Avaya Aura™ Call Center 5.2

Release 5.2

Avaya Aura™ Call Center 5.2 Call Vectoring and Expert Agent Selection (EAS) Reference

07-600780
Release 5.2
April 2009

© 2000 - 2009 Avaya Inc. All Rights Reserved.

Notice

While reasonable efforts were made to ensure that the information in this document was complete and accurate at the time of printing, Avaya Inc. can assume no liability for any errors. Changes and corrections to the information in this document might be incorporated in future releases.

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree 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 the Customer or End User.

Link disclaimer

Avaya Inc. is not responsible for the contents or reliability of any linked Web sites referenced elsewhere within this documentation, and Avaya does not necessarily endorse the products, services, or information described or offered within them. We cannot guarantee that these links will work all the time and we have no control over the availability of the linked pages.

Warranty

Avaya Inc. provides a limited warranty on this product. 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 through the Avaya Support Web site:

http://www.avaya.com/support

License

USE OR INSTALLATION OF THE PRODUCT INDICATES THE END USER'S ACCEPTANCE OF THE TERMS SET FORTH HEREIN AND THE GENERAL LICENSE TERMS AVAILABLE ON THE AVAYA WEB SITE http://support.avaya.com/LicenseInfo/ ("GENERAL LICENSE TERMS"). IF YOU DO NOT WISH TO BE BOUND BY THESE TERMS, YOU MUST RETURN THE PRODUCT(S) TO THE POINT OF PURCHASE WITHIN TEN (10) DAYS OF DELIVERY FOR A REFUND OR CREDIT.

Avaya grants End User a license within the scope of the license types described below. 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 End User. "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. "Software" means the computer programs in object code, originally licensed by Avaya and ultimately utilized by End User, whether as stand-alone Products or pre-installed on Hardware. "Hardware" means the standard hardware Products, originally sold by Avaya and ultimately utilized by End User.

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 Product that permits one user to interface with the Software. Units may be linked to a specific, identified Server.

Copyright

Except where expressly stated otherwise, the Product is protected by copyright and other laws respecting proprietary rights. Unauthorized reproduction, transfer, and or use can be a criminal, as well as a civil, offense under the applicable law.

Third-party components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information identifying Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site:

http://support.avaya.com/ThirdPartyLicense/

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 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 Web site:

http://www.avaya.com/support

Trademarks

Avaya, the Avaya logo, COMPAS, DEFINITY, and MultiVantage are either registered trademarks or trademarks of Avaya Inc. in the United States of America and/or other jurisdictions.

All other trademarks are the property of their respective owners.

Downloading documents

For the most current versions of documentation, see the Avaya Support Web site:

http://www.avaya.com/support

Avaya support

Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Support Web site:

http://www.avaya.com/support

Contents

Preface 27
Purpose. 27
Audience 27
Reasons for reissue 27
Related documents 28
Other Call Center documents 28
Associated application documentation 28
Availability 28

Call Vectoring fundamentals 31
About Call Vectoring fundamentals 31
Call management 31
About call management 32
Call flow 32
Caller control. 33
Call queuing to splits 33
Split queue priority levels 34
Agent work mode 34
Calling party feedback. 35
Dialed number identification service (DNIS) 36
Vector processing 37
About vector processing 37
Vector Directory Number 38
VDN variables 39
VDN Time Zone Offset 39
VDN Override. 39
VDN Override for trunk ASAI messages 42
VDN in a coverage path 44
Redirect on No Answer to a VDN 44
Service Observing VDNs 45
Vector control flow 45
Termination versus stopping 46
Programming capabilities 46
About Call Vectoring commands 46
Commands used by the Call Vectoring features 47

Call Vectoring overview 51
What is Call Vectoring? 51
Limitations of traditional ACD call processing 51
Benefits of Call Vectoring 54

Call Vectoring applications 57
List of example applications 57
Customer service center example 59
Automated attendant example 60
Data in/voice answer and data/message collection example 61
Distributed call centers example 65
Help desk example. 67
Insurance agency/service agency example 69
Warranty service (with EAS) example 72
Resort reservation service (with EAS) example 76
About resort the reservation service example 76
Placing the reservation 76
Specific number dialing 77
General number dialing 78
Call-back provisions 79
Attendant routing example 80
About attendant routing 80
Vector administration 81
Local attendant group access code 82
Incoming trunk calls to attendant group 82
Incoming LDN calls 83
QSIG CAS example 83
CAS branch 83
CAS main 84
Night station service example 85
Holiday Vectoring example 86
Network Call Redirection example 88
About the NCR example 88
Primary Vector 89
Status poll vector 89
Interflow Vector 90
BSR using EWT and agent adjustments example 90
About the BSR using EWT and agent adjustments example 91
Primary Vector 92
Status poll vector 92
Interflow Vector 93
Dial by Name 93
Vectors exercises 96

Emergency and routine service 96
Late Caller Treatment 99
Messaging option 101

Basic Call Vectoring 103
Command set 103
Types of Basic Call Vectoring commands 104
Treatment commands 104
Routing commands 105
Branching or programming commands 105
General considerations for Basic Call Vectoring 105

