a coverage path. The option allows calls to first go to a coverage and then be processed by Call Vectoring or Call Prompting. With the option, you can assign AUDIX to a vector-controlled hunt group and therefore enable access to the servers using a **`queue-to split`** or **`check split`** command.

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

Leaving recorded messages (VDN as the coverage point option)

```
VDN 1 (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 seconds 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 [audix split]
    9. wait-time 12 seconds hearing ringback
```

```
   10. announcement 1005 [“Please wait for an attendant to take your message.”]
   11. wait-time 50 seconds hearing music
   12. goto step 10 if unconditionally
   13. disconnect after announcement 1008 [“We cannot take a message now. Please call
back tomorrow.”]
```

In steps 3 and 8, the caller can choose to leave a recorded message, but the **queue-to split** command is used instead of the **messaging split** command. The call is actually queued to the AUDIX split.

However, a **messaging split** command does not queue the call to the split. Instead, if the command is successful, the caller is connected to the split so the caller can leave a message for the specified extension. However, termination to the split can be unsuccessful due to factors that cannot be checked by vector processing. For example, the AUDIX link does not function, or if all AUDIX ports are out of service.

As a result of the queuing process, you can include a wait-announcement loop after each **queue-to split** step and the appropriate loop can then 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.

Answer supervision considerations for the queue-to command

Answer supervision is returned when the call connects to an agent.

queue-to command feature interactions

The **queue-to** command can access a messaging system split or skill in cases where a VDN is assigned as a coverage point. To enable this function, you must assign the split or skill as a vector-controlled hunt group.

For BSR and LAI, the **queue-to** command can be treated either as a call acceptance vector command or a neutral vector command.

The **queue-to** command is treated as a call acceptance vector command when one of the following conditions is true:

- Call terminates to an agent.
- Call queues to a split or skill.
- BSR interflowed call is accepted at remote interflow vector.

The **queue-to** command is treated as a neutral vector command when the call neither terminates nor queues.

No COR checking is carried out when a **queue-to** step places a call to a split or skill.

queue-to command interactions with CMS

Calls queued using a **`queue-to split/skill`** command are tracked as CALLSOFFERRED and LOWCALLS/MEDCALLS/HIGHCALLS/TOPCALLS.

Split/skill calls are reported in the standard reports according to the final disposition of the call.

The presence of the command in a vector enables the calls that are serviced by the vector to be vector-directed. When such a call is answered by an agent, the call is tracked as ACDCALLS/ANSTIME, and it is reported as ACD Calls, Split/skill ACD Calls, and Avg Speed Ans. If the call is also queued to other splits/skills, OUTFLOWCALLS/OUTFLOWTIME is tracked in the first split/skill to which the call queues, and Flow Out is reported (unless the split/skill turns out to be the answering split/skill). DEQUECALLS/DEQUETIME is tracked in the second and third splits/skills if these splits/skills are not the answering split/skill, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time. However, if the second or third split/skill is the answering split/skill, INFLOWCALLS is tracked in the split/skill, and the call is reported as Flow In.

If the call abandons after the command queues the call to a split/skill, ABNCALLS/ABNTIME is tracked for the vector, the VDN, and the first split/skill to which the call is queued. The call is reported as Aban Call and Avg Aban Time. If the call is also queued to other splits/skills, DEQUECALLS/DEQUETIME is tracked in these splits/skills, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time.

BSR status poll calls are not counted as interflows. BSR interflows are now tracked as network interflowed calls (NETCALLS) by the CMS at the receiving switch. The CMS tracks a call's accumulated time-in-VDN as NETINTIME (that is, the NET_TIME value on the CMS at switch C combines the time a call has spent in VDNs at any previous locations, as communicated by information forwarding. The NETINTIME can be added to the time spent in the local switch to provide reports that include the total time the call has spent in the call center network (e.g., total ASA).

queue-to command interactions with BCMS

The total number of calls that are queued to the VDN using the **`queue-to`** 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 using the **`queue-to`** command as ACD calls to an agent is tracked as AVG SPEED ANS in the same report.

There is no added tracking for calls interflowed by BSR. BCMS tracks the calls as outflow in the VDN report.

reply-best command

Purpose

Use the **reply-best** command only in status poll vectors in multisite BSR applications, where the command returns best data for the location to the primary vector on the origin Communication Manager.

Syntax

```
reply-best
```

Note:

This **multisite BSR** command is available only when you activate the Virtual Routing feature.

Requirements

You must set **EAS** to y to use the **reply-best** command.

Operation

The purpose of the **reply-best** step is to return data for the best resource found by the **consider** series in a status poll vector to the primary vector in a multisite BSR application. The status poll vector executes in response to a call from a **consider** step in the primary vector. Each time the status poll vector executes, the **reply-best** step:

- Drops the incoming call without returning answer supervision.
- Returns status data to the primary vector using the ISDN DISCONNECT message.
- Clears the best data.
- Terminates processing in the status poll vector.

If the incoming call is not a trunk call, the **reply-best** command drops the call and logs a vector event. No status data is returned to the origin Communication Manager.

If the **consider** series yields no best data, the **reply-best** command drops the incoming call without returning answer supervision, terminates vector processing, and returns an infinite value for Expected Wait Time (EWT) in the DISCONNECT message. A **consider** series does not always produce best data due to any of the following reasons:

- All resources being checked are unstaffed.
- No resource being checked has an open queue slot.
- The best data is cleared before execution of the **reply-best** step, which can be because there are no **consider** steps in the status poll vector or because the vector contains a prior step that clears best data.

Answer supervision considerations for the reply-best command

The **reply-best** step does not return answer supervision.

reply-best command interactions with CMS/BCMS

Operation of the **reply-best** command is not reported or tracked by the CMS or by the BCMS.

return command

Purpose

The **goto vector** command can invoke a subroutine call. After the subroutine has processed, the **return** command returns vector processing to the step following the **goto vector** command.

Reason to use

When you use a subroutine, you need a command that returns vector processing to the calling vector.

Syntax

```
return
```

Operation

The subroutine return destination information for a **goto vector** command branch remains with the call until a **return** command is executed in a subsequent vector step, or until vector processing terminates for that call. Multiple return destinations, one for each **goto vector** command branch executed for the call, are stored for the call in Last In First Out (LIFO) order up to the limit of 8,000 or 400. When a return step is executed, the processing uses the most recent return destination for the call, which clears that return destination. A subsequent return step uses the next most recent return destination - and so on - until all return destinations for the call have been cleared.

The subroutine return destination information remains with the call through any subsequent vector processing, including subsequent goto vector commands. The exception is when a route-to number/digits to a VDN step is executed for the call or when vector processing ends for the call. When the route-to VDN step is executed, all subroutine return destinations stored for the call are cleared, and the call is removed from any queues. All return destinations for the call are also cleared when vector processing ends for the call.

When return destination information is not stored

If there is no subroutine return destination stored for the call when a return step is reached in vector processing, the return request is not processed and vector processing continues with the next step following the failed return step.

All data stored for the call remains with the call when the return command is executed. Also, the call remains in queue and continues to give any feedback, such as music.

Memory full conditions

An active subroutine call occurs when a goto vector command is executed. If the return destination space is full, the goto vector step still branches as determined by the conditional. When the return step reaches the branched-to vector, the following occurs:

- A *return destination memory full* vector event is generated
- Vector processing does not execute the return step and continues with the next step following this failed return step. If it is the last step, it is treated as a stop step.

route-to command

Purpose

Routes calls either to a destination specified by the 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 and valid entries

`route-to`	digits with coverage y or n				
	meet-me. The option is available only with meet-me conference vectors.				
	number	Up to 16 digits from 0 to 9 <digits> [A-Z, AA-ZZ, V1-V9] <digits>*<digits>A <digits>#<digits>A <digits>~p<digits>A <digits>~m<digits>A <digits>~s<digits>A <digits>~w<digits>A <digits>~W<digits>A ~r, ~r+ ~r*, ~r# *<digits>~L<digits>#	with coverage y, n	if digit >, >=, <>, =. or <=	0-9 #
				if interflow- qpos <= or >=	1-9
				unconditionally	
	name1, name2, or name3	with coverage y, n			

Table continues…

- **route to number**: Supports vector variables from A to Z or from AA to ZZ and VDN variables from V1 to V9. The variable value, in decimal digits, is defined elsewhere before the **route-to number** command is executed. Each variable whether a single or double character counts as two digits towards the maximum digits in the number field. The variable can be preceded by digits as long as the total is within the 16 digit or character position limit. The variable must always be the last entry and cannot be followed by a digit.
- <digits>: The notation <digits> means that more than one digit in the range of 0 to 9 can be inserted for the application.
- Pound (#) sign: The character is used in the threshold field to match a single pound (#) sign entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the "=" or "<>" comparators are valid.
- **route to name**: The parameter is available only with the Dial by Name feature.

Note:

- The **route-to** command supports service observing FACs, VDN observing by location, remote logout of agent FAC, remote access extension, attendant access number, and other destination numbers.
- The **route-to number** command is the destination and is entered in the number field. This field can contain an administration limit of a maximum of 16 decimal digits or combination of characters and numbers that total 16. Special notations such as "~p" with "a~" followed by a character are counted as two digits towards the 16.
- Use of a variable allows having a **route-to number** destination address of more than 16 digits since a variable can be assigned up 16 digits during processing and is combined with the entry in the number field.
- When the specified number is preceded by "~r", an NCR invocation is attempted back over the trunk group to the network service provider. The "~r" sequence is counted as two digit positions toward the 16 total. The "+" character is an indication for E.164 numbering required by some network service providers for NCR invocation over SIP trunks. The "+" character is counted as two digit positions towards the 16 total. The "~r" or "~r+" entries must be in the initial digit or character positions of the number field.
- By prefixing a VDN number with "~r*" or "~r#" in **route-to number** command, you can access an FAC, or a remote phone number over an NCR. Using the "*" prefix, you can also access a remote number for which you must dial "*9". For this, you must set up and call a VDN that includes 9 followed by the phone number. For example,

  ```
  route-to number *V1 if cov unconditionally
  ```

  command can route to an external number *913032451234 if *V1* is set up as 913032451234.

Operation

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

Operation details for the route-to command

You can use the **route-to** command with or without coverage.

The following table summarizes the operation of the **route-to** command for each destination type and the conditions with the commands.

Condition	cov = n any step	cov = y any step
Invalid destination	Go to the next step or stop vector processing	Go to the next step or stop vector processing
VDN extension		
Vector assigned	Go to a new vector	Go to a new vector
Vector has no steps	Stop vector processing Call remains in a queue and the caller continues to hear feedback. In the case where the **route-to** command fails and vector processing stops because of a busy station or a trunk group, you can implement retry in the vector. For retry, include an unconditional **goto** step as the last step for a loopback to the **route-to** command. To reduce processor occupancy, use an intermediate **wait-time** step with feedback and delay interval.	Stop vector processing
A call that Communication Manager routes to a VDN using **route-to number with cov = y unconditionally** functions similar to a directly dialed call instead of a VDN call. Therefore, the terminating station displays only the originating station information and does not show the VDN information. For other types of VDN calls, the terminating station displays the VDN name. When Communication Manager routes a call to a VDN, Communication Manager ends vector processing of the previous vector and removes the call from the queues that the previous vector establishes for the call.		
Station extension idle with all call appearances idle		
Call Forwarding (CF) ALL Active or CF direct agent calls	Forward the call, move to the next step, or stop vector processing	Forward the call, send the call to the coverage point, or play a busy tone
Coverage		
Direct agent calls	Ring an idle call appearance	Send the call to the coverage point
Extension all	Go to next step or stop vector processing	Send the call to the coverage point

Table continues…

Condition	cov = n any step	cov = y any step
Send All Calls (SAC)	Ring an idle call appearance	Send the call to the coverage point
None	Ring an idle call appearance	Deliver the call with coverage
Station extension active with idle two-way appearance		
CF-ALL active	Forward the call, move to the next step, or stop vector processing	Forward the call, send the call to the coverage point, or play a busy tone
Station extension busy with no idle two way appearance		
Extension in hunt group	Put the call in a queue, move to the next step, or stop vector processing	Put the call in a queue, send the call to the coverage point, or play a busy tone
CF-ALL Active or CF direct agent calls	Forward the call, move to the next step, or stop vector processing	Forward the call, send the call to the coverage point, or play a busy tone
Call waiting for analog station	Go to the next step or stop vector processing	Call continues to wait in a queue
Coverage		
Extension active	Go to the next step or stop vector processing	Send the call to the coverage point
Extension busy		Send the call to the coverage point
All		Send the call to the coverage point
Send All Calls (SAC)		Send the call to the coverage point
None, hunt, forward, or coverage destination is unavailable		Play a busy tone
Extension with incompatible Class of Restriction (COR)		Go to the next step or stop vector processing
Terminating Extension Group (TEG)		
All members are idle	Ring an idle call appearance	Deliver call with coverage
A member is active on TEG	Go to the next step or stop vector processing	Send the call to the coverage point or play a busy tone
No idle call appearance	Go to the next step or stop vector processing	Send the call to the coverage point or play a busy tone
Hunt group extension		
Idle agent	Ring an idle call appearance	Deliver call with coverage
No idle agent	• Cannot put the call in a queue: Go to the next step or stop vector processing	Play a busy tone Put the call in a queue

Table continues…

Condition	cov = n any step	cov = y any step
	• Put the call in a queue	
Extension on another node: Uniform Dialing Plan (UDP) DCS or non-DCS		
Trunk available	Deliver the call	Deliver the call
Trunk unavailable	Go to the next step or stop vector processing	Put the call in a queue or play a reorder tone
No DCS buffer for routing	Deliver the call with or without the DCS message	Deliver the call with or without the DCS message
Trunk Access Code (TAC) destination		
Trunk group no dial access	Go to the next step or stop vector processing	Route the call to a local attendant
Trunk available	Deliver the call	Deliver the call
Trunk unavailable	Go to the next step or stop vector processing	Put the call in a queue or play a reorder tone
Automatic Alternate Routing (AAR) or Alternate Route Selection (ARS) Feature Access Code (FAC) destination including subnet trunking		
Trunk group no dial access	Try the next route	Route the call to a local attendant
Trunk available	Deliver the call	Deliver the call
Other routes are available	Deliver the call	Try the next route
All routes are busy • No pattern queuing • Queuing assigned	Go to the next step or stop vector processing	• No Pattern Queuing: Play a reorder tone • Queuing assigned: Queue to pattern
Attendant queue (dial 0)		
Idle attendant	Ring an idle call appearance	Deliver the call with coverage
No idle attendant • Not in Night Service • In Night Service	Put the call in a queue	Put the call in a queue
Assigned destination	Deliver to Night Service	Deliver to Night Service
Unassigned destination	Put the call in a queue	Put the call in a queue
Individual attendant access		
Idle attendant	Ring an idle call appearance	Deliver the call with coverage
Busy attendant	Put the call in a queue, go to the next step, or stop vector processing	Put the call in a queue or play a busy tone
Centralized Attendant Service (CAS) with caller at a branch location		
Available Release Link Trunks (RLTs)	Ring an idle call appearance	Deliver the call with coverage
All RLTs busy	put the call in a queue, go to the next step, or stop vector processing	Put the call in a queue or play a busy tone

Table continues…

Condition	cov = n any step	cov = y any step
Inter-PBX attendant call		
Trunk group controlled	Route to local attendant	Route the call to a local attendant
Trunk available	Deliver the call	Deliver the call
Trunk unavailable	Go to the next step or stop vector processing	Play a reorder tone
Look-Ahead Interflow (LAI): Feature active and routes over ISDN-PRI with one exception. Any **`route-to with cov = y`** step that routes over ISDN-PRI cancels LAI. The exception occurs when a call reaches a vector with coverage to a VDN. Communication Manager does not forward calls but redirects calls that cover to a VDN. For covered calls, a **`route-to`** command with coverage functions as if the coverage field option is `n`. Therefore, a **`route-to with coverage = y`** routes covered calls with LAI over ISDN facilities if LAI is active.		
B-channel is available		
Bearer (B) channel unavailable	Go to the next step or stop vector processing	Put the call in a queue or play a reorder tone
• Accept call	Interflow succeeds. The originating Communication Manager server removes the call from any queue or feedback, such as music or ringback.	Call cut-through
• Reject call	Go to the next step or stop vector processing	Play a busy tone or disconnect the call
Receiving Communication Manager with LAI.		
• Accept call	Interflow succeeds	Call cut-through
• Reject call	Go to the next step at the receiving Communication Manager server or the originating Communication Manager server treats the call as rejected after a 2–minute time out	Play a busy tone or disconnect the call
• **`if interflow-qpos`**	Determine if the queued call is eligible for interflow	
Invalid destinations include the following: • Attendant Control of Trunk Group Access (ACTGA) destination • COR of the VDN. For example, origination restricted, Facility Restriction Level (FRL) of a VDN that is lower than that required for AAR or ARS pattern access • Empty, for example, zero collected digits or invalid route-to destination number • Incompatible calling and destination partitions • Incomplete number of digits for the AAR or ARS pattern • Maintenance busy station extension • Non AAR or ARS FAC		

Table continues…

Condition	cov = n any step	cov = y any step
• No routes assigned to the AAR or ARS pattern • Off-net forwarding destination • Unassigned extension number The collected digits must match a valid ARS analysis string if: • Call routing includes a Trunk Access Code (TAC) destination • The TAC is for a Central Office (CO) or a Foreign eXchange (FX) trunk with a **`route-to with coverage = n`** step If the collected digits do not match a valid ARS analysis string, Communication Manager treats the destination as invalid. For other trunk types with a **`route-to number`** or **`route-to digits with coverage = n`** step, the step succeeds, that is, Communication Manager ends vector processing when Communication Manager seizes the trunk. For a **`route-to with coverage = y`** step, the step succeeds if Communication Manager assigns the TAC.		