Variables in Vectors 107
About VIV 107
Variable definition parameters 108
Implementing vector variables 109
VIV job aid 110
Command syntax for vector variables 111
announcement commands with vector variables 112
collect command with vector variables 113
converse-on command with vector variables 113
disconnect command with vector variables 114
goto commands with vector variables 115
route-to command with vector variables 117
set command with vector variables 118
wait command with vector variables 118
VIV requirements 119
Understanding local and global variables 119
Definition of local and global variables 119
About local variables 120
About global variables 120
System-assigned vector variable types 121
System-assigned definition 121
ani type variable 122
asaiuui type variable 122
dow type variable 124
doy type variable 125
stepcnt type variable 125
tod type variable 127
vdn type variable 127

vdntime type variable . . . 129
User-assigned vector variable types . . . 130
User-assigned definition . . . 131
collect type variable . . . 131
value type variable . . . 133
VIV interactions and considerations . . . 135
VIV administration . . . 135
Example Variables for Vectors screen . . . 136
Required variable administration entries . . . 137
Optional FAC administration for value variables . . . 138
VIV vector examples . . . 139
Example application using time and day variables . . . 139
Example application using a value variable . . . 143
Example applications using Global Collect variables . . . 144
Example applications using vdn type variables . . . 146
Example application using a variable in other commands . . . 147
Example application using a variable in the converse-on command . . . 148
Troubleshooting vector variables . . . 149

VDN variables . . . 151
Description of VDN variables . . . 151
Reason to use . . . 151
VDN variable fields . . . 152
Where to use VDN variables . . . 152
announcement commands with VDN variables . . . 152
converse-on command with VDN variables . . . 153
disconnect command with VDN variables . . . 154
goto commands with VDN variables . . . 154
route-to command with VDN variables . . . 156
set command with VDN variables . . . 157
wait command with VDN variables . . . 158
Case studies . . . 158
Using one vector for different announcements . . . 158
Combining values in VDN variables to expand capacity . . . 159

Vector subroutines . . . 165
Overview of vector subroutines . . . 165
Reason to use . . . 165
The goto vector command and subroutines . . . 166
The @step parameter . . . 166

Example 1: Test for working hours . . . 167
Incoming call processing vector . . . 167
Checking working hours subroutine vector . . . 168

ANI /II-digits routing and Caller Information Forwarding (CINFO) . . . 169
Command sets . . . 169
ANI routing . . . 170
ANI basics . . . 170
ANI routing example . . . 172
Using ANI with vector routing tables . . . 172
II-digits routing . . . 174
II-digits basics . . . 174
II-digits codes . . . 175
II-digits routing example . . . 180
Caller Information Forwarding . . . 180
CINFO basics . . . 181
CINFO vector example . . . 183
CINFO interactions . . . 183

Multi-national Calling Party Number (CPN) prefixes . . . 185
About multi-national Calling Party Number (CPN) prefixes . . . 185
Administering multi-national CPN prefix options . . . 186
CPN Prefix routing example . . . 186
Before enabling the CPN prefix option . . . 187

Information Forwarding . . . 189
Data handled by Information Forwarding . . . 189
Information Forwarding benefits . . . 190
ISDN network requirements . . . 191
SIP network requirements . . . 192
Information Forwarding support for BSR and LAI . . . 192
Forwarding collected digits with interflowed call . . . 192
Forwarding accumulated in-VDN time . . . 193
Transport by way of globally-supported methods over ISDN . . . 193
Transport by way of globally-supported methods over SIP . . . 194
LAI backward compatibility issues . . . 195
ASAI shared UUI data conversion . . . 195
Determining user information needs . . . 196
User information rules . . . 196
Bytes length ranges for UUI user data . . . 198

Example . . . 199
Information Forwarding troubleshooting . . . 199

Adjunct (ASAI) Routing . . . 201
About Adjunct Routing . . . 201
Considerations for implementing adjunct routing . . . 202
Receiving and implementing an ASAI call route . . . 203
Data sent with an ASAI call route request . . . 205
Special vector processing considerations associated with adjunct routing . . . 206
Effects of ASAI link/application failure on vector processing . . . 206
Simultaneous processing of vector steps and ASAI call route requests . . . 210
Adjunct routing-initiated path replacement . . . 211
Phantom calls . . . 212
Single-step conference . . . 214
Multiple outstanding route requests . . . 215
Multiple call route request example . . . 216

Creating and editing call vectors . . . 217
Methods for entering a vector online . . . 217
Call Vector screen - basic administration . . . 218
Displaying vector variable information . . . 221
How to view vector variable information . . . 221
Display fields . . . 222
Examples . . . 223
Inserting a vector step . . . 224
Deleting a vector step . . . 224
Entering a comment out indication to an existing vector step . . . 225
Removing a comment out indication . . . 226
Creating and constructing a vector . . . 226
About creating and constructing a vector . . . 226
Step 1: Queuing a call to the main split . . . 227
Step 2: Providing feedback and delay announcement . . . 228
Step 3: Repeating delay announcement and feedback . . . 229
Step 4: Queuing a call to a backup split . . . 230
Step 5: Limiting the queue capacity . . . 232
Step 6: Checking for non business hours . . . 233
Duplicating Vectors . . . 234
Duplicate Vector command . . . 234
Duplicate Vector screen field descriptions . . . 235

Creating duplicate vectors 236

Call Prompting 237
About Call Prompting 237
Command set 238
Touch-tone collection requirements 238
Call Prompting digit entry - collect digits command 239
About the collect digits command 239
Removing incorrect digit strings 240
Entering variable-length digit strings 240
Entering dial-ahead digits 241
Functions and examples 241
Treating digits as a destination 242
Using digits to collect branching information 243
Using digits to select options 245
Displaying digits on an agent set 245
Passing digits to an adjunct 247
Creating Service Observing vectors 248
Dial-ahead digits - collect digits command 249
About dial-ahead digits 249
Dial-ahead digit vector examples 250
ASAI-requested digit collection 253
ASAI-provided dial-ahead digits - collect digits command 254
Considerations 254

Advanced Vector Routing - EWT and ASA 257
About Advanced Vector Routing features 257
Advanced Vector Routing command set 258
When to use wait time predictions 258
Expected Wait Time (EWT) 259
How EWT is calculated 260
EWT for a split 260
EWT for a call 261
Passing EWT to a VRU 262
Notifying callers of wait time without a VRU 263
Using EWT to route to the best split 264
Factors that affect EWT values 265
Troubleshooting EWT 266
Rolling Average Speed of Answer (ASA) 266

Rolling ASA versus interval ASA . . . 267
When to use rolling ASA . . . 267
Rolling ASA split calculation . . . 268
Rolling ASA VDN Calculation . . . 268
Combining VDN and ASA routing . . . 269
VDN Calls . . . 269
How VDN Call counts are calculated . . . 269
Using the counted-calls conditional . . . 271

Look-Ahead Interflow (LAI) . . . 273
About LAI . . . 273
LAI prerequisites . . . 274
Example of a two-switch configuration . . . 275
Command set . . . 275
How traditional LAI works . . . 277
LAI commands . . . 277
Example of traditional LAI . . . 279
Receiving switch operation . . . 279
How enhanced LAI works . . . 281
About enhanced LAI . . . 281
The simple way to achieve FIFO . . . 281
Detailed information about the interflow-qpos conditional . . . 282
When does a call not interflow? . . . 283
How the minimum EWT is set . . . 284
Example of single-queue multi-site operation . . . 285
Example of maintaining FIFO processing with LAI . . . 286
Single-queue FIFO considerations . . . 286
Example of LAI in a tandem switch configuration . . . 287
Sending switch operation . . . 287
Tandem switch operation . . . 288
Far-end switch operation . . . 288
LAI-initiated path replacement for calls in vector processing . . . 289
About path replacement for calls in vector processing . . . 289
Example vector . . . 289
DNIS and VDN override in an LAI environment . . . 290
About DNIS and VDN override . . . 290
DNIS information displayed to answering agent . . . 291
Originator's display . . . 292
LAI with network ADR . . . 292

Multi-site applications for Enhanced LAI 293
LAI considerations. 294
Troubleshooting for LAI 295

Best Service Routing (BSR) 297
About BSR 297
Benefits of BSR 298
Server and network requirements for BSR 300
Server requirements 301
Network requirements 301
Special BSR terminology 302
Single-site BSR 304
About single-site BSR 305
Command set - single site BSR 305
How BSR determines the best resource 307
Example of basic single-site BSR 310
User adjustments in single-site BSR 312
Example of single-site BSR with adjustments 314
Planning and administering single-site BSR 317
Planning 317
Administration 318
Troubleshooting for single-site BSR 319
Multi-site BSR 320
Multi-site BSR command set 320
Multi-site BSR applications 323
Example of multi-site BSR 326
BSR available agent strategies 331
More on status poll and interflow vectors 331
User adjustments in multi-site BSR 332
Example of multi-site BSR with limited trunking 333
Example of multi-site BSR with slow networks 339
Example for handling excessive wait times 342
Planning and administering multi-site BSR 342
About planning and administering multi-site BSR 342
Selecting or administering application plans 343
Administering the BSR Application Plan. 343
Local treatment for remotely queued IP and ISDN calls 346
About BSR local treatment for calls queued remotely 346
Overview of local treatment operations 346

Local treatment system requirements . . . 348
Local treatment administration . . . 348
Example vectors for the local treatment feature . . . 349
Special BSR local treatment considerations . . . 353
Troubleshooting for multi-site BSR . . . 354
Tips for writing BSR vectors . . . 354
BSR-initiated path replacement for calls in vector processing . . . 355
Example vector . . . 356

Holiday Vectoring . . . 357
Command set . . . 357
Branching/programming commands . . . 357
Holiday Vectoring overview . . . 358
Administering Holiday Vectoring . . . 359
Enabling Holiday Vectoring . . . 359
Setting up a Holiday Table . . . 360
Changing vector processing for holidays . . . 362
Holiday Vectoring considerations . . . 365

Service Hours Table Routing . . . 367
Service Hours Table Routing overview . . . 367
goto processing . . . 367
Administering Service Hours Table Routing . . . 368
Administering the Service Hours Table screen . . . 368
Administering the goto conditional . . . 369
Service Hours Table Routing considerations . . . 369
Service Hours Table Routing scenario . . . 370

Percentage allocation routing . . . 373
About percentage allocation routing . . . 373
Administering percentage allocation routing . . . 374
Percentage allocation routing example . . . 374
Considerations for implementing percentage allocation routing . . . 375
Rules for percentage allocation routing . . . 377

Network Call Redirection . . . 379
About NCR . . . 379
NCR options supported by PSTNs . . . 380
Protocols not supported by NCR . . . 380
Network Call Transfer-type options . . . 380