If you administer **coverage** as `y`, Communication Manager removes the call from vector processing when vector processing reaches the route-to step regardless of the facility or remote Communication Manager availability. Even if the destination is available, Communication Manager removes the call out of the queue and also removes any feedback, such as music or ringback.

If the receiving Communication Manager vector rejects the call, the busy command or forced disconnect command defines subsequent call treatment.

Communication Manager treats the call as if the destination is directly dialed. Treatment includes coverage, call forwarding, and incomplete call treatment, that is, busy, reorder, or intercept tones. The answering station sees only the caller name and the number, unless you administer **Display VDN for Route-To DAC** on the Vector Directory Number (VDN) screen as `y`.

Communication Manager treats a routed call with an **`adjunct routing link`** command in the same way as a routed call with a **`route-to with coverage = y`** command.

Conditional route-to statements

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.

Destinations for the route-to command

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

- Internal extension. For example, split or hunt group and station.
- VDN extension

- Attendant or attendant hunt group or queue
- Attendant access number
- Remote extension (UDP/DCS)
- External number such as a TAC or AAR/ARS FAC followed by a public or private network number. For example, 7-digit ETN and 10-digit DDD.
- Remote access extension.
- Service Observing (SO) FAC
- Another Avaya switch via Network Call Redirection (NCR)
- Remote Logout of Agent FAC
- Forced Agent Logout/AUX work By Location/Skill FACs

Note:

The settings of the Class of Restriction (COR) of the active VDN are used for calling permissions when routing via the **`route-to`** command. The route-to digits command can be used to implement an automated attendant function.

Command completion and failures

- The **`route-to digits`** command fails if no digits are collected. 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 coverage set to `n`, for a covered call regardless of the coverage setting.

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

- The endpoint is alerted.
- The endpoint has Call Forwarding or Night Service enabled and the night service destination is alerted.
- The endpoint has off-premises Call Forwarding, that is, UDP Hunt Night Service, enabled and a trunk is seized.

The system 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.

About the number field

If the number is a VDN extension

The following events occur:

- Vector processing terminates within the current vector and the call is removed from any queues.
- Any call-related data such as dial-ahead digits and collected digits remain with the call.
- If the current VDN is administered with override, the new VDN overrides current VDN information.
- Processing of the vector associated with the routed-to VDN extension begins.

If the number is an AAR/ARS FAC plus digits, or if it is a remote Uniform Dial Plan (UDP) extension

Standard Automatic Alternate Routing (AAR) / Automatic Route Selection (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 Facility Restriction Level (FRL) / Class of Restriction (COR) is restricted and vector processing continues at the next vector command.

If the number is a Trunk Access Code (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, etc.), and vector processing continues at the next vector command.

If the number is any other number, such as an FAC other than an AAR/ARS or Service Observing

The command is unsuccessful, and vector processing continues at the next vector command.

Abbreviated Dialing special characters

You can use the Abbreviated Dialing (AD) special characters in the number field when programming an AD list entry. Each of the following characters instructs the system to take a different action when dialing reaches the point where the character is stored. Each special character counts as two digits towards the maximum count of 16 digits.

- **`~p`** (pause): Delay the transmission of the digits following the character for 1.5 seconds.
- **`~w`** (wait): Wait for up to 30 seconds for the far end dial tone.
- **`~m`** (mark): Change the digits following the character to outpulse DTMF digits.
- **`~s`** (suppress): Suppress the display of digits following the character.

- **~W** (indefinite wait): Wait indefinitely for a far end dial tone. Use the “~W” character when the response time is more than 30 seconds.

You can use the following variables in the number field alone or in combination with special characters and preceding digits. Each variable whether a single or double character counts as two digits towards the maximum count.

- *A - Z* or *AA – ZZ* vector variables defined in the Variables For Vectors screen.
- *V1 - V9* VDN variables assigned to the active VDN for the call.

You can use another special character, “~L” (location) in the **route-to number** command when programming the VDN Observing by Location feature. The location character indicates that the digits following the character “~L” comprise the location ID number of the location to be observed. You can use up to three digits to indicate the location ID number.

Using the route-to command for NCR

You can use variables with the ~r special character in the **route-to** number command to activate Network Call Redirection (NCR). The digits following the invocation character are the Service Provider network redirect to phone number address without inclusion of an internal access code. Use the any of the following formats:

- ~r<*number up to 14 digits*> - This option allows you to enter a specific number. For example, ~r13035552345. This format can contain only up to 14 digits for the redirection address because ~r takes up 2 digit spaces of the 16-digit number field.
- ~r*[A-Z or AA-ZZ]* - This option allows you to enter a vector variable as the redirect to address. For example, ~rA. The variable can have a value up to 16 digits during processing since it is not counted towards the number field 16 digit limit during administration. The variable letter(s) can have preceding digits following the ~r. For example ~r123AB. The variable character(s) always count as two digits towards the 16 digit number field limit.
- ~r*[V1-V9]* - This option allows you to enter a VDN variable. For example ~rV1. This variable can also have a value up to 16 digits and can have preceding digits following the ~r. The variable always counts as two digits towards the 16 digit number field limit.
- ~r+<*number up to 12 digits*> - This option allows you to enter a specific number for the special case where a network Service Provider requires the + character to indicate E.164 numbering for NCR invocation over a SIP trunking interface with the network. For example, ~r +305558754. This format can contain only up to 10 digits for the redirection address because ~r and + both take up 2 digit spaces each for a total of 4 spaces of the 16-digit number field.
- ~r+*[A-Z or AA-ZZ]* - This option allows you to enter a vector variable as the E.164 numbered redirection address. For example, ~r+A. The variable can have a value up to 16 digits during processing since it is not counted towards the number field 16 digit limit during administration. The variable letter(s) can have preceding digits following the ~r+. For example ~r+123AB. The variable character(s) always count as two digits towards the 16 digit number field limit.
- ~r+*[V1-V9]* - This option allows you to enter a VDN variable as the E.164 numbered redirection address. For example ~r+V1. This variable can also have a value up to 16 digits and can have

preceding digits following the ~r+. The variable always counts as two digits towards the 16 digit number field limit.

For examples, see *Using route-to number ~r vector step to activate NCR* in *Avaya Aura® Call Center Elite Feature Reference*.

Coverage parameter

The optional coverage parameter determines if coverage must 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.

route-to number command

The **`route-to number`** command is used to route calls to a vector-programmed number.

About interflow routing

Calls can be routed to a programmed number using a process known as Interflow.

Interflow allows calls, directed to a split, to be redirected to an internal or an external destination. For Basic Call Vectoring, the destination is represented by a number programmed in the vector. The number must be provided in the **`route-to number`** command and is associated with one of the following destination types:

- Attendant or attendant queue
- Local extension
- Remote (UDP) extension
- External number
- VDN
- Feature Access Code
- Remote phone number for which you must dial *9 as prefix

Interflow Routing considerations

- Do not allow calls to interflow back and forth between vectors on remote servers and local servers. This process can cause a single call to use up all available trunks.
- When the **`route-to number`** command is used to chain multiple vectors for enhanced processing capabilities, the following events occur:

1. Vector processing begins at the first step in the vector assigned to the routed-to VDN.

2. The call is removed from any queues to which the call was previously assigned.
3. Any previously assigned wait treatment is disabled.
4. Processing continues in the receiving vector at step 1.

Call interflow example

```
VDN (extension=1000   name=``Billing Service''  vector=55)
Vector 55:
      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 seconds 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=100)
  Vector 100:
      1. announcement 3900 [“All our agents are busy. Please leave a message.”]
      2. messaging split 18 for extension 3000
      3. disconnect after announcement 2505 [“Please call back tomorrow.”]
```

In the example, vector 55 provides a series of initial vector steps that test the queue status for split 1. Based on the outcome of the tests, the call is connected to split 1 or vector processing branches to step 8.

In step 8 a **`route-to number`** command specifies extension number 2020, which is a VDN assigned to vector 100. When the **`route-to number`** command is executed, vector processing in vector 55 terminates, the call is removed from the split 1 queue and vector processing continues with step 1 in vector 100.

When control is passed to the second vector, step 1 provides the caller with an appropriate announcement and then step 2 executes a **`messaging split`** command that attempts to queue the call to the message service split or else terminate the call to either a message service agent or AUDIX voice port. If either of the attempts succeeds, the caller can leave a message. If none of the attempts succeed, the command fails and vector processing continues at the next vector step.

Tip:

Use an announcement to inform the caller that the messaging connection was unsuccessful.

Service Observing routing

When the Service Observing feature is enabled, **`route-to number`** commands can be used to allow call monitoring from a local station or other remote location. The following example shows a vector that connects a call to a Service Observing FAC.

Important:

The following example does not provide security checks and must be used only in situations where security is not a concern.

Vector for Service Observing FAC

```
1. wait-time 0 secs hearing ringback
2. route-to number #12 with cov n if unconditionally (Listen-only FAC)
3. busy
```

In the example, the caller is connected to a listen-only Service Observing FAC. Once connected, the person who is service observing must dial the extension number to be observed. To observe in a listen or talk mode, the observer must dial a different VDN.

Answer supervision considerations for the route-to command

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.

route-to command feature interactions

Interaction	Description
Attendant queue	A call that the **route-to** command processes can wait in an attendant queue and Communication Manager removes the call from vector processing. The **route-to** command can gain access to public and private networks. If the **route-to** command dials the attendant and if the system is in Night Service, Communication Manager routes the call to the Direct Inward Dialing (DID) Listed Directory Number (LDN) night destination. You can use the extension number that you assign to an attendant console as the command argument.
Automatic Alternate Routing (AAR) or Automatic Route Selection (ARS)	The **route-to** command can specify AAR or ARS access codes. The **route-to** command can make AAR or ARS calls that implement subnet trunking, which is call routing over trunk groups that end in Communication Manager with different dial plans.
Authorization codes	Communication Manager disables authorization codes for routing calls through VDNs. The **route-to** command fails and Communication Manager does not prompt for an authorization code in the following scenarios: • Authorization codes are active for the system. • A **route-to** command in a prompting vector gains access to AAR or ARS. • The Facility Restriction Level (FRL) of the VDN does not have permissions to use the chosen routing preference.
Bridged appearance	If the destination of a **route-to** command is a station with bridged appearances, Communication Manager updates the button lamps associated with the bridged appearance.
Class of Restriction (COR)	When you apply COR checking to a **route-to number** or **route-to digits** step, Communication Manager uses the COR number of the latest

Table continues…

Interaction	Description
	VDN. Communication Manager uses the COR number to determine the time-of-day (tod) routing chart of the Partitioned Group Number (PGN). This PGN determines the route tables that Communication Manager must use on a call.
Coverage	For a call that covers or forwards to a VDN, the **`route-to with coverage y`** command functions similar to the **`route-to with coverage n`** command. For a covered or forwarded call, Communication Manager disables the coverage option because Communication Manager cannot redirect such a call further. Communication Manager forwards a **`route-to with coverage y`** command to a station that has call forwarding active. When a **`route-to with coverage n`** command initiates a call over ISDN-PRI facilities and **LAI** is **y**, Communication Manager treats the call on a Look-ahead basis. However, if you use the **`route-to with coverage y`** command, Communication Manager interflows the call unconditionally. **Note:** If the **`route-to`** command routes calls to a display station without coverage, the station displays the following: a = Originator Name to VDN Name.
Direct Outward Dialing (DOD)	If vector processing executes the **`route-to`** command and DOD is in effect, Communication Manager compares the COR numbers of the latest VDN and the called facility to determine whether Communication Manager can forward the call. If access is not permitted, the **`route-to`** command fails and vector processing continues to the next step. The route-to command can also fail in the following scenario: • When you assign a VDN with a COR that requires access codes.
Look-ahead Interflow (LAI)	For LAI, Communication Manager treats the **`route-to`** command as a call acceptance vector command or as a neutral vector command. Communication Manager treats the command as a call acceptance vector command when one of the following is true: • Command terminates to a valid local destination. • Command seizes a non-PRI trunk. • Command execution results in an LAI call attempt and the Communication Manager server at the far end accepts the call. Communication Manager treats the command as a neutral vector command when one of the following is true: • Termination is unsuccessful. • Trunk is not seized. • The Communication Manager server at the far end denies the LAI call attempt.

Table continues…

Interaction	Description
Messaging system	The **route-to** command can call a messaging system extension. If this happens, Communication Manager treats the call as a direct call to the messaging system, and the calling party can retrieve messages. If the call covers to a VDN, the **route-to** command supports a remote messaging system interface to a local hunt group extension that is assigned as a remote messaging system hunt group. This hunt group forwards the call to the destination in a manner similar to when you assign the hunt group as a coverage point. Communication Manager treats the DCS link down condition for a call that covers to a VDN as a direct call to the messaging system. **Note:** The hunt group of the remote messaging system has no members and is not vector-controlled.
Pickup group	If the **route-to** command calls a station that is a member of a Pickup group, any member of the group can receive the call.
Service Observing	You can use the **route-to** command to start Service Observing. For more information, see *Avaya Aura® Call Center Elite Feature Reference*.
Tenant Partitioning	If you use Tenant Partitioning, Communication Manager checks the TN of the Vector Directory Number (VDN) during call processing. Communication Manager then determines whether to deliver calls to the route-to destination. Communication Manager does not route calls to the destination in the following scenario: • The TN matches, but the Facility Restriction Level (FRL) of the Class of Restriction (COR) assigned to the VDN has a value that is lower than the value of the destination FRL, Instead, Communication Manager logs a denial event and vector processing moves to the next vector step.

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

route-to command interactions with CMS

`Route-to` station or attendant			
Database item	**Report heading**	**Notes**	
• OutflowCalls • OutflowTime	• Flow out • Vector flow out • DequeueCalls or DequeueTime	First split	
		• Dequeued calls • Dequeued average queue time	Second or third split
InTime	DequeueCalls or DequeueTime	Average time in vectors	
• ConnectCalls • ConnectTime	Other calls connect	Answered calls on G3	

Route-to trunk			
Database item	**Report heading**	**Notes**	
• OutflowCalls • OutflowTime	• Flow out • Vector flow out • VDN flow out	First split	
DequeueCalls or DequeueTime	• Dequeued calls • Dequeued average queue time	Second or third split	

Route-to VDN			
Database item	**Report heading**	**Notes**	
• OutflowCalls • OutflowTime	• Flow out • Vector flow out • VDN flow out	First split	
	DequeueCalls or DequeueTime	• Dequeued calls • Dequeued average queue time	Second or third split
	InTime	Average time in vector	
InflowCalls	• Vector flow in • VDN flow in	• New vector • New VDN	
• InterflowCalls • InterflowTime	VDN flow interflow	-	

Route-to split or hunt group		
Database item	**Report heading**	**Notes**
• OutflowCalls • OutflowTime	Flow out	First split
DequeueCalls or DequeueTime	• Dequeued calls • Dequeued average queue time	Second or third split
InTime	Average time in vector	—
CallsOffered	—	New split
• MedCalls • HighCalls	—	• No priority • Priority

route-to command interactions with BCMS

A call advanced to another position using the **route-to** command is tracked as outflow in the VDN report. A call answered by an attendant using the command is also tracked as outflow.

There is no added tracking for calls interflowed by BSR. BCMS tracks these calls as outflow in the VDN report.

set command

Use the **set** vector command to perform the following tasks:

- Perform numeric and digit string operations
- Assign values to a user-assignable vector variable or to the digits buffer during vector processing

You can control the call flow through the vectors based on specific circumstances for individual calls. The **set** vector step allows the following types of variable entries:

- A to Z and AA to ZZ user-assigned local or global collect vector variables
- A to Z and AA to ZZ system-assigned vector variables, for example, ani, asaiuui, and doy
- V1 to V9 VDN variable types
- A directly-entered numeric value
- The collected digits buffer where digits from the caller are stored

Reason to use

This command adds powerful and flexible programming functionality to vector processing because all other commands allow you to use only fixed values. This command allows you to manipulate variables using mathematics and digit operators.

Syntax and valid entries

The basic syntax of the **set** command is:

set [vector variable, Digits] = [operand1] [operator] [operand2]

set	user-assigned type (Only global or local collect type vector variables can be assigned using the set command.) A-Z or AA-ZZ	=	user-assigned type A-Z or AA-ZZ	ADD, SUB, MUL, DIV, CATL, CATR, MOD10, or SEL	user-assigned type A-Z or AA-ZZ
	asaiuui A-Z or AA-ZZ		system-assigned A-Z or AA-ZZ		system-assigned A-Z or AA-ZZ
					directly-entered numeric string (Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.)
			V1-V9		V1-V9
	digits (The collected digits buffer holds up to 16 digits.)		digits		digits
			none		none

Variable, digits buffer, and asaiuui

Variable

You can enter user-assigned A to Z and AA to ZZ collect vector or the asaiuui variable types in the **Variable** field. The collect vector variable can be either local or global.