Network Call Deflection (NCD) 382
AT&T In-Band Transfer and Connect 383
NCR considerations 386
Limitations on call redirection 386
NCR operational considerations 387
NCR and Information Forwarding 387
UUI data included in Information Forwarding 388
UUI data forwarding 388
PSTN terms used for UUI transport service 388
NCR feature interactions 389
NCR implementation methods 390
NCR activation using call vectoring methods 391
NCR activation using ASAI Call Transfer and third-party Merge/Release operations 395
NCR activation using station call transfer or call conference/release operations . 396
NCR activation using ASAI adjunct route operations 397
NCR administration 397
Basic administration. 398
Station or ASAI transfer or conference/release administration 400
Reserving trunk group B-channels for NCT-type redirection operations 401
Administering NCR with AT&T In-Band Transfer and Connect. 404
General administration for AT&T In-Band Transfer and Connect 404
Setting up DTMF announcements for AT&T In-Band Transfer and Connect 406
Call vectoring methods used with AT&T In-Band Transfer and Connect service . . 407
NCR troubleshooting 408

Network Call Redirection with SIP 413
About NCR support with SIP 413
SIP Network Call Redirection protocols 414
SIP limitations on call redirection 414
SIP NCR and Information Forwarding 414
UUI data included in Information Forwarding 415
UUI data forwarding 415
SIP NCR feature interactions 415
Summary of call vectoring-activated NCR operations 417
BSR queue-to best vector step activation of NCR. 417
NCR administration for SIP 419
NCR troubleshooting 420

Attendant Vectoring 423
About Attendant Vectoring 423
Command set 424
Treatment commands 425
Routing commands 426
Branching/programming commands 429
Overview 430
Vector screen 431
Console Parameters screen 432
TN assignments 433
Restrictions 434
Attendant queue 434
Hunt group queue 434
Redirecting calls to attendant VDNs 434
Night service 435
Attendant VDNs 435
Attendant Vectoring and attendant VDNs 437
Intercept attendant group calls 437
Allow override 438
Interflow between vectors 438
Music source 438
Attendant Vectoring and multiple queueing 438
Restrict queueing to only one type of queue 439
Allow multiple priority queueing within hunt queues 439
Allow multiple hunt group queueing 439
Considerations 439

Meet-me Conference 441
About Meet-me Conference 441
Command set 441
Information collection commands 442
Treatment commands 443
Routing commands 444
Branching/programming commands 444
Administering Meet-me Conference 445
Activating the Meet-me Conference feature 445
Creating a Meet-me Conference VDN 446
Creating a Meet-me Conference vector 447
Interactions 448
Security issues 450

Capacity issues 450
Meet-me Conference call processing scenario 451
Troubleshooting 453
Conference call drops 453
Sound volume is too low 454

Expert Agent Selection 455
What is EAS?. 455
EAS benefits 456
About EAS benefits 457
Skill-based call distribution 457
Interruptible Aux 457
Greatest need call distribution 458
Percent allocation call distribution 458
Percentage allocation routing 458
ACD queuing and vector commands 458
Route calls by skill level. 458
EAS considerations 459
Expert Agent Selection (EAS) terminology 460
EAS-PHD - 60 skills/16 skill levels 461
Switch administration for the EAS feature 462
EAS administration screens 462
Other screens that support EAS Agent LoginID 463
Identifying caller needs 467
About identifying caller needs 467
DNIS/ISDN called party 469
Call Prompting/VRU Digits/CINFO digits 469
Host database lookup 470
Direct Agent Calling 470
Functions and examples 473
Administering skills 473
Preference Handling Distribution 481
Logical Agent capability. 481
Delivering the call to the skill queue 482
Routing the call to an agent. 487
EAS feature interactions 493
EAS adjunct interactions 498
ASAI interactions with EAS 498
Messaging system 501

CMS 501
Speech-processing adjuncts 501
Upgrading to the EAS environment 502

Service Level Maximizer 503
SLM requirements 503
SLM operations 504
SLM agent selection 504
SLM call selection 504
SLM target service levels and agent opportunity costs 505
SLM benefits 507
Auto reserve agents 507
Agent selection rules in mixed skill environments 509
SLM administration 510
SLM algorithms 511
Criteria for choosing algorithms 511
Administering the ASL algorithm 511
SLM reporting 512
Evaluating target service level compliance 512
Evaluating auto reserve rates 513
SLM feature interactions 513

Maximum Agent Occupancy (MAO) 515
Overview 515
When to use MAO 516
MAO administration 516
Determining when an agent is pending availability due to MAO 517
Manual-in mode 517
Auto-in mode 517
Manual override of MAO aux mode 518
Default AUX work reason code for MAO pending state 518
Evaluating MAO using CMS reports 518
MAO feature interactions 519

Call Vectoring commands 521
About Communication Manager contact center packages 522
Communication Manager options required to enable vector commands 522
Vector command description 526
command 527

Purpose . . . 527
Syntax and valid entries . . . 528
Considerations . . . 528
Operation . . . 529
adjunct routing link command . . . 530
Purpose . . . 530
Syntax and valid entries . . . 530
Requirements . . . 530
The adjunct routing link process . . . 531
Feature interactions . . . 534
Avaya IQ interactions . . . 535
CMS interactions . . . 535
BCMS interactions . . . 537
announcement command . . . 538
Purpose . . . 538
Syntax and valid entries . . . 538
Requirements . . . 538
Operation . . . 539
Answer supervision considerations . . . 543
Feature interactions . . . 543
Avaya IQ interactions . . . 544
CMS/BCMS interactions . . . 544
busy command . . . 545
Purpose . . . 545
Syntax . . . 545
Requirements . . . 545
Operation . . . 546
Answer supervision considerations . . . 546
Feature interactions . . . 546
Avaya IQ interactions . . . 547
CMS interactions . . . 547
BCMS interactions . . . 547
check command . . . 548
Purpose . . . 548
Syntax and valid entries . . . 549
Requirements . . . 549
Operation . . . 550
Check split command . . . 551
Check skill for available agents with level preference . . . 552
Answer supervision considerations . . . 553

Feature interactions 553
Avaya IQ interactions 554
CMS interactions 554
BCMS interactions 555
collect digits command 556
Purpose 556
Syntax and valid entries 556
Requirements 556
Operation 557
Answer supervision considerations 561
Feature interactions 561
Avaya IQ interactions 561
CMS/BCMS interactions 561
consider command 561
Purpose 562
Syntax and valid entries 562
Requirements 562
Operation 562
Recommendations 564
Answer supervision considerations 565
Feature interactions 565
Avaya IQ interactions 565
CMS/BCMS interactions 565
converse-on command 566
Syntax and valid entries for the converse-on command 566
Requirements for the converse-on command 566
Operation 567
converse-on split command 570
Answer supervision considerations 572
Feature interactions 573
Avaya IQ interactions 577
CMS interactions 577
BCMS interactions 577
disconnect command 579
Purpose 579
Syntax and valid entries 579
Requirements 580
Operation 580
Answer supervision considerations 580
Feature interactions 581

Avaya IQ interactions 581
CMS interactions. 581
BCMS interactions 582
goto step and goto vector commands 583
Purpose. 583
Syntax and valid entries 584
Requirements 589
Operation 589
Time adjustments using goto conditionals 593
Comparing none, # and numeric digits 593
Media gateway, port network, and server vector conditionals 595
Feature interactions 598
Avaya IQ interactions 598
CMS/BCMS interactions 598
messaging command 600
Purpose. 600
Syntax and valid entries 600
Requirements 601
Operation 601
Answer supervision considerations 603
Feature interactions 603
Avaya IQ interactions 604
CMS interactions. 604
BCMS interactions 605
queue-to command 606
Purpose. 606
Syntax and valid entries 606
Requirements 607
Operation 607
queue-to split command 608
Answer supervision considerations 611
Feature interactions 612
Avaya IQ interactions 612
CMS interactions. 612
BCMS interactions 613
reply-best. 614
Purpose. 614
Syntax 614
Requirements 614
Operation 615

Avaya IQ interactions 615
CMS/BCMS interactions 616
return command 617
Purpose 617
Reason to use 617
Syntax 617
Operation 617
route-to command 619
Purpose 619
Syntax and valid entries 620
Requirements 620
Operation 620
Route-to number command 624
Answer supervision considerations 626
Feature interactions 626
Avaya IQ interactions 628
CMS interactions 629
BCMS interactions 631
set command 632
Description of the set command 632
Reason to use the set command 632
Syntax and valid entries 633
Variable, digits buffer, and asaiuui 633
Operand1 638
Operand2 639
Operators 639
Set command considerations 640
stop command 642
Purpose 642
Syntax 642
Requirements 642
Operation 642
Answer supervision considerations 643
Feature interactions 643
Avaya IQ interactions 643
CMS interactions 644
BCMS interactions 644
wait-time command 645
Purpose 645
Syntax and valid entries 645

Requirements . . . 646
Operation . . . 646
Considerations . . . 649
Feature interactions . . . 651
Avaya IQ interactions . . . 652
CMS/BCMS interactions . . . 652

Appendix A: Job aids . . . 653
Vector commands job aid . . . 653
Vector variables job aid . . . 664

Appendix B: Vector management and monitoring . . . 667
Implementation requirements for the Call Vectoring features . . . 667
Basic Call Vectoring Requirements . . . 668
Call Prompting Requirements . . . 668
G3V4 Enhanced Vectoring Requirements . . . 669
Advanced Vector routing requirements . . . 669
Vectoring (Best Service Routing) requirements . . . 669
CINFO requirements . . . 670
Look-Ahead Interflow requirements . . . 671
Adjunct Routing requirements . . . 671
Holiday Vectoring requirements . . . 672
Network Call Redirection requirements . . . 672
Variables in Vectors requirements . . . 672
VDN variables requirements . . . 672
3.0 Enhanced Vectoring requirements . . . 673
Enabling the Vector Disconnect Timer . . . 673
Upgrading to a Call Vectoring environment . . . 673
Changing and testing a vector . . . 674
Identifying links to a vector . . . 675
Finding all occurrences of a digit string . . . 676

Appendix C: Considerations for the vectoring features . . . 677
Displaying VDN names for vector-initiated DACs . . . 677
Operations . . . 678
Prerequisites . . . 679
Administering the Display VDN for Route-To DAC feature . . . 679
Creating vectors that use the Display VDN for Route-to DAC feature . . . 680
Interactions with other Communication Manager features . . . 682
Transferring calls to VDNs . . . 683