Note:

You cannot use the system-assigned A to Z and AA to ZZ vector variables in this field, except for asaiuui.

digits

A digits buffer is associated with each call. This buffer can be populated by a **collect** command execution, a **set** command assignment to “digits” [set digits = ...], or when the Adjunct Switch Application Interface (ASAI) sends “collected digits”.

The buffer is a storage location in the software associated with the caller that holds the digits that have been collected.

Once populated, the digits buffer:

- Can be sent over the ASAI in event messaging such as adjunct route

- Forwards with the call in shared User-to-User Information (UUI)
- Can be passed with the **`converse-on`** command as data
- Displays the number to the agent
- Is sent to the reporting adjunct such as CMS in a message when the assignment is complete
- Used to route calls using the **`route-digits`** vector command
- Does not include dial-ahead digits

Assign to asaiuui variable type

Use the **`set`** command to assign a value to a defined vector variable and to replace or append the value in the ASAI UUI string associated with a call. The replace or append operation to the stored ASAI UUI digits is based on the start and length parameters defined for the *asaiuui* type vector variable.

The capability is available only when Call Center is upgraded to Release 4.0 or later and both **Vectoring (Variables)** and **Vectoring (3.0 Enhanced)** are enabled.

The rules for determining the value of a **`set`** command are similar to those for the operation of a digits type variable assignment. The assignment of the resultant value to an *asaiuui* type variable is different from the assignment to a *collect* type variable. The defined start and length of the *asaiuui* variable is applied with either a string or an arithmetic operation. See “Rules” for detailed information.

When the ASAI UUI string for a call is changed via the set operation, the changed string:

- Is sent to the reporting adjunct for inclusion in the call record.
- Is passed by a subsequent event report to an ASAI adjunct in an ASAI IE.
- Is passed by an LAI/BSR interflow.
- A DIGITS type 5 message with the changed string is sent to the connected reporting adjuncts.

You can use the vectoring information to make ASAI routing decisions or to provide adjunct display information to an agent.

For example, you can choose to:

- Assign a higher priority status to calls waiting at both remote and local call centers beyond a certain duration.
- Provide additional information that can be obtained during vector processing to an ASAI connected adjunct.
- Provide additional information to forward with the call.

Rules

- When a **set** command assigns a value to a defined *asaiuui* vector variable, the **`set`** command operation replaces or appends digits in the ASAI UUI string assiociated with a call as defined by the start and length definition for the vector variable. This is also true for empty ASAI UUI strings.

Note:

You can remove digits from the ASAI UUI string by replacing the digits with zeros. This method is effective for removing proprietary or private information from the string.

- The start digit position defines the point in the ASAI UUI string where the resultant value digit string begins.
- The length parameter defines the number of digits from the resultant value digit string to place into the ASAI UUI string.

Example: Demonstrating start and length parameters

```
The current ASAI UUI for a call is 15723924459
Define A as asaiuui type with start=4 and length=5
If the set operation result is 85670
The ASAI UUI string after execution will be 15785670

459
```

- The ASAI UUI string associated with the call can be up to 96 digits.
- The ASAI UUI string is the user data portion of ASAI UUI IE or ASAI UUI portion of the shared UUI, which contains a total of 99 bytes. 2 bytes for the header (op code and the data length) plus a protocol discriminator byte and a maximum of 96 bytes of actual user data. The data length byte value includes the protocol discriminator byte plus the actual user data bytes.
- The protocol discriminator is unusually set 0x00 (user specified) or to 0x04 (indicating IA5 characters) by the adjunct and the setting is retained by Communication Manager. If the Call Vectoring set command initially creates the ASAI UUI data, the protocol discriminator is set to 0x04 by Communication Manager.
- The **`set`** command can only change the actual data bytes following the protocol discriminator. The assignment affects the set of bytes defined by the start and length parameters for the vector variable and is limited to the decimal digits (0-9) subset of the ASCII (IA5) character set.
- The ASAI user data sent to the reporting adjunct (Avaya CMS/IQ) contains only the actual user data bytes (decimal digits 0-9) without the protocol discriminator byte.
- Any digits already in the ASAI UUI string that are not in the range of the *asaiuui* type variable start and length definition are retained.
- The **`set`** command assignment to the *asaiuui* variable can process up to 16 digits at a time. Use multiple **`set`** command steps with different variable definitions to change more digits.

Example: Assigning 32 digits to a caller ASAI UUI string

```
Define A as asaiuui type with start=1, length=16
Define B as asaiuui type with start=17, length=16
V1 = Assigned VDN variable 1234567890123456
V2 = Assigned VDN variable 6543210987654321

set A = V1 SEL 16
set B = V2 SEL 16

ASAI UUI for caller = 12345678901234566543210987654321
```

- If the start position is greater than 1 and the ASAI UUI string is empty or nulls occur before the start position, the null digit positions ahead of the start position are padded with zeros.

Example: Start position preceded by nulls

```
The current ASAI UUI for a call is 145
Define A as asaiuui type with start=45 and length=3
```

```
If the set operation result is 86532
The ASAI UUI string after execution will be 1450865
```

- You can assign a single # character in the ASAI UUI string to use as a delimiter when the definition of the vector variable is length = 1. Use a **`collect`** command step to assign the # character to a *collect* type vector variable that can be assigned to the *asaiuui* type variable. Another way is to use A = none DIV 0 to put a # character in the string.

  If the length definition of the *asaiuui* type variable is greater than 1, the assignment operation will fail as described in “Invalid results”.

Example: Assigning a # character to the fifth digit position using divide by 0

```
Define A as asaiuui type with start=5, length=1
ASAI UUI for caller = 1234567890

set A = none DIV 0

ASAI UUI for caller = 1234#67890
```

- If the result of a set string operation, such as CATL, CATR, or SEL is greater than 16 digits, you can use the *asaiuui* variable definition to truncate the right end of the string to the required number of digits.

Example: Truncating result to 16 digits

```
Define A as asaiuui type with start=1, length=16“digits” (from the collect step) =
1234567890

set A = digits CATR 1234567890

ASAI UUI for caller = 1234567890123456
```

- UUI can be transported betweenCommunication Manager systems using what is called *Service Provider* format or *Shared* format. The Service Provider format only carries a single data element usually containing ASAI/CTI user data associated with the call. The Service Provider UUI IE contains an op code (0x7E), a data length byte and a protocol discriminator byte. This is then followed by up to 96 data bytes. The Shared format is a multi-data element format that can contain UCID, ASAI user data, collected digits, VDN name, etc. associated with the call. Each data element is designated by a unique op code defined by the Avaya shared UUI specification. The ASAI user portion of the shared UUI consists of the op code (0xC8) and the data length byte followed by the user data.
- With Service Provider format the protocol discriminator is unusually set to 0x00 (indicating that the following data is in a user specified format) or to 0x04 (indicating IA5 ASCII characters) by the adjunct and the setting is retained by Communication Manager.
- The protocol discriminator for the shared UUI format is usually set to 0x00 (user specified), however the protocol can be set to 0x04 which indicates IA5 (ASCII) coding of the characters. In this case the protocol is meaningless since the Shared format has a mixture of coding including binary, BCD and IA5.
- If the Call Vectoring set command initially creates the ASAI UUI data, the protocol discriminator is always set to 0x04 in both the Service Provider and Shared cases by Communication Manager, otherwise the protocol discriminator setting is not changed. In the case of Shared format the UUI IE protocol discriminator is set to 0x04 regardless of what else is carried by the shared UUI.
- The **`set`** command can only change the actual data bytes and cannot access the protocol discriminator. The **`set`** command assignment affects the set of bytes defined by the start and

length parameters for the vector variable and is limited to the decimal digits (0 - 9) subset of the ASCII (IA5) character set.

- The SIP UUI format is basically the same as the ISDN format but without the first two bytes (the 0x7E op code and length byte). The SIP UUI is in an ASCII string of hex characters (two for each byte) starting with the UUI protocol discriminator byte (00 or 04).
- The ASAI user data sent to the reporting adjunct (Avaya CMS/IQ) contains only the actual user data bytes (decimal digits 0-9) without the protocol discriminator byte.

Invalid results

An invalid result, that is, a failed **`set`** command step logging a vector event and continuing at the next step without changing the ASAI UUI string will occur if:

- The result of a set arithmetic operation (ADD, SUB, MUL or DIV) is greater than the 10 digit 4294967295 digit string.
- The resultant value has fewer digits than the *asaiuui* variable length definition.
- A # character appears in either operand for an arithmetic operation, except in the special case described in “Rules”.
- The length definition with a # assignment is greater than 1.
- The result of a SUB operation is negative.
- A division operation is attempted using *none* or 0, except for a length of 1.
- The result of a MOD 10 operation or any other invalid operation is a #.

Operand1

Operand1 is the left operand. Operand1 can be any of the following:

- The user-assigned A to Z and AA to ZZ collect vector variables. The collect vector variable can be either local or global.
- The system-assigned A to Z and AA to ZZ vector variables, such as ani, asaiuui, and doy.
- V1 to V9 VDN variables.
- digits: The collected digits buffer for the current contents of the call.
- none: A keyword denoting a null or empty string for a string operator, or a 0 for an arithmetic operator.

Operand2

Operand2 is the right operand. Operand2 can be any of the following:

- The user-assigned A to Z and AA to ZZ collect vector variables. The collect vector variable can be either local or global.
- The system-assigned A to Z and AA to ZZ vector variables, such as ani, asaiuui, and doy.
- V1 to V9 VDN variables.
- **`digits`**: The collected digits buffer for the call.
- **`none`**.

- A directly-entered numeric value.

Operators

There are three types of operators:

- Arithmetic operators:
  - The ADD operator adds operand1 and operand2.
  - The SUB operator subtracts operand2 from operand1.
  - The MUL operator multiplies operand1 by operand2.
  - The DIV operator divides operand 1 by operand2.
- String operators:
  - The CATL operator concatenates the operand2 digit string to the left end of operand1.
  - The CATR operator concatenates, or appends, the operand2 digit string to the right end of operand1.
  - The SEL operator selects from operand1 the right-most number of digits specified by operand2.
- MOD10 validation. The MOD10 operator validates account numbers, membership numbers, credit card numbers, and checks string lengths using the Modulus 10 algorithm. This is also referred to as the Luhn algorithm. MOD10 is a special string operator.

set command considerations

Dial-ahead digits and the digits buffer

The digits buffer in the **`set`** command does not include dial-ahead-digits, nor does the digits buffer overwrite any current dial-ahead digits unless there is a subsequent collect step.

Digit buffer	Dial-ahead digits	Command: set digits = digits ADD 1111
1234	5678	Sets the digits buffer to 2345 and the dial-ahead digits remain as 5678

Digits buffer	Dial-ahead digits	Command: collect 4 digits
2345	5678	Sets the digits buffer to 5678 and the dial-ahead digits do not contain any digits

DIGITS message

A DIGITS message is sent to Avaya Call Management System when the **`set`** command changes the digits content. Only the last digits sent are saved for the call.

Allowed assignments

Assignment is only allowed to a collect type or asaiuui type vector variable, or to the Digits buffer. If a **`set`** command attempts to assign a value to a system-assignable vector variable or any other

unsupported variable type during vector processing, the **set** command fails and a new assignment not allowed vector event is logged. Vector processing continues at the next step in the vector.

Assigning a new value to a collect variable

If a **set** command assigns a new value to a collect user-assignable vector variable, this new value applies to all subsequent references to that variable in vectors and displays in the Variables for Vector table in the Assignment field.

Determining the number of digits

To determine if the number of digits in variable A is 6 digits, use the following example.

```
1. set B = A MOD10 6
2. goto step 8 if B = # [if it branches to 8, A does not have 6 digits]
3. ...[else A does have 6 digits]
```

Clearing the digits buffer

Once all necessary processing for the collected digits buffer has completed, this example describes how to use the **set** command to clear the collected digits buffer. This prevents the answering agent from viewing the data in the digits buffer. When the agent answers the call, the **Info: display** will be blank.

```
1. collect 9 digits after announcement 4501 for none
2. ...
3. ... [steps 2, 3, and 4 process the collected account number]
4. ...
5. set digits = none CATR none [removes all digits in the collected    digits buffer]
6. queue-to skill 1st pri l
```

Advanced set command rules and applications

About arithmetic operations

set [*variables*, **digits**] **=** *operand1* [**ADD, SUB, MUL, DIV**] *operand2*

ADD, SUB, MUL, and DIV perform the following operations:

- An ADD operation performs unsigned long integer addition.
- A SUB operation performs unsigned long integer subtraction.
- A MUL operation performs unsigned long integer multiplication.
- A DIV operation performs unsigned long integer division.

Rules and considerations for arithmetic operations

The following rules and considerations apply to all types of arithmetic operations.

	Results	Operand1	Operand2
Field length	6	6	10
Entries allowed	[A-Z, AA-ZZ] digits	[A-Z, AA-ZZ] V1-V9 digits none	[A-Z, AA-ZZ] V1-V9 digits none numeric values 0-9999999999
Variable or digits buffer *contents*	• #, 0-4295967295 • Leading zeros ignored (007 = 7)	• none, #, 0-9999999999999999 • Leading zeros allowed (007 = 007)	
Variable or digits buffer *evaluation*	• Valid numeric results are 0-4295967295 • Greater than 4295967295 = # • Less than 0 = # • Leading zeros ignored (007 = 7)	• none = 0 • # = # • Greater than 4295967295 = # • Leading zeros ignored (007 = 7)	
Start and length parameters	• Results greater than the length definition = #. • Digits buffer length definition is always 16. • The start definition is ignored.	N/A	N/A

Invalid results for arithmetic operations

During arithmetic operations, a variable or digits buffer can be assigned the # character. The # character signifies an invalid value, an overflow value, or an underflow value.

Examples: Dividing by 0 or none, results in an overflow value. Subtracting by a negative, results in an underflow value.

The # character is always a processing result. You can test the # character in a **`goto`** command by making the # character equivalent to the keyword used in the threshold field.

Example 1: **`goto step x if A = #`**

Example 2: **`goto step x if A <> #`**

The length parameter in arithmetic operations

The length parameter as defined for vector variables is applied only when the resulting set command integer value is assigned to a vector variable using the following rules:

- The length definition for the assigned variable specifies the maximum number of digits of the resulting set command operation value that can be assigned to the variable.
- The resulting integer digit string from the set command operation must be equal to or less than the length definition and never greater than a 10-digit integer value of 4295967295.

- If the result is greater than 4295967295 or the number of digits is greater than the length definition for the variable, the result is converted to a # character to indicate an overflow value. For example, if the definition for variable A is length = 5 and the result of the arithmetic operation is 1234567, the assignment to variable A is # indicating an invalid result.
- If the resulting string is shorter than the length definition, leading zeros are not added to the digit string to match the variable length definition.

Note:

Length is always defined as 16 digits for assignment to the digits buffer. The rules described in this section are applied for assignment to the digits buffer in the same manner as assignment to a vector variable using length = 16.

About string operations

```
set [variables, digits] = operand1 [CATR, CATL, SEL] operand2
```

CATL, CATR, and SEL perform the following operations:

- The CATL function concatenates, or attaches, the operand2 (the right-side operand) digit string to the left or most significant end of operand1 (the left-side operand).
- The CATR function concatenates, or attaches, the operand2 digit string to the right or least significant end of operand1.
- The SEL operation uses the number of digits specified by operand2 to select digits from the digit string in operand1. The digit count starts from the right-most digit position in operand1. For example, **`set A = 1234 SEL 1`** sets A to 4.

Rules and considerations for CATR and CATL operations

The following rules and considerations apply to CATR and CATL string operations. The SEL operation has different rules and considerations.

	Results	Operand1	Operand2
Field length	6	6	16
Entries allowed	[A-Z, AA-ZZ] digits	[A-Z, AA-ZZ] V1-V9 digits none	[A-Z, AA-ZZ] V1-V9 digits none numeric values 0-9999999999999999
Variable or digits buffer *contents*	• none, 0-9999999999999999 • Leading zeros allowed (007 = 007)	• none, #, 0-9999999999999999 • Leading zeros allowed (007 = 007)	
Variable or digits buffer *evaluation*	• none is a null string • 0-9999999999999999 are a string of digits	• none = 0 (null string) • # = none • Leading zeros allowed (007 = 007)	

Table continues…

	Results	Operand1	Operand2
	• Leading zeros allowed (007 = 007) • No invalid results.		

Rules and considerations for SEL operations

The following rules and considerations apply to SEL string operations. The CATL and CATR operations have different rules and considerations.