VDN Return Destination 683
About VDN Return Destination 683
User scenario — remote access with host provided security 685
User scenario — saving in trunk facilities between call centers 686

Appendix D: Troubleshooting vectors 689
Criteria for success/failure of call vectoring commands 689
Unexpected feature operations 695
Unexpected command operations 696
Converse command debugging 704
Tracking unexpected events 707
Display events criteria 707
Display events report 708
Vector events 709
Clearing events 726
Global variables can change during processing 726

Appendix E: Advanced multi-site routing 729
Application architecture in multi-site BSR 729
User adjustments 730
User adjustments and the balance in wait times 731
Status polling in BSR 731
About status polling 732
How long do status polls take? 732
Intelligent polling 734
Efficient polling patterns in large networks 735
How many switches should one switch poll? 735
Which remote switches should each switch poll? 736
Considerations for low volume splits/skills 738
About low volume splits/skills 739
Minimizing variations in wait time 740

Appendix F: Advanced information forwarding for ISDN trunks 743
About advanced information forwarding 743
Non-QSIG protocol 744
QSIG trunk group 744
Send Codeset 6/7 LAI IE option interactions 745

Appendix G: Functional differences for DEFINITY G2 and Communication Manager . . . 749

Appendix H: Call Vectoring/EAS and BCMS/CMS interactions . . . 751
About Call Vectoring/EAS and BCMS/CMS interactions . . . 751
CMS/BCMS tracking in a Call Vectoring environment . . . 752
About CMS/BCMS tracking . . . 752
Defining and interpreting call flows . . . 752
Using CMS and BCMS reports to evaluate Call Vectoring activity . . . 761
CMS reports . . . 761
BCMS reports . . . 762
Using CMS in an EAS environment . . . 763
Agents and their skills . . . 763
DAC calls . . . 764
Non-ACD calls . . . 764
VDN skill preferences . . . 764
EAS administration from CMS . . . 765

Appendix I: Operation details for the route-to command . . . 767

Appendix J: Advanced set command rules and applications . . . 773
Arithmetic operations . . . 773
About arithmetic operations . . . 773
Rules and considerations for arithmetic operations . . . 774
Invalid results for arithmetic operations . . . 775
The length parameter in arithmetic operations . . . 775
String operations . . . 776
About string operations . . . 776
Rules and considerations for CATR and CATL operations . . . 777
Rules and considerations for SEL operations . . . 778
Invalid results for string operations . . . 778
Start and length parameters in string operations . . . 779
MOD10 operations . . . 779
About the MOD10 operation . . . 780
Information about the MOD10 algorithm . . . 780
Rules and considerations for MOD10 operations . . . 780
MOD10 results . . . 781
Invalid results for MOD10 operations . . . 782
Start and length parameters in MOD10 operations . . . 782

Appendix K: Set command examples . . . 783
Calculation examples . . . 783

Arithmetic operation examples . . . 783
String operation examples . . . 787
MOD10 operation examples . . . 789
Application examples . . . 790
Validating numbers . . . 790
A 19-digit credit card validation . . . 794
Using bilingual announcements . . . 794
Collecting an account number . . . 795
Percentage routing using VDN variables . . . 796
Assigning ASAI UUI values . . . 803

Appendix L: Notifying callers of queue position example . . . 805
Scenario . . . 805
Solution . . . 805
Prerequisites . . . 805
Tips for setting up the interflow-qpos conditional . . . 806

Appendix M: Call flow and specifications for converse - VRI calls . . . 807
Converse call placement . . . 807
Data passing . . . 808
Using the pound sign . . . 809
How the outpulse sequence works . . . 809
Values administered for <data_1> and <data_2> . . . 810
Administering time delays . . . 811
When the VRU hangs up during data passing . . . 812
Ensuring robust operation of VRU data passing . . . 812
VRU data collection . . . 813
Default and IVR converse settings . . . 813
Script execution . . . 813
Data return . . . 814
Script completion . . . 816
Switch data collection . . . 817

Appendix N: Security issues . . . 819
Remote access . . . 819
Front-ending remote access . . . 819
Replacing remote access . . . 820
EAS . . . 821
Limiting outside access using VDN COR restrictions . . . 821
Vector-initiated service observing . . . 822

Voice response integration 822
Attendant Vectoring 823
Remote logout of agent 823

Appendix O: Setting up a call center 825
About setting up call centers 825
Call Vectoring/non-EAS option 826
Non-EAS Worksheet #1: Call center objectives 830
Non-EAS Worksheet #2: Current split operation 831
Non-EAS Worksheet #3: Customer needs 832
Non-EAS Worksheet #4: Vector design 833
EAS Worksheet #1: Call center objectives 834
EAS Worksheet #2: Current split operation 836
EAS Worksheet #3: Customer needs 837
EAS Worksheet #4: Individual Agent Skills 838
EAS Worksheet #5: Agent Skills 839
EAS Worksheet #6: VDN Skill Preferences 840
EAS Worksheet #7: Vector Design 841

Appendix P: Converting a call center to EAS 843
Before you begin. 843
Step 1: Pre-EAS cutover administration for the system 844
Step 2: Pre-EAS cutover administration for the CMS 848
Step 3: Pre-EAS cutover administration for AUDIX 848
Step 4: Pre-EAS cutover administration for ASAI 848
Step 5: EAS cutover 849

Appendix Q: Feature availability 851

Appendix R: Improving performance 855
About improved performance. 855
Looping examples 856
Audible feedback 856
Look-Ahead interflow 857
Check 859
Other examples 860
After business hours 861
Look-ahead interflows 861

Glossary 863

Index 877

Preface

This section contains the following topics:

- Purpose on page 27
- Audience on page 27
- Reasons for reissue on page 27
- Related documents on page 28
- Availability on page 28

Purpose

The purpose of this guide is to provide detailed information about the Call Vectoring and Expert Agent Selection (EAS) features for a Avaya Call Center Release 5.0.