	Results	Operand1	Operand2
Field length	6	6	6
Entries allowed	[A-Z, AA-ZZ] digits	[A-Z, AA-ZZ] V1-V9 digits none	[A-Z, AA-ZZ] V1-V9 digits none numeric values 0-999999
Variable or digits buffer *contents*	• none, 0-9999999999999999 • Leading zeros allowed (007 = 007)	• none, #, 0-9999999999999999 • Leading zeros allowed (007 = 007)	
Variable or digits buffer *evaluation*	• none is a null string • 0-9999999999999999 are a string of digits • Leading zeros retained (007 = 007) • No invalid results.	• none = none • # = none • Leading zeros retained (007 = 007)	• none = 0 • # = 0 • Greater than 16 = 16 • Leading zeros ignored (007 = 7)

Invalid results for string operations

There are no invalid results for string operations. The results are either decimal digits or null (contains no digits or is set to **`none`**).

Start and length parameters in string operations

The start and length parameters defined for vector variables are both applied only when the resulting set command string operation value is assigned to a vector variable using the following rules:

- The start and length definition for the assigned variable specifies the portion of the resulting set command operation digit string that is selected for the assignment.
- The start definition for the assigned variable is always used to select the portion of the result string to use in the assignment. If start is defined as 1, the portion selected includes all of the left-side digits. For example, if variable S has start = 1, and the result string is 123..., the assignment to S is 123 ... When the start position is greater than 1, the left-side digits before the start position are deleted. For example, if S = 2 and the result string is 123..., the assignment to S is 23...

- The resulting string after application of the variable start position definition must be equal to or less than the length definition. If the string length is greater than the length definition for the variable, the right-side digits are deleted so that the total string length is equal to the length definition. For example, if the definition for variable S is start = 2, length = 5 and the result of the string operation is 1234567, the assignment to variable S is 23456.
- If the resulting string is shorter than the length definition, the string will not have leading zeros added to match the variable length definition.

Note:

For an assignment to the digits buffer, the start position is always defined as 1 and the length is always defined as 16 digits. These rules are applied for assignment to the digits buffer in the same manner as assignment to a vector variable using start = 1 and length = 16.

About the MOD10 operation

```
set [variables, digits] = operand1 MOD10 operand2
```

The MOD10 operator validates account numbers, membership numbers, credit card numbers, and checks string lengths using the Modulus 10 algorithm. This is also referred to as the Luhn algorithm.

You can also use this operation to check for digit-string length. If the length of the digit string in operand1 has the length specified in operand 2, the result is a single digit in the range of 0-9. Otherwise, the result is a #.

Information about the MOD10 algorithm

The MOD10 operator is used to verify the validity of numbers that follow the LUHN-10 or Modulus 10 algorithm. For information on the algorithm see, Secrets of the LUHN-10 algorithm

The Web site describes how the algorithm works and provides a list of merchants and organizations that use the algorithm.

Rules and considerations for MOD10 operations

The following rules and considerations apply to MOD10 operations.

	Results	Operand1	Operand2
Field length	6	6	6
Entries allowed	[A-Z, AA-ZZ] digits	[A-Z, AA-ZZ] V1-V9 digits none	[A-Z, AA-ZZ] V1-V9 digits none numeric values 0-9999999999999999
Variable or digits buffer *contents*	#, 0-9	• none, #, 0-9999999999999999 • Leading zeros allowed (007 = 007)	

Table continues…

	Results	Operand1	Operand2
Variable or digits buffer *evaluation*	• If the numeric value of operand2 is greater than the number of digits in operand1, then the results = # • Valid modulus 10 or Luhn results = 0-9.	• none = none • # = none • Leading zeros retained	• none = # • # = # • Greater than 16 = # • Leading zeros ignored (007 = 7)
Start and length parameters	Do not apply start or length definitions. Start and length are always 1.	N/A	N/A

MOD10 results

The MOD10 function returns one of the following results.

Result	Description
0	The tested digits match the specified number of digits, which is specified in operand2 and passed the Modulus 10 algorithm.
1-9	The string, account number, or credit card number is complete but not valid.
#	A “#” character is returned due to any of the following reasons: • For string length checks, if the string does not match the specified number of digits. • For account number, membership number, or credit card number validations, the received number in operand1 does not match the number of digits specified by operand2. • There is an invalid combination of entries in either operand. For example, you directly or indirectly used either `none` or `#` in either operand. • The result is something other than a digit string in either operand or there are more than two digits in operand2.

Invalid results for MOD10 operations

During MOD10 operations, a variable or digits buffer can be assigned a “#” character. The “#” character signifies that the number of digits in operand1 does not equal the number evaluated in operand2.

Example: The following example results in setting the digits buffer to “#” because operand1 has no digits, but operand2 specifies 16 digits.

```
set digits = none MOD10 16
```

Start and length parameters in MOD10 operations

For MOD10 operations, only a one-digit length is returned. Start is not used.

set command examples

Calculation examples

Arithmetic operation examples

ADD examples

The following table provides examples for using the ADD operator. The ADD operator adds operand1 and operand2.

Command	Result
set A = A ADD 456 A = 000123	123 + 456 = 579 This operation is useful for counters or loop control variables.The result does not retain the zeros.
set A = A ADD none A = 9999999999	9999999999 + 0 = # + 0 = # If an operand contains a value greater than 4294967295, the result is null (#).
set A = A ADD B A = 000123 B = #	123 + # = # The # character in a variable is seen as invalid (#).
set A = A ADD A A = none (or null string)	0 + 0 = 0 none is interpreted as 0
set B = A ADD 456 Variable definitions: • A = 1234 • B = collect type, length = 3, start = 2 (start is ignored)	1234 + 456 = 1690 = # This is an overflow since the result was greater than three digits.
set B = A ADD 456 Variable definitions: • A = 1234 • B = collect type, length = 5, start = 2 (start is ignored)	1234 + 456 = 1690 No leading zeros are included in the result.

SUB examples

The following table provides examples for using the SUB operator. The SUB operator subtracts operand2 from operand1.

Command	Result
`set A = A SUB 1`	123 - 1 = 122

Table continues…

Command	Result
A = 000123	This operation is valuable for count-down variables. If an operand has preceeding zeros and is subtracted from or by another operand, the leading zeros are ignored. The resulting value does not retain the leading zeros.
`set A = A SUB 0456` A = 000123	123 - 456 = -333 = # The smallest result allowed is 0. Negative values are invalid (#).
`set A = A SUB 770` A = 777	777 - 770 = 7 The result is 7, not 007.
`set A = A SUB 10` A = 4294967296	# - 10 = # If an operand contains a value greater than 4294967295, the result is invalid (#).
`set A = A SUB #` A = 000123	123 - # = # Special characters in a variable are invalid.
`set A = A SUB 100` Variable definitions: • A = 123456 • B = collect type, length = 5, start = 3 (start is ignored)	123456 - 100 = 123356 = # This is an overflow condition because the result was greater than five digits.

MUL examples

The following table provides examples for using the MUL operator. The MUL operator multiplies operand1 by operand2.

Command	Result
`set A = A MUL 10` **`A = 000123`**	123 x 10 = 1230 The leading zeros in variable A are ignored.
`set A = A MUL 2` **`A = 4294967295`**	4294967295 x 2 = 8589934590 = # If the result is greater than 4294967295, the result is invalid (#).
`set A = A MUL 5` **`A = #`**	# x 5 = # The # character in a variable makes the result invalid (#).
`set B = A MUL 10` Variable definitions: • A = 000123 • B: collect type, length = 3, start = 2 (start is ignored)	123 x 10 = 1230 = # The leading zeros are ignored in the operation. The result is greater than three digits, causing an overflow condition.

DIV examples

The following table provides examples for using the DIV operator. The DIV operator divides operand1 by operand2.

Command	Result
set A = A DIV 10 A = 000123	120 / 10 = 12 Leading zeros are ignored and the results are not padded with leading zeros.
setA = A DIV 2 A = 5	5 / 2 = 2.5 = 2 The result is rounded down to the nearest whole integer. Results do not contain any decimal values.
set A = A DIV 1000 A = 000123	123 / 1000 = 0.123 = 0 The result is rounded down to the nearest whole integer.
setA = A DIV A A = 9999999999	# / # = # This operation is an operand overflow because the operand values exceed 4294967295.
set A = A DIV # A = 000123	000123 / # = # A # character in an operand makes the operation invalid (#).
set B = A DIV 100 Variable definitions: • A = 12345678 • B: collect type, length = 4, start = 3 (start is ignored)	12345678 / 100 = 123456.78 = 123456 = # The result is rounded down to the nearest integer, but it is an invalid result (#) because it is greater than four digits.
set B = A DIV 100 Variable definitions: • A = 12345678 • B: collect type, length = 6, start = 3 (start is ignored)	12345678 / 100 = 123456.78 = 123456 The result is rounded down to the nearest integer. This result is valid because the result is six digits in length.

String operation examples

CATL examples

The following table provides examples for using the CATL operator. The CATL operator concatenates the operand2 digit string to the left end of operand1.

Command	Result
set A = B CATL 0123 **B = 56789**	56789 CATL 0123 = 012356789
set A = A CATL A **A = #**	# CATL # = none
set A = B CATL 0123	56789 CATL 0123 = 012356789 = 2356

Table continues…

Command	Result
Variable definition: • *A = collect type, length = 4, start = 3* • **B = 56789**	Four digits were selected starting at position 3 in the resulting digit string.
set digits = A CATL 0123 **digits = length=16, start=1** **A = 123456789012345**	123456789012345 CATL 0123 = 0123123456789012345 = 0123123456789012 The first 16 digits are selected and stored in the digits buffer.
set A = A CATL A **A = none (null string or empty)**	none CATL none = none

CATR examples

The following table provides examples for using the CATR operator. The CATR operator concatenates, or appends, the operand2 digit string to the right end of operand1.

Command	Result
set A = B CATR 0123 **B = 56789**	56789 CATR 0123 = 567890123
set A = A CATR A **A = #**	# CATR # = none
set A = B CATR 0123 Variable definition: • *A = collect type, length = 4, start = 3* • **B = 56789**	56789 CATR 0123 = 567890123 = 7890 Four digits were selected starting at position 3 in the resulting digit string.
set digits = A CATR 0123 **digits: length = 16, start = 1** **A = 123456789012345**	123456789012345 CATR 0123 = 01234567890123450123 = 1234567890123450 NOTE: 1234567890123450 is stored in the digits buffer.
set A = A CATR A **A = none (null string or empty)**	none CATR none = none

SEL examples

The following table provides examples for using the SEL operator. The SEL operator selects the right-most number of digits from operand1. The number of digits selected from operand1 is specified by operand2.

Command	Result
set A = B SEL 12 Variable definitions: • A = collect type, length = 10, start = 3	1234567890123 SEL 12 = 234567890123 = 4567890123 Ten digits were selected starting at position 3.

Table continues…

Command	Result
• B = 1234567890123	
set A = B SEL 5 The number of digits specified in operand2 is more than the number of digits in operand1. B = 1234	1234 SEL 5 = 01234 The result contains all of the digits in operand1 with leading zeros as necessary.
set A = B SEL 1 Operand1 contains no digits (#) or is set to zero. B = #	# SEL 1 = 0 The result is null padded with a zero. The SEL 1 adds a leading zero.
set A = B SEL C Operand2 contains no digits or is set to zero. B = 1234 C = #	1234 SEL # = 1234 SEL 0 = none The result is none.
set A = B SEL 3 B = 120005	120005 SEL 3 = 005 The zeros are retained in the result.
set A = B SEL C B = 1234567890123456 C = 99	1234567890123456 SEL 99 = 1234567890123456 Operand2 contains a number greater than 16, therefore a maximum of 16 digits is selected from operand1.
set A = B SEL 16 A = collect type, length = 10, start = 3 B = 1234567890123	1234567890123 SEL 16 = 0123456789 Selects 10 digits starting at position 3.

MOD10 operation examples

The following table provides examples for using the MOD10 operator. The MOD10 operator validates account numbers, membership numbers, credit card numbers, and checks string lengths using the Modulus 10 algorithm.

Command	Result
set B = A MOD10 13 **A = #**	# MOD10 13 = # The operation is invalid if either operand contains a “#” character or “none”.
set B = digits MOD10 A **A = 20**	B = # The operation is invalid because operand2 contains a number greater than 16.
set B = digits MOD10 A **A = none**	B = # The operation is invalid.

Table continues…

Command	Result
`set A = digits MOD10 5` `digits = 123456`	123456 MOD10 5 = # The operation is invalid because operand1 contains a digit string that has more digits than what is specified in operand2.
`set A = digits MOD10 5` `digits = 1234`	1234 MOD10 5 = # The operation is invalid because operand1 contains a digit string that has fewer digits than what is specified in operand2.
`set B = A MOD10 13` `A = 1234567890128`	1234567890128 MOD10 13 = 0 (zero) A 0 result means that the tested digits match the specified number of digits and passes the Modulus 10 algorithm.
`set B = A MOD10 13` `A = 1234567890123`	1234567890123 MOD10 13 = 5 A 1-9 result means that the tested digits match the specified number of digits but fails the Modulus 10 algorithm.

Application examples

Validating numbers example

The XYZ company wants a write a vector subroutine that validates a credit card number before a query is sent to the appropriate credit card company for their validation.

The XYZ company created a subroutine that does the following tasks:

1. Prompt the user for the type of credit card used for the purchase.
   - 1 - diners club
   - 2 - american express
   - 3 - visa
   - 4 - master card
   - 5 - discover
2. If a valid card type (1-5) is entered, prompt for the credit card number.
3. Validate the first four digits of each card number prefix.

   The following table provides a list of card number identifiers and card number lengths for the major credit card companies.

Company	Card number identifier	Card length
Diner's Club or Carte Blanche	300xxxxxxxxxxx 305xxxxxxxxxxx 36xxxxxxxxxxxx 38xxxxxxxxxxxx	14
American Express	34xxxxxxxxxxxxx	15

Table continues…

Company	Card number identifier	Card length
	37xxxxxxxxxxxxx	
VISA	4xxxxxxxxxxxx 4xxxxxxxxxxxxxxx	13, 16
MasterCard	51xxxxxxxxxxxxxx 55xxxxxxxxxxxxxx	16
Discover	6011xxxxxxxxxxxx	16

4. Perform a Luhn or Modulus 10 check on the whole number.
5. If the validation fails validation, indicate that the card number entered is invalid and prompt again for the credit card type.
6. If the card validates, return and let variable A contain the type of card, and the digits buffer contain the validated credit card number.

Examples

```
                            VARIABLES FOR VECTORS

Var Description                 Type    Scope Length Start Assignment       VAC

A   credit card type            collect L     1      1
B   4 digit prefix of cred card collect L     4      1
C   modulus 10 result           collect L     1      1
```

```
                                 CALL VECTOR

    Number: 3                Name: credit card chk
                                          Meet-me Conf? n           Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? n   ASAI Routing? y
 Prompting? y   LAI? n  G3V4 Adv Route? y   CINFO? n   BSR? y   Holidays? y
 Variables? y   3.0 Enhanced? y
01 collect      1    digits after announcement 4501     for A
  (Prompt for credit card type, 1-diners, 2-american, 3-visa, 4-mc, 5-discover)
02 collect      16   digits after announcement 4502     for B
  (Prompt for credit card number followed by #)
03 goto step   7           if B                in     table A
  (check if credit card prefix matches appropriate card type)
04 announcement 4503        ("Bad number,Try again")
05 goto step   1           if unconditionally
06
07 goto step   13          if A                =      1 (Diners Club)
08 goto step   17          if A                =      2 (American Express)
09 goto step   20          if A                =      3 (Visa)
10 goto step   22          if A                >=     4 (Master Card, Discover)
11 goto step   1           if unconditionally  (unknown)
12
13 set          C      = digits MOD10 14      (Diners Club, check)
14 goto step   1           if C                <>     0 (not valid, try again)
15 return             (OKAY! VALID! RETURN!
                            digits buffer contains
                            valid creditcard number)
16
17 set          C      = digits MOD10 15      (American Express)
18 goto step   14          if unconditionally
19
20 set          C      = digits MOD10 13      (Visa check 13 digits)
21 goto step   15          if C                =      0 (OKAY! VALID! RETURN!)
22 set          C      = digits MOD10 16      (VISA, MASTER CARD,
```

```
                                          DISCOVER chk 16 digits)
23 goto step   14          if unconditionally
```

```
add vrt 1                                                        Page   1 of   3

                              VECTOR ROUTING TABLE

      Number: 1       Name: Diners Club         Sort? n

           1: 300?
           2: 301?
           3: 302?
           4: 303?
           5: 304?
           6: 305?
           7: 36??
           8: 38??
--------------------------------------------------------------------------------
add vrt 2                                                        Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 2       Name: American Express    Sort? n
           1: 34??
           2: 37??
--------------------------------------------------------------------------------
add vrt 3                                                        Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 3       Name: VISA                Sort? n
           1: 4???
--------------------------------------------------------------------------------
add vrt 4                                                        Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 4       Name: MASTER CARD         Sort? n
           1: 51??
           2: 52??
           3: 53??
           4: 54??
           5: 55??
--------------------------------------------------------------------------------
add vrt 5                                                        Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 5       Name: MASTER CARD         Sort? n
           1: 6011
-------------------------------------------------------------------------------
```

A 19-digit credit card validation

This example determines the validity of a 19-digit credit card number as is becoming commonplace in EMEA. A 19-digit credit card number is reformatted to a 16-digit segment and a 3-digit segment. The credit card number is valid if the sum of the modulus 10 results for segment 1 and segment 2 are equal to 0 or 10 as described in the following example.

```
1. collect 16 digits after announcement 2300 for A
2. collect 3 digits after announcement 2301 for B
(Create two separate digit strings for LUHN validations)
3. set C = A CATL 0 (insert a dummy 0 at the beginning of the number)
4. set D = A SEL 1 (grab the sixteenth digit)
5. set D = D CATR B (concatenate the 16th digit with the last 3 digits)
6. set C = C MOD10 16
7. set D = D MOD10 4
```

```
8. set C = A MOD10 16
9. set D = B MOD10 4
10.set E = C ADD D
11.goto vector 20 at step 1 if E = +0 [pass 0 test]
12.fail treatment for an invalid credit card
```

A valid number results in a 0 or a multiple of 10.

Using bilingual announcement example

You can program a vector to support bilingual or multilingual announcements. The following table describes the example vectors and announcements.

Announcement	Description
3000	Announcement in English and Spanish asking users to choose between English or Spanish prompts.
1001	Greeting in English [*Welcome ...*]
1002	[*Please enter your ID*]
1003	[*Please wait*]
2001	Greeting in Spanish [*Bienvenida...*]
2002	[*Incorpore su identificación*]
2003	[*Por favor espera*]

Notice that the announcements are administered so that extensions starting with 1 are in English and extensions starting with 2 are in Spanish.

```
1. collect 1 digit after hearing announcement 3000 for A
     a. goto step 1 if A > 2
     b. route to operator if A <= 0
2. set B = A CATR 001 [B = A001, A = 1 or 2]
3. announcement B
4. set B = A CATR 002 [B = A002, A = 1 or 2]
5. collect 16 digits after hearing announcement B for none
6. queue to skill
7. set B = A CATR 003 [B = A003, A = 1 or 2]
8. wait .. hearing B then music
9. goto step 8 if unconditionally
```

Because you can append at the beginning of the string or at the end, you can place a language digit at the beginning or at the end of the string. This example places the language digit at the beginning of the string.

Collecting an account number examples

The first example describes a vector designed to collect an account number without using the **set** command. The second example describes how to use the **set** command to solve the problem.

Example 1: Without using the set command

The following vector shows how to collect an account number. Agents can see the account number if the call is answered by the available agent after steps 1 to 3 are executed. Agents cannot see the

account number after the customer is prompted at step 4. This is because the collect in step 4 overwrites the digits buffer that is sent to the agent's display.

```
 1. collect 9 digits after announcement 4501 [announcement 4501 asks customers for their
account number. Nine digits are stored in the digits
    buffer after customers enter their account number]
 2. queue-to skill 1st  pri h  [Queue the call]
 3. wait-time 30 secs hearing music [call waits, digits buffer = variable A = 9 digits]
 4. collect 1 digits after announcement 4502 [Press 1 if customer wants to leave a
message. The digits buffer now contains the response,
    overwriting previous collected digits.]
 5. goto step 8 if digits = 1 [Check if the caller wants to leave a message]
 6. goto step 3 if unconditionally [The caller does not want to leave a message. Go back
to wait.]
 7. stop
 8. messaging skill 2nd for extension active [Caller leaves a message]
 9. treatment if messaging skill out of service
```

Example 2: Using the set command

The following vector shows how to use the **`set`** command to solve the problem presented in Example 1. The vector in this example overwrites the digits buffer with the initially-stored account number in step 1. While a call is waiting to be answered, the digits buffer is refreshed with the account number.

```
 1. collect 9 digits after announcement 4501 for A [announcement 4501 asks customers for
their account number. Nine digits are stored in the
    digits buffer and the A variable after customers enter their account number.]
 2. queue-to skill 1st  pri h  [Queue the call]
 3. wait-time 30 secs hearing music [call waits, digits buffer = varaible A = 9 digits]
 4. collect 1 digits after announcement 4502 for B [Press 1 if customer wants to leave a
message. The digits buffer and variable B now contain     the response.]
 5. set digits = A SEL 9 [Reset the digits buffer to contain the original 9 digits
collected in step 1]
 6. goto step 9 if B = 1 [Check if the caller wants to leave a message]
 7. goto step 3 if unconditionally [The caller does not want to leave a message. Go back
to wait.]
 8. stop
 9. messaging skill 2nd for extension active [Caller leaves a message]
10. treatment if messaging skill out of service
```

Percentage routing examples

Starting with Communication Manager 5.2, Percentage Allocation Routing using Policy Routing Tables (PRTs) assigned to Vector Directory Numbers (VDNs) is available. The following is an example of how to implement an application using variables and vectoring.

For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

The XYZ company wants to route:

- 25% of calls to the ABC company in India.
- 25% of calls to the DEF company in China.
- 50% of calls to the XYZ company local call center through XYZ company's Communication Manager.

The XYZ company can allocate call types into the following three groups of call handlers: One for India, one for China, and one for the local call center.

Example 1: Percentage routing using VDN variables

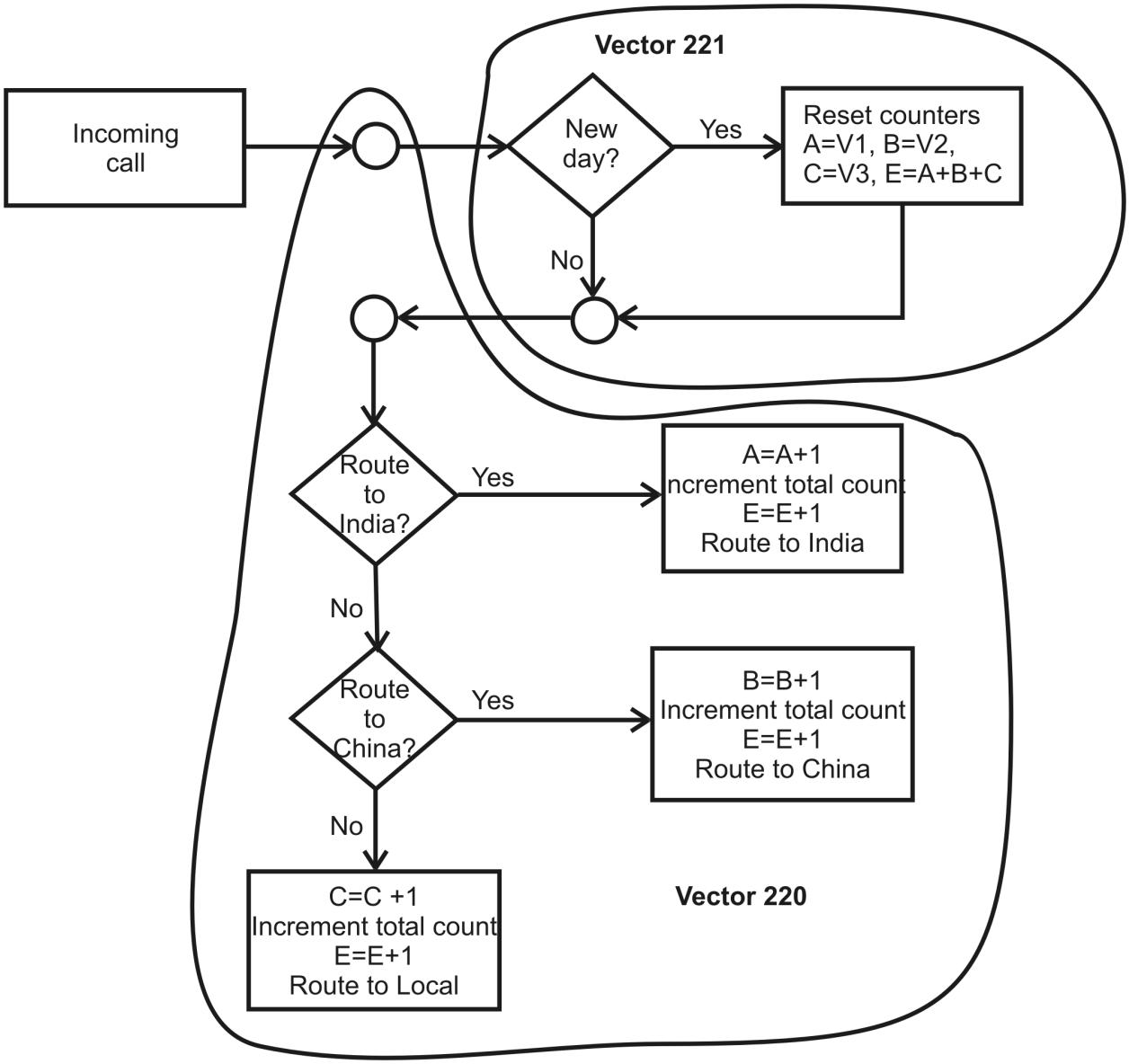
Overview of the tasks in percentage routing using VDN variables example

In this example, an XYZ company uses the **`Set`** command and VDN variables to perform the following tasks:

- Setup vector and VDN variable definitions
- Use vector 220 as a primary vector to calculate percentages, and route calls accordingly
- Use vector 221 as a subroutine vector to initialize routing for the initial call that is performed every day
- Use VDN 2220, with assigned VDN variable values associated with percentage routing, as the VDN that calls vector 220. In this example, the following types of calls are routed to VDN 2220:
  - Toll-free
  - Direct Inward Dialing (DID)
  - Interactive Voice Response (IVR)
  - Transferred by a local agent

Diagram of tasks in percentage routing using VDN variables example

The following flowchart provides an overview of this example.



- New day - True when the dow variable is different from the stored dow value of the first call.
- Route to India - True when the percentage of calls to India is less than the percentage value stored in the VDN variable V1=25%. Variable A (initialized to V1) is the count of calls routed to India. This value is incremented before the call is routed. The total count of variable E is also incremented.
- Route to China - True when the percentage of calls to China is less than the percentage value stored in the VDN variable V2=25%. Variable B (initialized to V2) is the count of calls routed to China. This value is incremented before the call is routed. The total count of variable E is also incremented.
- Percentage of calls
  - India = (A/A+B+C) * 100

- China = (B/A+B+C) * 100
- Local = (C/A+B+C) * 100

Setting up percentage routing example

Procedure

1. Define the Variables for Vectors using the Variables for Vectors screen.

   Example:

```
change variables                                                        Page 1
of x

                                 Variables for Vectors

 Var  Description                 Type    Scope Length Start Assignment       VAC
 A    India Routed Calls          collect   G     16     1
 B    China Routed Calls          collect   G     16     1
 C    Locally Retained Calls      collect   G     16     1
 D    Calculated Percentage       collect   L     16     1
 E    Total Routed Calls Today    collect   G     16     1
 F    Day-of-Week for First Call  collect   G     16     1
 G    Day-of-Week (1=Sun..etc.)     dow     G     16     1        2
```

2. Set up the VDN you want used as the called VDN.

   This VDN calls the vector with the assigned VDN variable values associated with Percentage Routing. In the following example, the VDN is 2220 and the vector number is 220.

   Example:

```
change vdn 2220                                          Page   1 of   3
                           VECTOR DIRECTORY NUMBER

                            Extension: 2220
                                Name*: Percent Routing
                          Destination: Vector Number        220

                  Meet-me Conferencing? n
                    Allow VDN Override? n
                                   COR: 1
                                   TN*: 1
                              Measured: external

               Service Objective (sec): 20
        VDN of Origin Annc. Extension*:
                           1st Skill*:
                           2nd Skill*:
                           3rd Skill*:

* Follows VDN Override Rules
```

```
change vdn 2220                                          Page   2 of   3
                           VECTOR DIRECTORY NUMBER

                                AUDIX Name:
                       Return Destination*:
                    VDN Timed ACW Interval*:
```

```
                    Observe on Agent Answer? n

  Send VDN as Called Ringing Name Over QSIG? n

             Display VDN for Route-To DAC*? n
           VDN Override for ASAI Messages*: no

                  Allow Conference to VDN*? n
         Reporting for PC Predictive Calls? n
          Pass Prefixed CPN to VDN/Vector*? system

* Follows VDN Override Rules
```

3. Define a VDN variable for each country where calls are routed.

   Example:

```
change vdn 2220                                          Page   3 of   3
                            VECTOR DIRECTORY NUMBER

                                VDN VARIABLES*

                    Var  Description       Assignment
                    V1   India             25
                    V2   China             25
                    V3   Local             50
                    V4
                    V5
                    V6
                    V7
                    V8
                    V9




                   VDN Time-Zone Offset*: + 00:00
                  Daylight Saving Rule*: system

* Follows VDN Override Rules
```

4. Use the Call Vector screen to set up a primary vector that calculates percentages and routes calls accordingly.

   This is the main vector for processing calls placed to VDN 2220.

   Example:

```
change vector 220                                              Page   1 of   6
                                  CALL VECTOR

    Number: 220              Name: Percent Route        Background BSR Poll? n
                                        Meet-me Conf? n           Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing? n
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? n   Holidays? y
 Variables? y   3.0 Enhanced? Y
01 goto vector 221 @ step 1 if unconditionally
02 set D = A MUL 100 [D = calculated %, A = India routed calls]
03 etc. use steps 01-21 from the existing vector on page 402-403
```

5. Set up a subroutine vector to initialize the routing every day for the first call.

   This subroutine is called by step 1 in vector 220.

Example:

```
change vector 221                                                Page   1 of   6
                                  CALL VECTOR

    Number: 221              Name: Reset New Day        Background BSR Poll? n
                                             Meet-me Conf? n           Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing? n
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? n   Holidays? y
 Variables? y   3.0 Enhanced? Y
01 goto vector step 8 if F = G [F = day of week for first call, G = day of week]
02 etc. use steps 01-08 of the existing vector on page 403
```

Note:

1 = Sun, 2 = Mon, 3 = Tues, 4 = Wed, 5 = Thurs, 6 = Fri, 7 = Sat

6. Run a **`list trace vdn`** command to verify that each variable is updating correctly.

Example 2: Percentage routing using PRT

Overview of tasks in percentage routing using Policy Routing Table example

Overview of tasks for percentage routing using Policy Routing Table :

- Define VDN 2220 that incoming calls route to assigning the destination to a PRT table.
- Define the PRT table to route calls based on the desired percentages.
- Define the three VDNs that will route the calls to appropriate destination using a VDN variable V1 for route to destination phone number.
  - VDN: 2221 routes to China via 97051001 assigned to V1
  - VDN2: 2222 routes to India via 97051002 assigned to V1
  - VDN3: 2223 route to the local call center via 97051003 assigned to V1
- Define routing vector 220.

Setting up percentage routing using Policy Routing Table

The XYZ company wants to:

- Route 25% of all calls to the ABC company in India
- Route 25% of all calls to the DEF company in China
- Route 50% of all calls to local call center of the XYZ company using Communication Manager

The following are the steps that the XYZ company has to perform to set up percentage routing using the Policy Routing tables (PRT):

Procedure

1. Define VDN 2220 for call handling.

   Example:

```
change vdn 2220                                          Page   1 of   3
                            VECTOR DIRECTORY NUMBER

                             Extension: 2220
                                 Name*: Percent Routing using PRT
```

```
                          Destination: Policy Routing Table   8

                 Meet-me Conferencing? n
                   Allow VDN Override? n
                                  COR: 1
                                  TN*: 1
                             Measured: external

              Service Objective (sec):
       VDN of Origin Annc. Extension*:
                           1st Skill*:
                           2nd Skill*:
                           3rd Skill*:



* Follows VDN Override Rules
```

2. Define the Policy Routing Table (PRT) to allocate percentages of routing to destinations.

```
change policy-routing-table 8                                        Page 1 of 1

                            POLICY ROUTING TABLE

Number: 2    Name: policy 8        Type: percentage        Period: 100-count

Index  Route-to  VDN   Target  Actual  Call
       VDN       Name    (%)     (%)   Counts

1      2221      ABC    25     0.0*    0
2      2222      DEF    25     0.0*    0
3      2223      XYZ    50     0.0*    0
4
5
6

                Totals  100            0
```

3. Define the VDNs (2221, 2222, and 2223) on the Vector Directory Number (VDN) screen for each routing to the destination set to vector number 220 as shown in the Setting up percentage routing example.
4. As required by the application, administer page 2 of the Vector Directory Number (VDN) screen.
5. Set the route destination numbers on page 3 for the VDN variable V1 as the following:
   a. VDN 2221: assign 97051001 to V1 (routes to China)
   b. VDN 2222: assign 97051002 to V1(routes to India)
   c. VDN 2223: assign 97051003 to V1 (routes to the local call center)
6. Define the routing vector 220.

```
change vector 220                                                Page   1 of   6
                                  CALL VECTOR

    Number: 220              Name: Percent Route        Background BSR Poll? n
                                              Meet-me Conf? n           Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing? n
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? n   Holidays? y
 Variables? y   3.0 Enhanced? Y
```

```
01 route-to number V1 with cov n if unconditionally
```

Routing to the PRT VDN acts as a **route-to** command and triggers VDN Override.