Audience

This guide is intended primarily for personnel who use Call Vectoring or EAS. A knowledge of Automatic Call Distribution (ACD) is assumed.

The level of your expertise in Call Vectoring and/or EAS determines how you use the guide.

Reasons for reissue

The following features have been added to this document to support Release 5.2:

- Route calls by skill level
- Percentage allocation routing
- Interruptible Aux work
- Increased vector loop count and step count before break
- Multinational CPN Prefixes

Related documents

You might find the following Avaya documentation useful. This section includes the following topics:

- Other Call Center documents on page 28
- Associated application documentation on page 28

Other Call Center documents

These additional documents are issued for Avaya Call Center applications:

- *What's New in Avaya Aura™ Call Center 5.2* - Provides a high-level overview of the new features available for the most-current release.
- *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* - Provides feature descriptions and some implementation guidance for call center features.
- *Avaya Aura™ Communication Manager 5.2 Software - Basic Call Management System (BCMS) Operations* - Provides information on the use of the BCMS feature for ACD reporting.
- *Avaya Business Advocate User Guide* - Provides a general understanding of how Avaya Business Advocate can be used for call and agent selection.

Associated application documentation

The most recent application documentation for Avaya Communication Manager and Avaya Call Management System is available on the Avaya Support web site: http://support.avaya.com.

Availability

Copies of Avaya Call Center documentation are available from The Avaya Support Web site, http://support.avaya.com.

Note:
Always visit the Avaya Support Web site to verify you have the latest version of the Call Center documentation. Additional information about new software or hardware updates will be contained in future issues of this document. New issues of this document will be placed on the Web site when available.

Call Vectoring fundamentals

This section describes the fundamental components of Call Vectoring and includes the following topics:

- About Call Vectoring fundamentals on page 31
- Call management on page 31
- Vector processing on page 37
- Programming capabilities on page 46

About Call Vectoring fundamentals

The manner in which a call is processed depends how the switch is implemented and how the Call Vectoring software is implemented on the switch. The success of the call processing relies on:

- The resources that are available to process a call (for example: agents, splits, software, hardware). This is called call management.
- How the call is processed using vector processing, including VDN usage, vector control flow, and intelligent use of the vector programming capabilities.

Call management

This section includes the following topics:

- About call management on page 32
- Call flow on page 32
- Caller control on page 33
- Call queuing to splits on page 33
- Split queue priority levels on page 34
- Agent work mode on page 34
- Calling party feedback on page 35
- Dialed number identification service (DNIS) on page 36

About call management

When a call is placed to a switch enabled with Call Vectoring, the call is directed to an appropriate vector by means of a Vector Directory Number (VDN). A VDN is a soft extension number that is not assigned to an equipment location. A VDN maps to a single vector, but one or more VDNs can map to the same vector.

Once the call goes to a vector, call routing and treatment are determined by the commands in the vector. Processing starts at the first step and proceeds through the vector. Empty steps are passed over, and the vector process stops after the last step is reached.

However, one vector can direct the call to another vector or VDN, which in turn can direct the call to yet another vector, and so forth, up to a maximum of 10,000 vector steps per call. When a call enters vector processing, a loop counter keeps track of the number of vector steps executed. If the loop counter exceeds 10,000, a **stop** command is executed.

The following sections discuss how calls are routed and queued by way of Call Vectoring. Subsequent sections discuss agent states, priority levels, caller feedback, and caller control.

Call flow

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

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

Multiple split queuing: Allows a call to queue to up to three splits.

Intraflow: Allows calls that are unanswered at a split within a predefined time to be redirected to one or more other splits on the same switch. If redirection depends on a condition to be tested, the process is referred to as conditional intraflow.

Interflow: Allows calls that are directed to a vector to be redirected 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 [Uniform Dialing Plan (UDP)], an external number, or a VDN.

Look-Ahead Interflow (LAI): Can be implemented for call centers with multiple ACD locations that are connected by way of ISDN PRI. This method allows a call to interflow only if a remote location is better equipped to handle the call. LAI can occur only when the proper conditions at the receiving switch are met.

Best Service Routing (BSR): Allows the switch to compare specified splits or skills, identify the split or skill that will provide the best service to a call, and deliver the call to that resource. If no agents are currently available in that split or skill, the call is queued. BSR is available in single-site and multi-site versions. Single-site BSR compares splits or skills on the switch where it resides to find the best resource to service a call. Multi-site BSR extends this capability across a network of switches, comparing local splits or skills, remote splits or skills, or both, and routing calls to the resource that will provide the best service.

Adjunct Routing: Allows the switch to request a routing destination from an adjunct processor by way of Adjunct Switch Application Interface (ASAI). When this feature is enabled, the switch sends the ASAI adjunct a message that contains information about the calling party. The adjunct uses this information to determine, from its databases, the best place for the switch to send the call. The adjunct then passes this routing information back to the switch.