Assigning ASAI UUI values

Pass In-VDN Time to an Adjunct example

The ABC Call Center must provide the call processing time to the ASAI adjunct. The ASAI adjunct applies the information as part of the decision to select the appropriate VDN to route the call with adjunct routing. Because the call center is a multiple server system, the call can spend a significant amount of time at previous call center locations before being interflowed to the local site. The application uses the in-processing time along with other information about the call to give higher priority to calls that have been waiting beyond a certain time.

With Communication Manager 4.0 or later, this application can be accomplished using the *vdntime* type vector variable to provide the accumulated “in-VDN” time in seconds and the **set** command assignment to a *asaiuui* type vector variable to populate the ASAI UUI for the call with amount of time the call has waited.

In this example, define vector variable “A” as a *asaiuui* type with start 1, length 4 and scope local. Define vector variable “V” as a *vdntime* type, predefined with scope local and length 4.

The following is an example vector for this application:

```
01 wait 0 secs hearing ringback
02 set A = V SEL 4
03 adjunct routing link 1
04 wait 10 secs hearing ringback
```

Step 2 assigns the accumulated i“n-VDN” time in seconds to the ASAI UUI for the call in the first 4 digit positions of the ASAI UUI user information digit string. Step 3 forwards that “in-VDN” time to the adjunct via the ASAI route_request message sent to the adjunct in the **ASAI UUI IE** field along with the VDN extension and caller ANI. The adjunct can decide what VDN to send the call to via the route_select message.

Display combined wait time and account number to agent example

By using the **set** command assignment to a vector variable defined as the “asaiuui” type, an adjunct-provided account number can be combined with the call in-VDN wait time to be displayed to an agent. The account number is provided by the adjunct in the first six digit positions of the ASAI UUI IE forwarded in a route_select message. The combined ASAI UUI can then be seen by the agent using the Display UUI IE Button feature added in Communication Manager 3.1. This can be in addition to collected digits already associated with the call which is seen by the agent via the Caller-Info display capability, either on the 2nd line of a 2-line display set or on the 1st line by pressing the **callr-info** button.

In this example, vector variable *A* is defined as the *asaiuui* type with length 4 and start 7, and vector variable *V* is assigned as the *vndtime* type. The call is processed through adjunct routing and routed to VDN2 via the route_select message, which includes the ASAI UUI IE with the caller's 6-digit account number, for example, 123456, found during the adjunct processing for the route_request message sent by Communication Manager when executing the adjunct routing command in a

previous vector. For this example, let the in-VDN time for the call be 25 seconds, that is, *V* has the value 25.

The following example vector is assigned to VDN2:

```
01 set A = V SEL 4
02 ...
```

In step 1 V SEL 4 converts the in-VDN time 25 to 0025, places the value (0025) in the ASAI UUI string (123456) starting at position 7. The resultant ASAI UUI contains 1234560025. The agent views the string when the agent presses the **uui-info** button.

stop command

Purpose

The **stop** command halts the processing of any subsequent vector steps.

Syntax

```
stop
```

For information about unexpected results, see Troubleshooting vectors.

Requirements

No special requirements.

Operation

A vector stops processing when:

- A vector step includes a **stop** command
- The last step vector step is processed
- 10,000 vector steps have been processed

The **stop** command halts the processing of any subsequent vector steps. After the **stop** command is processed, any calls that are already queued remain queued, and any wait treatment is continued. Wait treatments include silence, ringback, system music, or alternate audio or music source.

Note:

If a call is not queued when vector processing stops, the call is dropped and tracked as an abandon by both Avaya CMS and BCMS.

The following example shows a vector that uses a **stop** command:

Stopping vector processing

```
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 seconds hearing ringback
5. stop
6. busy
```

In the above example, if the **stop** command is reached, the caller remains in queue at split 21 and continues to hear a ringback. Further vector processing is stopped and does not continue to step 6. Therefore, callers connected to split 21 do no hear a busy signal.

Answer supervision considerations for the stop command

The **stop** command has no effect on answer supervision.

stop command feature interactions

For LAI, the **stop** command is treated as a neutral vector command. When a call drops, the **stop** command is treated as a denial command.

stop command interactions with CMS

When the **stop** command or the end of the vector is encountered, vector INTIME is recorded. This is reported as Avg Time in Vector.

VDISCCALLS database item in the VDN tables pegs call that pass all the way through a vector without ever having been queued.

stop command interactions with BCMS

None.

wait-time command

Purpose

Use the **wait-time** command to create a vector that delays the call with audible feedback. A delay step is provided by the **wait-time** command, which allows the caller to remain on hold for the time indicated in the command.

Syntax and valid entries

wait-time	0-999 secs (for seconds)	hearing	music ringback silence

Table continues…

			i-silent		
	0-480 mins (for minutes) (This option is not available for vector administration done through Avaya Call Management System or Visual Vectors.)		audio source ext. (This consists of a valid announcement or music source extension that is defined on the announcement audio sources form.) A-Z, AA-ZZ V1-V9	then	music ringback silence continue **Note:** The continue treatment is valid only with multiple audio or music sources. The treatment indicates that the caller continues to hear the alternate audio or music source using an announcement until another vector command takes effect
	0-8 hrs (for hours)				

Requirements

Basic Call Vectoring or Call Prompting software must be installed. Also, a music-on-hold port must be provided for the music treatment. Multiple Audio/Music Sources for Vector Delay requires the **Vectoring (G3V4 Enhanced)** field to be enabled.

wait-time command basic 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 terminal).
- Switch receives a destination from the ASAI adjunct.
- Vector disconnect timer expires.

Wait times up to eight hours are allowed for customers who want to use the ASAI Phantom Call feature to track email and fax messages in split queues.

Call delay with audible feedback

The following example shows an announcement that includes the **wait-time** command in a delay step with audible feedback.

```
announcement 2556 [All our agents are busy.]
wait-time 20 seconds hearing music
```

In the example, a caller waits for a minimum of 20 seconds before an agent answers the call. During this wait period, the caller listens to the system music, which is one type of feedback that is available with the **wait-time** command.

If the delay step is the final effective step in the vector, the audible feedback continues beyond the specified duration. In a vector, a final effective step is defined as the last vector step or a vector step that is followed by a **stop** step.

Audible feedback continues:

- Until the call is answered or abandoned, or when the call is not queued when vector processing stops, the call is dropped.
- While a call is queued to any split that is routed to by a **converse-on split** command and data is being passed to a Voice Response Unit (VRU).
- During the wait period before the connection of an announcement or a Touchtone Receiver (TTR).

  For more information, see *Avaya Aura® Call Center Elite Feature Reference*.

Multiple audio or music sources on delay

You can specify an alternative audio or music source for a vector **wait-time** step. This alternative source can be any extension number that is administered on the Announcements/Audio Sources screen.

With Multiple Audio/Music Sources, you can tailor the wait-time feedback to the interests, tastes, or requirements of the audience. You can provide specific types of music or music with overlays of advertising that relate to the service provided by the splits or skills that the vector serves.

Following is an example of an announcement that includes an alternative audio or music source in the **wait-time** step:

Call delay with multiple audio/music source feedback

```
announcement 2556 [All of our agents are busy. Please hold.]
     wait-time 20 seconds hearing 55558 then music
```

When the **wait-time** step is processed, the caller is connected to extension 55558 for 20 seconds. At the end of 20 seconds, the next vector step is executed. The “then” option in the **wait-time** step specifies one of the following:

- What the caller hears if the caller cannot be connected to the specified source.
- When the call is waiting in queue, what the caller hears if the call is not answered in 20 seconds.

In the example, if the call is not answered in 20 seconds, the caller hears the system music until a subsequent announcement, busy, collect, converse-on, disconnect or wait-time step is encountered.

You can specify `music`, that is, the system music, `ringback`, `silence`, or `continue` for the “then” option. When you specify `continue`, the caller continues to hear the alternative audio or music source until the source is replaced by a subsequent vector step regardless of the time specified in the **wait-time** step.

Call delay with continuous audible feedback

You can use alternate audio or music sources in vector loops to provide continuous audible feedback as shown in the following example vector steps.

```
1. ...
2. ...
3. ...
4. wait-time 30 secs hearing 55558 then continue
5. route-to number 913034532212 with cov n
6. goto step 4 if unconditionally
```

In the example shown above, a look-ahead call attempt is placed every 30 seconds on behalf of the caller. If extension 55558 is a long, barge-in, repeating announcement, the caller hears announcement 55558 all the way to the end without the announcement being restarted each time vector processing returns to step 4.

Multiple music sources on hold

You can use the Tenant Partitioning Tenant Number (TN) to associate different music sources for each TN.

- Without EAS, the COR setting of the station or extension that puts the call on hold determines whether music-on-hold is applied.
- With EAS, the COR setting of the logical agent ID is used to determine whether music-on-hold is applied.
- The TN assigned to the destination extension number is associated with a music source number on the Tenant screen.
- The physical location, that is, the port, of the music source is assigned on the Music Sources screen.
- The TN is assigned to the active VDN on the Vector Directory Number screen.

- During vectoring, a **`wait hearing music`** command attaches the vector delay music source that is defined by the TN for the active VDN.
- Alternately, you can also use the Multiple Music Sources for Vector Delay feature to specify music sources. A **`wait hearing extension then...`** command applies the vector delay source. In this case, the music source is defined by the extension specified on the Announcements or Audio Sources screen, rather than the TN assigned to the VDN.
- The TN administered for extensions on the Announcement or Audio Sources screen applies only to direct calls to the announcement extension. For the calls, the announcement or music source assigned to the TN is what the caller hears.
- During vector processing, if the **`converse`** vector command connects the call to an agent when the call remains under vector control and the agent puts the call on hold, the active VDN applies music-on-hold.
- When a vector routes a call to another destination by a **`queue`**, **`check`**, **`route-to`**, or **`messaging split`** command, Communication Manager uses the TN of the last active VDN to determine the music source for music-on-hold.
- In ACD systems without vectoring and where music-on-hold applies, the TN assigned to the called hunt group extension determines which music source callers hear while in queue or on hold.

wait-time command considerations

Music when indicated as a treatment refers to the system music, not to an alternate music source. The tenant number of the active VDN determines the system music the caller hears. You can allow callers to hear a music source other than the one assigned to the active VDN, however, by directly specifying an extension for an audio source with a command such as: `wait-time 30 secs hearing 4301 then music`

The “i-silent” keyword is for use with adjunct routing-ADR/Lookahead Interflow applications. I-silent provides silence for the specified time, but it is neutral to LAI while all other wait treatments (even with 0 secs settings) provide acceptance.

Multiple audio or music sources

The expanded **`wait-time _ secs hearing <extension> then <treatment2>`** command provides what is known as Multiple Audio or Music Sources wait treatment. The <extension> parameter defines an audio or music source that is assigned on the Announcements/Audio Source screen.

The source can be interfaced by way of one of the following:

- Analog/DS1/0 (Line Side T1/E1) station ports.
- AUX-Trunks.
- An Integrated Announcement board.

Any of the listed announcement or audio source types can be configured to do either of the following:

- Play at the beginning with queuing, with the **Queue** field set to `y`, which is always used for call center applications.
- Barge-in operation, that is, the **Queue** field set to `b`.

In addition, integrated board announcements can be set to play once, that is, integrated, or to repeat after each playing continuously, that is, integ-rep.

The <treatment2> parameter refers to the treatment that the caller hears after the source specified by <extension> finishes playing, or the wait-time period expires. The <treatment2> parameter is also provided if the caller cannot be connected to the source. Failure to connect to the source can result from conditions such as:

- source not available - extension or source not assigned.
- source disconnected.
- source busy.
- queuing not assigned.

If the <extension> source is not available when the **`wait`** step is reached in the vector one of the following results occurs:

- If **<treatment2>** is set to `continue`, the caller returns to what the caller hears before the **`wait-time`** step.
- If **<treatment2>** is set to `music`, `ringback`, or `silence`, vector processing still waits for the specified wait-time while the caller hears <treatment2>. When the wait-time period expires, the next step in the vector is executed, irrespective of the <treatment2> setting. The caller continues to hear <treatment2> until a subsequent step changes the treatment. For example, if **<treatment2>** is set to `continue`, and the <extension> source, integ-rep or continuous analog/DS1 or AUX-Trunk, is still playing, the caller continues to hear the source until a subsequent vector steps changes the treatment.

Note:

If the <extension> source stops playing or is disconnected, the caller hears silence.

If the audio or music source specified by the <extension> stops before the wait-time period expires, or the caller cannot be connected to the source, the caller hears the source specified by the <extension> segment of the vector. In this case, if **<treatment2>** is specified as `continue`, the caller hears silence.

Answer supervision considerations for the wait-time command

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.

wait-time command feature interactions

Feature	Description
Music-on-Hold	To use the **`wait-time`** command with music as the treatment, you must administer **music-on-hold**. Otherwise, the caller hears silence. When Tenant Partitioning is in use, the tenant number of the active VDN determines the system music that is heard. Feedback continues while a subsequent vector step queues for an announcement or for a TTR.
LAI	For LAI, the **`wait-time`** command is treated as a call acceptance vector command in all cases, except i-silent, which is a neutral vector command.

wait-time command interactions with CMS/BCMS

The **`wait-time`** command is not tracked by CMS or BCMS. Vectors with wait-time steps are only accessible to CMS if the time unit is administered in seconds.

Chapter 14: How to improve performance

Improved performance depends on the following basic principles:

- Minimize the number of vector steps to process a call.
- Do not use vector steps which have a substantial probability of failure such as the following:
  - Calls made outside of business hours.
  - Queue to groups with less than desirable resources or characteristics.

Inefficient looping wastes processing resources. For example, performance can be compromised when a vector loops through steps too often. This is especially true with long queue times.

The following are some looping examples with suggestions on how to maximize performance:

- Audible Feedback
- Look-Ahead Interflow
- Check

Examples other than looping are as follows:

- After Business Hours
- Look-Ahead Interflow

All looping examples in this section use only loops within a single vector. You must 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:

Test vectors for performance in addition to call flow.

Related links

After business hours example on page 72
Look-Ahead Interflow example on page 72
After business hours example on page 72
Look-Ahead Interflow example on page 72

Looping examples

Audible feedback examples

Note:

Evaluate the length of the wait period between repetitions of an announcement and increase the length. For optimum performance, add a second announcement after the initial announcement and repeat the second announcement less often.

In the following example, an announcement indicates all representatives as busy. Hold the announcement every 10 seconds as long as the call is in queue.

Example: 10-second announcement interval

```
1. queue-to split 1
2. announcement 2770  ["All representatives are busy. Please hold."]
3. wait-time 10 seconds hearing music
4. goto step 2 if unconditionally
5. stop
```

The following example repeats the announcement every 60 seconds, improving performance.

Example: 60-second announcement interval

```
1. queue-to split 1
2. announcement 2770    ["All representatives are busy. Please hold."]
3. wait-time 60 seconds hearing music
4. goto step 2 if unconditionally
5. stop
```

In the following example, a second announcement indicates all representatives as still being busy. Hold, in addition to the initial announcement, and repeat the second announcement less often, say after every 120 seconds, improving performance.

Example: Follow-up announcement

```
1. queue-to split 1
2. announcement 2770    ["All representatives are busy. Please hold."]
3. wait-time 120 seconds hearing music
4. announcement 2771    ["All representatives are still busy. Please continue to hold."]
5. goto step 3 if unconditionally
6. stop
```

The following table compares the relative processing cost of the three examples by looking at the approximate number of vector steps executed when processing the call. Let us say the first announcement is 3 seconds long and the second announcement is 4 seconds long. The approximate number of vector steps executed for an audible feedback is indicated in the following table.

Initial conditions	Example 10 seconds announcement interval	Example 60 seconds announcement interval	Example follow-up announcement
An agent is available in split 1	1	1	1
Queuing time of 5 minutes	70	15	9

When a call is queued for 5 minutes, the number of vector steps drops dramatically when the time between announcements is increased and drops even more when a second announcement is added and the time between announcements is increased again. 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.

Related links

How to improve performance on page 67
How to improve performance on page 67

Look-Ahead Interflow examples

Use the interflow-qpos conditional to achieve first in, first out (FIFO) or near-FIFO call processing. If you do not have the interflow-qpos conditional, add a wait period between successive LAI attempts and extend the waiting period.

The following example continuously attempts an LAI as long as the call is in queue or until a look-ahead attempt succeeds.

Example: continuous look ahead - no delay

```
1. queue-to split 1 pri l
2. announcement 3000
3. wait-time 20 seconds hearing music
4. route-to number 93035555555 cov n if unconditionally
5. goto step 4 if unconditionally
```

The following example adds a delay so that the LAI attempt occurs every 10 seconds.

Example: look ahead with a 10–second delay

```
1. queue-to split 1 pri l
2. announcement 3000
3. wait-time 20 seconds hearing music
4. route-to number 93035555555 cov n if unconditionally
5. wait-time 10 seconds hearing music
6. goto step 4 if unconditionally
```

The following example increases performance by increasing the delay between the LAI attempts to 30 seconds.

Example: look ahead with a 30–second delay

```
1. queue-to split 1 pri l
2. announcement 3000
3. wait-time 20 seconds hearing music
4. route-to number 93035555555 cov n if unconditionally
```

```
5. wait-time 30 seconds hearing music
6. goto step 4 if unconditionally
```

The following table compares the relative processing cost of the three examples by looking at the approximate number of vector steps executed when processing the call with an announcement that is 5 seconds long.

Table 3: Approximate number of vector steps executed for look-ahead interflow examples

Initial conditions	Example look ahead with no delay	Example look ahead with a 10–second delay	Example look ahead with a 30–second delay
An agent is available in split 1	1	1	1
Queuing time of 5 minutes	up to 1,000	85	30

Related links

How to improve performance on page 67
How to improve performance on page 67

Check examples

When using the **check** command to queue a call to backup splits, ensure that an adequate time has elapsed before checking the backup splits again.

Note:

With EWT, the programming style used in this example is not optimal. The best approach is to use EWT to locate an appropriate split for the call and queue the call.

Continuous check

The following example checks the 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 seconds 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
```

Check with 10 second delay

The following example 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 seconds 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 seconds hearing music
10. goto step 4 if unconditionally
```

Agent availability status cannot change every 10 seconds.

Check with 30 second delay

The following example adds a delay of 30 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 seconds 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 seconds hearing music
10. goto step 4 if unconditionally
```

The following table 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 4: Approximate number of vector steps executed for check examples

Initial conditions	Example continuous check	Example check with 10-second delay	Example check with 30-second delay
An agent is available in split 1	1	1	1
Queuing time of 5 minutes	up to 1,000	190	65

If a call is queued for 5 minutes, the number of vector steps drop dramatically under the following two conditions:

- when a delay is added before checking the backup splits again
- when the length of the delay is increased again

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.

Related links

How to improve performance on page 67
How to improve performance on page 67

After business hours example

Recommendation: Test to see if the destination resources are available (such as during business hours) before queuing.

The following example queues calls to a hunt group regardless of the time of the call. When calls reach an office after business hours, the announcement is repeated until the caller hangs up.

Unconditional queuing to hunt group

```
1. queue-to split 1
2. announcement 5000  ("All agents are busy. Please hold.")
3. wait-time 120 seconds hearing music
4. announcement 5001   ("All agents are still busy. Please continue to hold.")
5. goto step 3 if unconditionally
```

The next example tests for business hours before queuing the call. If the call reaches the office after business hours, an announcement informs the caller of the business hours and the call is terminated.

Queue to hunt group with time-of-day conditional

```
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 seconds 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.")
```

In the first example, unnecessary processing occurs when a call queues after business hours and the call is terminated only when the caller hangs up. As indicated in the second example, it is more economical to test for business hours before queuing a call.

Related links

How to improve performance on page 67
How to improve performance on page 67

Look-Ahead Interflow example

Note:

When using LAI, check if the receiving office is open for business.

Scenario: The sending Communication Manager is in Los Angeles with office hours from 8:00 AM to 5:00 PM PST and the receiving Communication Manager is in New York with office hours from 8:00 AM to 5:00 PM EST (05:00-14:00 PST). There is a three hour difference between the two locations.

The following example routes calls to the New York Communication Manager.

Unconditional LAI

```
1. queue-to split 1
2. route-to number 99145555555 cov n if unconditionally
```

```
3. announcement 2770      (All agents are busy. Please hold.)
4. wait-time 120 seconds hearing music
5. goto step 3 if unconditionally
6. stop
```

The next example tests first to see if the New York Communication Manager is open before requesting a queue to the New York Communication Manager, preventing unnecessary processing.

LAI with Time-of-Day (TOD) condition

```
1. queue-to split 1
2. goto step 4 if time-of-day is all 14:00 to all 05:00
3. route-to number 99145555555 cov n if unconditionally
4. announcement 2770        (All agents are busy. Please hold.)
5. wait-time 120 seconds hearing music
6. goto step 4 if unconditionally
7. stop
```

Refer to the next example if you enabled **Advanced Routing**. In this case, the Expected Wait Time feature can be used to determine if placing an LAI call attempt is useful.

LAI with Expected Wait Time (EWT) and TOD conditions

```
1. queue-to split 1
2. goto step 5 if expected-wait for call < 30
3. goto step 5 if time-of-day is all 14:00 to all 05:00
4. route-to number 99145555555 cov n if unconditionally
5. announcement 2770        (All agents are busy. Please hold.)
6. wait-time 120 seconds hearing music
7. goto step 5 if unconditionally
8. stop
```

In the examples, note that there is no reason to attempt an interflow if the call is answered quickly at the main Communication Manager.

Related links

How to improve performance on page 67
How to improve performance on page 67

Chapter 15: Call Vectoring job aid

Vector commands job aid

The vector command job aid lists Call Vectoring commands, along with the various conditions, parameter options and values available for use with each command.

Most vector commands require more than one input value for the command, as well as for various parameters such as an announcement extension number, time interval, and maximum queue size. When the minimum and maximum ranges for command parameter values are identical for all Avaya switch platforms, the limiting ranges are specified in the job aid. Alternately, when the minimum and maximum ranges for a parameter value are not the same among Avaya switch platforms, the upper limit of a value range is indicated by the term switch max.

To determine the maximum values you can use in Call Vectoring commands, see System Capacities Table for Communication Manager on Avaya Aura® Media Servers.

#

Syntax and valid entries

#	A comment command that adds a note with up to 71 characters.
	A comment out command that tells a vector step to ignore processing. Use the edit function, <esc> f6 to insert the command.

adjunct routing link

Syntax and valid entries

adjunct routing link	1-64 -CTI Link ID Link capacity varies with releases and configurations. For more information, see *Avaya Aura® Communication Manager System Capacities Table*. A-Z, AA-ZZ V1-V9

announcement

Syntax and valid entries

`announcement`	extension number A-Z, AA-ZZ V1-V9

busy

Syntax

`busy`	Terminates vector processing after playing a busy signal

check

Syntax and valid entries

`check`	best	if	expected wait	< 1-9999 seconds		
			unconditionally			
			wait improved	> 0-9999 seconds		
	skill	hunt group skills for VDN:1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	if	available-agents > 0-1499	all-levels
						pref-level skill level 1 [Skill levels are 1-16 (1 is best, 16 is lowest). Skill level 2 must be greater than or equal to skill level 1.]
						pref-range skill level 1 to skill level 2
	skill	hunt group skills for VDN: 1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	if	calls-queued < 1-999 expected-wait < 1-9999 seconds oldest-call-wait < 1-999 seconds rolling-asa <1-999 seconds staffed-agents > 0-1499 wait-improved > 0-9999 seconds unconditionally	
	split	hunt group				

Table continues…

	split	hunt group	pri: l=low, m=medium, h=high, or t=top	if	available-agents > 0-1499

collect digits

Syntax and valid entries

collect digits	ced	for `none` or `[A-Z, AA-ZZ]`		
	cdpd			
	1-16	after announcement	extension number. `none`, `A-Z, AA-ZZ`, or `V1-V9`	for `none` or `[A-Z, AA-ZZ]`

consider

Syntax and valid entries

consider	location (multisite BSR only)	1-255, A-Z, AA-ZZ, V1-V9	adjust by 0-100 percent, A-Z, AA-ZZ, or V1-V9
	(Skill) hunt group skills for VDN: 1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	
	(Split) hunt group		

converse-on

Syntax and valid entries

converse-on	(Skill) hunt group skills for VDN: 1st, 2nd, or 3rd	pri: l=low, m=medium, h=high, or t=top	passing	6-digit string, *, #, ani, vdn, digits, qpos, wait, [A-Z, AA-ZZ], V1-V9	and	6-digit string, *, #, ani, vdn, digits, qpos, wait, [A-Z, AA-ZZ], V1-V9
	(Split) hunt group			none		none

disconnect

Syntax and valid entries

disconnect	after announcement	extension number `none`

		A-Z, AA-ZZ V1-V9

goto step and goto vector

Syntax and valid entries

Note:

The maximum vector limit is less on some platforms. Use the help key to determine the applicable limit for your system. The maximum number of port networks and media-gateways varies with the server platform. For example, the S8710 server supports up to 64 port networks and 250 media gateways. Check capacity tables for supported limits.

goto step 1–99 if **goto vector 1– 8000 @ step 1–99 if**					
A-Z or AA-ZZ *V1-V9*			>, <, =, <>, >=, or <=	threshold value or digit string: 1–16, wildcards (? or +), A-Z or AA-ZZ, V1–V9	
			= or <>	none or pound (#) sign	
			in table not in table	1–999, A-Z or AA-ZZ, V1–V9	
ani			>, <, =, <>, >=, or <=	threshold value or digit string: 1–16, wildcards (? or +), A-Z or AA-ZZ, V1–V9	
			=, or <>	none or pound (#) sign	
			in table not in table	1–999, A-Z or AA-ZZ, V1–V9	
available agents	in skill	hunt group skill for VDN: 1st, 2nd, or 3rd	>, <, =, <>, >=, or <=	0-1499, 1-1500, A-Z or AA-ZZ, V1-V9	
	in split	hunt group			
calls-queued	in skill	hunt group skill for VDN: 1st, 2nd, or 3rd	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-098, 1-999, A-Z or AA-ZZ, V1-V9
	in split	hunt group			
counted-calls	to vdn	extension, latest, or active	>, <, =, <>, >=, or <=	0-098, 1-999, A-Z, AA-ZZ, V1-V9	
digits			>, <, =, <>, >=, or <=	threshold value or digit string: 1–16, wildcards (? or +), A-Z or AA-ZZ, V1–V9	
			= or <>	none	

Table continues…

`goto step` 1–99 if **`goto vector` 1– 8000 @ step 1–99 if**						
			=	meet-me access. You can use the option only with meet-me conference vectors.		
			in table not in table	1–999, A-Z or AA-ZZ, V1–V9		
expected-wait for	best		>, <, =, <>, >=, or <=	0-9999 (in seconds), A-Z or AA-ZZ, V1-V9		
	call		>, <, =, <>, >=, or <=			
	split		hunt group	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-9998 (in seconds), 1-9999 (in seconds), A-Z or AA-ZZ, V1-V9
	skill		hunt group skill for VDN: 1st, 2nd, or 3rd			
holiday			in table not in table	1-999, A-Z or AA-ZZ, V1-V9		
ii-digits			>, <, =, <>, >=, or <=	2–digit string, wildcards (? or +), A-Z or AA-ZZ, V1-V9		
			= or <>	none		
			in table not in table	1–999, A-Z or AA-ZZ, V1-V9		
interflow-qpos			>, <, =, <>, >=, or <=	1–9, A-Z or AA-ZZ, V1-V9		
media-gateway			H.248 gateway ID 1–999	= or <>	registered	
			all			
			any			
`goto step` command only • meet-me full • meet-me idle The options are available only with meet-me conference vectors						
no match						
oldest-call-wait in	skill	hunt group for VDN: 1st, 2nd, or 3rd	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-998 (in seconds), 1-999 (in seconds), A-Z or AA-ZZ, V1-V9	
	split	hunt group				

Table continues…

`goto step 1–99` if **`goto vector 1– 8000` @ step `1–99` if**						
port network			port network ID 1– 999	= or <>	registered	
			all			
			any			
queue- fail. Is available only with the Attendant Vectoring feature						
rolling-asa for	skill		hunt group for VDN: 1st, 2nd, 3rd	>, <, =, <>, >=, or <=	0-998 (in seconds), 1-999 (in seconds), A-Z or AA-ZZ, V1-V9	
	split		hunt group			
			vdn extension, latest, or active			
server			= or <>	main, ess, or lsp		
service-hours			in table not in table	1-999 (in seconds), A-Z or AA-ZZ, V1-V9		
staffed-agents	skill	hunt group for VDN: 1st, 2nd, or 3rd	>, <, =, <>, >=, or <=	1-1499, 1-1500, A-Z or AA-ZZ, V1-V9		
	split	hunt group				
time-of-day is	mon, tue, wed, thu, fri, sat, sun, or all		hour: 00–23 minute: 00–59	to	mon, tue, wed, thu, fri, sat, sun, or all	hour: 00–23 minute: 00–59
wait-improved	best		>, <, =, <>, >=, or <=	0-9998 (in seconds), 1-9999 (in seconds), A-Z or AA-ZZ, V1-V9		
	skill		hunt group for VDN: 1st, 2nd, 3rd	pri l=low, m=medium, h=high, or t=top	>, <, =, <>, >=, or <=	0-9998 (in seconds), 1-9999 (in seconds), A-Z or AA-ZZ, V1-V9
	split		hunt group			
unconditionally						

- Wild cards: The question (?) mark matches any digit from 0 to 9 at the specified position. The plus (+) sign matches any or no characters at the specified position.
- Threshold field test: Use the word none to test for an empty digits string. Use the pound (#) sign to match a single pound (#) sign that the caller enters or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
- VDN latest and active: Latest refers to the VDN specified for the current vector and active refers to the VDN specified by the VDN Override settings.

messaging

Syntax and valid entries

messaging	skill	hunt group (A valid hunt group is an ACD split or skill or a non-ACD hunt group assigned for AUDIX, remote AUDIX, MSA, or QSIG MWI on the hunt group.) VDN skill: 1st 2nd 3rd	for	extension number latest active A-Z, AA-ZZ V1–V9
	split	hunt group		

queue-to

Syntax and valid entries

queue-to	attd-group This option is available with Attendant Vectoring.		
	attendant	`extension number`	
	best		
	hunt-group	`group number` A valid group number is a vector-controlled hunt group of any type such as Automatic Call Distribution (ACD) or Uniform Call Distribution (UCD).	priority (pri) • l=low • m=medium • h=high • t=top
	skill	Vector Directory Number (VDN) skills • **1st Skill** • **2nd Skill** • **3rd Skill**	
	split	hunt group	

reply-best

Syntax

```
reply-best
```

Note:

This **`multisite BSR`** command is available only when you activate the Virtual Routing feature.

return

`return`	The **`goto vector`** command can invoke a subroutine call. After the subroutine has processed, the **`return`** command returns vector processing to the step following the **`goto vector`** command.

route-to

Syntax and valid entries

`route-to`	digits with coverage y or n				
	meet-me. The option is available only with meet-me conference vectors.				
	number	Up to 16 digits from 0 to 9 <digits> [A-Z, AA-ZZ, V1-V9] <digits>*<digits>A <digits>#<digits>A <digits>~p<digits>A <digits>~m<digits>A <digits>~s<digits>A <digits>~w<digits>A <digits>~W<digits>A ~r, ~r+ ~r*, ~r# *<digits>~L<digits>#	with coverage y, n	if digit >, >=, <>, =. or <=	0-9 #
				if interflow- qpos <= or >=	1-9
				unconditionally	
	name1, name2, or name3	with coverage y, n			
• **`route to number`**: Supports vector variables from A to Z or from AA to ZZ and VDN variables from V1 to V9. The variable value, in decimal digits, is defined elsewhere before the **`route-to number`** command is executed. Each variable whether a single or double character counts as two digits towards the maximum digits in the number field. The variable can be preceded by digits as long as the total is within the 16 digit or character position limit. The variable must always be the last entry and cannot be followed by a digit.					

Table continues…

- <digits>: The notation <digits> means that more than one digit in the range of 0 to 9 can be inserted for the application.
- Pound (#) sign: The character is used in the threshold field to match a single pound (#) sign entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the “=” or “<>” comparators are valid.
- **route to name**: The parameter is available only with the Dial by Name feature.

Note:

- The **route-to** command supports service observing FACs, VDN observing by location, remote logout of agent FAC, remote access extension, attendant access number, and other destination numbers.
- The **route-to number** command is the destination and is entered in the number field. This field can contain an administration limit of a maximum of 16 decimal digits or combination of characters and numbers that total 16. Special notations such as “~p” with “a~” followed by a character are counted as two digits towards the 16.
- Use of a variable allows having a **route-to number** destination address of more than 16 digits since a variable can be assigned up 16 digits during processing and is combined with the entry in the number field.
- When the specified number is preceded by “~r”, an NCR invocation is attempted back over the trunk group to the network service provider. The “~r” sequence is counted as two digit positions toward the 16 total. The “+” character is an indication for E.164 numbering required by some network service providers for NCR invocation over SIP trunks. The “+” character is counted as two digit positions towards the 16 total. The “~r” or “~r+” entries must be in the initial digit or character positions of the number field.
- By prefixing a VDN number with “~r*” or “~r#” in **route-to number** command, you can access an FAC, or a remote phone number over an NCR. Using the “*” prefix, you can also access a remote number for which you must dial “*9”. For this, you must set up and call a VDN that includes 9 followed by the phone number. For example,

  ```
  route-to number *V1 if cov unconditionally
  ```

  command can route to an external number *913032451234 if *V1* is set up as 913032451234.

set

Syntax and valid entries

The basic syntax of the **set** command is:

set [vector variable, Digits] = [operand1] [operator] [operand2]

set	user-assigned type (Only global or local collect type vector variables can be	=	user-assigned type A-Z or AA-ZZ	ADD, SUB, MUL, DIV, CATL, CATR,	user-assigned type A-Z or AA-ZZ

Table continues…

	assigned using the set command.) A-Z or AA-ZZ			MOD10, or SEL	
	asaiuui A-Z or AA-ZZ		system-assigned A-Z or AA-ZZ		system-assigned A-Z or AA-ZZ
					directly-entered numeric string (Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.)
			V1-V9		V1-V9
	digits (The collected digits buffer holds up to 16 digits.)		digits		digits
			none		none

stop

```
stop
```

wait-time

Syntax and valid entries

wait-time	0-999 secs (for seconds)	hearing	music ringback silence i-silent		
	0-480 mins (for minutes) (This option is not available for vector administration done through Avaya Call Management System or Visual Vectors.)		audio source ext. (This consists of a valid announcement or music source extension that is defined on the announcement audio sources form.) A-Z, AA-ZZ V1-V9	then	music ringback silence continue **Note:** The continue treatment is valid only with multiple audio or music sources. The treatment indicates that the caller continues to hear the
	0-8 hrs (for hours)				

Table continues…

					alternate audio or music source using an announcement until another vector command takes effect

Vector variables job aid

ANI

Variable	Description	Scope	Specification	Maximum digit length	Assigns
ani	Holds the phone number of the caller	L	Start digit position and Length	16	incoming call data

ASAIUUI

Variable	Description	Scope	Specification	Maximum digit length	Assigns
asaiuui	Holds call-specific user data associated with the caller	L	Start digit position and length	16 out of a total of 96	Incoming call or ASAI application data

Collect

Variable	Description	Scope	Specification	Maximum digit length	Assigns
collect	Holds user-defined digits associated with the call for control, routing or special treatment that can be assigned a value by the Variables for Vectors table, collect digits steps or the	L G P	Start digit position and length	16	The for parameter of the **`collect digits`** command or assignment in the variables table

Variable	Description	Scope	Specification	Maximum digit length	Assigns
	`set` vector command.				

DOW

Variable	Description	Scope	Specification	Maximum digit length	Assigns
dow	Holds the current day of week for processing	G	None	1	The main server system clock (1-7) - for example, 1 = Sunday

DOY

Variable	Description	Scope	Specification	Maximum digit length	Assigns
doy	Holds the current day of year for processing	G	None	Always 3	The main server system clock (1-365 or 1 -366 in a leap year)

Stepcnt

Variable	Description	Scope	Specification	Maximum digit length	Assigns
stepcnt	Provides the count of vector steps executed for the call, including the current step	L	None	4	The vector processing step counter

TOD

Variable	Description	Scope	Specification	Maximum digit length	Assigns
tod	Holds the current time of day in 24-hour format	G	None	Always 4	The main server system clock - for

Variable	Description	Scope	Specification	Maximum digit length	Assigns
					example, 02:19 = 2:19 am

Value

Variable	Description	Scope	Specification	Maximum digit length	Assigns
value	Holds a single numerical digit (0-9) for user-defined processing	G	None	1	A user-defined value entered using the Variable Access Code (VAC) Feature Access Code (FAC) procedure or assignment in the variables table

VDN

Variable	Description	Scope	Specification	Maximum digit length	Assigns
vdn	Holds the VDN extension number of the call for processing	L	Active or Latest VDN	13	Routing for a call

VDNTime

Variable	Description	Scope	Specification	Maximum digit length	Assigns
vdntime	Provides the time taken, in seconds, by a call center to process a call	L	None	Always 4	Time in vector processing including prior processing for a call routed by BSR/LAI

Chapter 16: Resources

Documentation

See the following related documents.

Title	Use this document to:	Audience
Understanding		
Avaya Aura® Call Center Elite Feature Reference	Know about Automatic Call Distribution (ACD) and Call Vectoring features.	All users of Call Center Elite
Avaya Aura® Call Center Elite Overview and Specification	Know about Call Center Elite features, performance, specifications, security, licensing information.	Implementation engineers and sales engineers
Using		
Administering Avaya Aura® Call Center Elite	Administer Call Center Elite features.	Implementation engineers and system administrators

Finding documents on the Avaya Support website

About this task

Use this procedure to find product documentation on the Avaya Support website.

Procedure

1. Use a browser to navigate to the Avaya Support website at http://support.avaya.com/.
2. At the top of the screen, enter your username and password and click **Login**.
3. Put your cursor over **Support by Product**.
4. Click **Documents**.
5. In the **Enter your Product Here** search box, type the product name and then select the product from the drop-down list.
6. If there is more than one release, select the appropriate release number from the **Choose Release** drop-down list.
7. Use the **Content Type** filter on the left to select the type of document you are looking for, or click **Select All** to see a list of all available documents.

For example, if you are looking for user guides, select **User Guides** in the **Content Type** filter. Only documents in the selected category will appear in the list of documents.

8. Click **Enter**.

Training

The following courses are available on www.avaya-learning.com. Enter the course code in the **Search** field, and click **Go** to search for the course.

Course code	Course title
AVA00741WEN	Introduction to Call Center Operations
AVA00742WEN	Avaya Call Center - Analyze, Design, and Plan Implementation
5C00091E	Avaya Aura® Call Center Elite Virtual Campus Offering
5C00091I	Avaya Aura® Call Center Elite Implementation and Configuration
5C00091V	Avaya Aura® Call Center Elite Implementation and Configuration

Viewing Avaya Mentor videos

Avaya Mentor videos provide technical content on how to install, configure, and troubleshoot Avaya products.

About this task

Videos are available on the Avaya Support website, listed under the video document type, and on the Avaya-run channel on YouTube.

Procedure

- To find videos on the Avaya Support website, go to http://support.avaya.com and perform one of the following actions:
  - In **Search**, type `Avaya Mentor Videos` to see a list of the available videos.
  - In **Search**, type the product name. On the Search Results page, select **Video** in the **Content Type** column on the left.
- To find the Avaya Mentor videos on YouTube, go to www.youtube.com/AvayaMentor and perform one of the following actions:
  - Enter a key word or key words in the **Search Channel** to search for a specific product or topic.
  - Scroll down Playlists, and click the name of a topic to see the available list of videos posted on the website.

Note:

Videos are not available for all products.

Support

Go to the Avaya Support website at http://support.avaya.com for the most up-to-date documentation, product notices, and knowledge articles. You can also search for release notes, downloads, and resolutions to issues. Use the online service request system to create a service request. Chat with live agents to get answers to questions, or request an agent to connect you to a support team if an issue requires additional expertise.

Glossary

ACD	Automatic Call Distribution (ACD) is a telephony feature for processing and distributing inbound, outbound, and internal calls to groups of extensions.
ACW	An agent enters the After Call Work (ACW) mode to complete ACD call-related activities, such as filling forms or taking notes. An agent in the ACW mode is unavailable to receive ACD calls.
adjunct routing	A means of evaluating calls before the calls are processed by requesting information from an adjunct. Communication Manager requests instructions from an associated adjunct and makes a routing decision based on agent availability or caller information.
adjunct-controlled split	An ACD split that is administered to be controlled by another application. Agents logged in to such splits must do all telephony work, ACD login, ACD logout, and work mode changes through the adjunct, except for auto-available adjunct-controlled splits, wherein agents cannot log in, log out, or change the work modes.
adjunct-monitored call	An adjunct-controlled call, active-notification call, or call that provides event reporting over a domain-control association.
ANI	Automatic Number Identification (ANI) is a display of the calling number for agents to gain access to information about the caller.
ASAI	Adjunct-Switch Application Interface (ASAI) is an Avaya protocol that applications use to gain access to the call-processing capabilities of Communication Manager.
association	A communication channel between adjunct and switch for messaging purposes. An active association is one that applies to an existing call on the switch or to an extension on the call.
attendant	A person at a console who provides personalized service for incoming callers and voice-services users by performing switching and signaling operations.
auto-in	A call-answering mode in which an agent automatically receives ACD calls without pressing any button to receive calls.

AUX work	Agents enter the Auxiliary (AUX) work mode for non-ACD activities, such as taking a break, going for lunch, or making an outgoing call. Agents in the AUX work mode are unavailable to receive ACD calls.
best	The split/skill or location that can provide the best service to a caller as determined by BSR.
BSR	A feature that provides singlesite and multisite load balancing and maximizes staffing resources. Communication Manager uses Best Service Routing (BSR) to compare skills and to route calls to the best skill.
check best	A vector command that Communication Manager uses to verify if the best found split or skill meets all the conditions in the vector. For example, Communication Manager can use the **`check`** command to verify if the best found split or skill has the best Expected Wait Time (EWT).
CO	Central Office (CO) is a switch that a local phone company owns to provide local phone service (dial-tone) and access to toll facilities for long-distance calling.
consider sequence	A consider series plus a **`queue-to best`**, **`check-best`**, or **`reply-best`** step is called a consider sequence.
consider series	A series of **`consider`** commands typically written in sets. A set of **`consider`** commands is called a consider series.
manual-in	A call-answering mode in which an agent must press **manual-in** to receive an ACD call.
queue	An ordered sequence of calls waiting to be processed.
queue-to best	A vector command for queuing calls to the best split or skill that is determined by a consider series.
status poll	A call that Communication Manager makes to gain status data from a remote place in a multisite BSR application plan.
VDN	Vector Directory Number (VDN) is an extension number that directs calls to a vector. VDNs can represent a call type or a service category, such as Billing or Customer Service.
vector-controlled split	A hunt group that you can gain access to only by dialing a VDN extension.
work mode	A function that an agent performs during the work shift. ACD work modes include AUX work, auto-in, manual-in, and ACW.

Index

Special Characters

..... 201
character 266, 269, 270

A

AAR access codes 254
abbreviated dialing
    special characters 250
account number collection example 279
ACD
    call processing 15
active VDN 24
ADD examples 271
adjunct routing
    interactions
        CMS 177
adjunct switch application interface (ASAI) 173
advanced routing 76
agents
    available 20
    staffed 20
Allowed assignments 264
ANI routing 76
ANI without VRT 132
announcement
    delay 180
answer supervision considerations
    converse-on 212
    disconnect 218
    messaging command 232
    queue-to command 238
    reply-best 240
    route-to command 254
    stop command 289
    wait-time 294
Application Enablement Services (AES) 173
arithmetic operations
    about 265
    considerations 265
    invalid results 266
    rules 265
    start and length 266
ARS access codes 254
asaiuui type variable
    used with the set command 260
assigning a new value
    collect variable 265
attendant call queuing 254
authorization code 254

B

basic call vectoring 75
BCMS
    interactions
        disconnect command 219
best service routing
    example 58
bilingual announcements example 279
branching
    conditional 26
    unconditional 26
BSR
    multisite 76
    singlesite 76
    virtual call center 76
    with LAI 76
busy command
    interactions
        BCMS 185
        CMS 185

C

Call delay
    continuous audible feedback 292
call detail recording (CDR) 213
call flow
    converse-VRI calls 200
    methods 18
call processing 159
call prompting 76
call routing with coverage 244
call treatment 20, 27
    personalization 30
    testing 30
call vector screen
    administering 144
CATL examples 273
CATL operations
    considerations 267
    rules 267
CATR examples 274
CATR operations
    considerations 267
    rules 267
class of restriction (COR) 213
comment command 171
comments
    entering 150
    removing 151
conditionals
    reason to use 227

converge parameter 252
converse call placement 200
converse-on command
interactions
BCMS 217

D

determining the number of digits example 265
dial-ahead digits and the digits buffer 264
digits 259
digits buffer clearing example 265
direct department calling (DDC) 213
direct inward dialing (DID) 254
disconnect command 217
DIV operator
DIV examples 272
duplicate vectors 158

E

enhanced vectoring 76
EWT routing 76
example vector
delay with multiple audio/music source feedback 291
leaving recorded messages 231
stopping vector processing 288

F

feature access code
administering 111
feature interactions
check digits 288
consider command 198
disconnect command 218
goto command 229
stop command 289

G

goto command
basic operation 223
interactions
BCMS 229
CMS 229
goto step command 219
goto vector command 219

H

holiday vectoring 76
example 54

I

II-Digits routing 76
information forwarding
CINFO
buffer storage 139
collect digits 138
command set 129
interactions 140
internal transfer to VDN 139
string length 139
UEC IE storage 138
vector example 140
wildcards 139
interflow
look-ahead 18
interflow-qpos conditional, using 76
interflow routing
considerations 252
description 252
intraflow 18
invalid character 266, 269, 270
invalid destinations 244
IVR converse settings 205

J

job aid 303

L

LAI
example 72, 301
examples 69, 298
latest VDN 24
leaving a message 231
limit on calls in queue 234
listed directory number (LDN) 254
look-ahead interflow
example 72, 301
LUHN-10 algorithm 269

M

maximizing performance 70, 299
media-gateway vector conditional 227
messaging 19
messaging command 230
example 231
interactions
BCMS 233
CMS 233
MOD10 algorithm 269
MOD10 operations
considerations 269
examples 275

MOD10 operations *(continued)*
invalid results 270
rules 269
start and length 270
MUL examples 272

N

network call redirection
example 56

P

percentage routing setting 285
performance
improving 70, 299
maximizing 70, 299
processing cost
comparisons 70, 299
port-network vector conditional 227

Q

queue limit 234
queue-to command 234
interactions
BCMS 239
CMS 239
queuing
multiple split 236
queuing calls to best skill 234
queuing calls to local resources 234
queuing priority level 234
queuing to multiple skills 234

R

redirection
VDN 25
redirection on no answer (RONA) 25
removing calls from queues 159
reply-best command interactions
CMS and BCMS 241
routed-to VDN 159
route-to command
destinations 248
interactions
BCMS 258
CMS 257
route-to number command 252
routing
adjunct 18
ANI
call types 130
internal transfers 130
string length 130
vector routing tables 130
wildcards 130
best service 18
caller-selected 19
calls 18
CPN prefix
example 142
multi-national 142
II-digits
codes 134
example 137
routing calls 27

S

SEL examples 274
SEL operations 268
sequential flow 26
server conditionals
syntax 228
server vector conditional 227
service observing 253
set command 258
setting percentage routing 285
singlesite BSR 76
skills, comparing 76
split
multiple 20
queue
priority levels 19
stop command
example 288
syntax 288
stop command interactions
CMS 289
string operations 267
start and length 268
SUB examples 271
subnet trunking 254
support 318
syntax
gateway conditionals
syntax 228
server conditionals 228

T

terminating vector processing 234
time delay administration 203

V

variables in vectors
considerations 103
interactions 103
requirements 89

VDN 15, 22
coverage path 24
creating duplicates 158
duplicate 157
implementation 23
time zone offset 23
variables
announcement extension 118
assigning digits 124
converse-on command 118
disconnect command 118
fields 117
goto command 119
route-to command 120
separation 125
set command 121
wait command 121
VDN Override 24
VDN parameters 24
vector 15
changing 164
command
description 170
messaging 232
reply-best 240
commands
adjunct routing link 173
announcement 179
busy 184
check 185
creating 144, 152
deleting steps 150
duplicate 158
editing 144
example
stopping vector processing 288
identifying links 165
inserting steps 149
processing 15
subroutines 126
testing 164
variables
administration 104
announcement extension 83
collect command 83
converse-on command 84
diconnect command 85
global 90
goto command 85
implementation 81
job aid 106
local 90
local persistent 91
parameters 80
route-to command 87
scope 89
set command 88
wait command 89
variable type
ani 91
asaiuui 92
collect 99
dow 93
doy 94
stepcnt 94
tod 95
value 102
VDN 96
vdntime 97
vector command 219
vector commands 234
..... 303
adjunct routing link 303
announcement 304
busy 304
check 304
collect digits 305
consider 305
converse-on 305
disconnect 305
job aid 303
messaging 309
queue-to 309
reply-best 309
return 241, 310
route-to 24, 310
set 311
stop 312
wait-time 312
vector commands job aid 303
vectoring
benefits 30
commands
adjunct routing 27
announcement 27
busy 27
check backup 27
collect digits 27
consider 27
converse-on 27
disconnect 27
goto step 27
goto vector 27
messaging 27
queue-to 27
queue-to attd-group 27
route-to digits 27
route-to number 27
stop 27
wait-time 27
considerations 75
emergency and routine service 63
examples
attendant routing 49

examples (continued)
examples (continued)
automated attendant 34
customer service center 32
data collection 34
dial by name 60
distributed call centers 37
DIVA 34
help desk 39
insurance agency 39
message collection 34
messaging option 66
night station service 53
notify callers about queue position 45
QSIG CAS 52
resort reservation service 47
warranty service 42
fundamentals 15
vector processing 159
vector processing skips step 234
vector subroutines 76
vector variables 76
ani 313
asaiuui 313
collect 313
dow 314
doy 314
job aid 313–315
stepcnt 314
tod 314
value 315
vdn 315
vdntime 315
videos 317
voice response script 210
VRI
capabilities 210
VRU
execution of VRU script 210
outpulsing data 212
script execution 27
tandemed to ASAI host 210
used as an external announcement 210

W

wait-time command
basic operation 290
considerations 293
example 291
interactions
BCMS 295
CMS 295
look-ahead interflow 295
music on hold 295
purpose 289
syntax 289
work mode
after call work 20
auto-in 20
auxiliary 20
manual-in 20