Caller control

Call Vectoring allows for the temporary transfer of call management control to the caller by several methods:

Caller-Selected Routing: This method prompts the caller to input information in the form of dialed digits from a touchtone telephone or from an internal rotary telephone that is located on the same switch. The capability is available if Call Prompting is enabled. A recorded announcement is usually used for prompting purposes. Once the caller inputs the digits, the call is routed to the correct department or destination. This procedure can significantly reduce the number of transferred calls and thus better satisfy the caller's needs.

In addition, if Call Prompting and Call Vectoring (CINFO) are enabled, the vector can collect caller-entered digits that are passed from the network by way of an ISDN message. These digits can be used to enhance caller control in the same way as digits that are collected directly by the switch.

Messaging: The caller can leave a voice message in the event that the call cannot be or has not yet been answered. When messaging is enabled, control is eventually passed to the messaging system split.

Call queuing to splits

Basic Call Vectoring can queue calls to up to three splits simultaneously at any one of four priority levels. This process is called multiple split queuing. The first split to which a call is queued is called the main split, and the second and third split are designated as backup splits. Multiple split queuing enables more efficient utilization of agents, and thus provides better service to callers.

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

- The call is connected to the agent.
- The call is removed from any other queues. Announcements, music, ringback, or other audio source are terminated.
- Vector processing is terminated.

For more information about multiple split queuing, see Multiple split queuing on page 609.

Split queue priority levels

If Call Vectoring is not enabled, queued calls are tracked at one of two priority levels: Medium or High. If a call is queued using Call Vectoring, the call can be assigned one of four priority levels: Top, High, Medium, or Low. Within each priority level, calls are processed sequentially as they arrive.

Note:
A direct agent call is always given the highest priority, and is usually delivered before a call that is directed to a split. The exception is when skill-level Call Handling Preference is optioned and the skill that is administered to receive direct agent calls is not administered as the agent's highest skill level. A direct agent call is an ACD call that is directed to a specific ACD agent rather than to any available ACD agent in the split. For more information, see Direct Agent Calling on page 470.

Note:
If a call is already queued to one or more splits that are currently intended to serve as backup splits, the call could be requeued at the new priority level that is indicated in the command step. For more information on requeuing, see Call Vectoring commands on page 521.

Agent work mode

Call Vectoring can make call management decisions according to real-time agent work modes:

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

These work mode states can appear as conditions within the **`check split`** and **`goto`** Call Vectoring commands, so that the commands can be made to check the number of staffed or available agents.

If a hunt group is not monitored, agents in the hunt group do not have log-in, log-out, or work modes. In such cases, staffed-agents is synonymous with administered, and available-agents is the number of agents who are ready to receive a hunt group call.

For ACD calls, agent states are further defined by the relevant work mode. The following list describes these modes:

After Call Work (ACW) Mode: The agent is unavailable to receive any ACD calls for any split. This mode can be used when the agent is doing ACD call-related work and can be implemented on a timed basis. This is known as Timed ACW. The system automatically places the agent into ACW after the agent completes a call that was received while in the manual-in work mode. In addition, the system can be administered through the Vector Directory Number or Hunt Group screens to automatically place agents into ACW for an administered period of time following the completion of each ACD call that is received while in the auto-in work mode.

Auto-In Work Mode: The agent is available to receive calls and allows the agent to receive a new ACD call immediately after disconnecting from the previous call. When Multiple Call Handling is enabled, an agent in Auto-In Work Mode can elect to receive ACD calls by placing the active call on hold.

Auxiliary-Work Mode: The agent is unavailable to receive any ACD calls for the specified split. This mode can be used when an agent is performing activities that are not associated with the ACD, such as going on a break.

Manual-In Work Mode: The agent is available to receive calls. After the agent disconnects from an ACD call, they are automatically puts into the After Call Work Mode.

Note:
When Multiple Call Handling is enabled, an agent in Manual-In Work Mode can receive additional ACD calls by placing an active call on hold. For more information about agent work modes and Multiple Call Handling, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Calling party feedback

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

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

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

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

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

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

For a CO call that has answer supervision already supplied by way of the processing of an **announcement** or the issuing of a **wait-time** command, the caller may hear any of the following:

- Announcement when any **announcement** command is processed.
- Ringback, silence, system music, or an alternate audio 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.

Examples of how subsequent caller feedback is provided in a vector are provided in Basic Call Vectoring on page 103.

Dialed number identification service (DNIS)

In the traditional ACD arrangement, each agent in a given split is trained to answer calls that are relevant to one specific purpose. However, a call center may wish to use agents trained to address multiple types of calls. This arrangement can allow resources to be used in a more efficient manner, with fewer agents overall and less administrative intervention by the ACD manager. For example, where 5 agents might be needed in each of three smaller splits (15 agents total) to handle 3 types of calls, only 11 or 12 agents might be needed in the combined split.

A network service known as Dialed Number Identification Service (DNIS) is available to exploit multi-skill agent capabilities. DNIS enables a unique multidigit number based on the dialed number associated with the call. The unique number may be sent to an agent, sent to a host computer with ASAI applications, used to provide different treatments for the call, and so forth.

The DNIS number is a function of the telephone number dialed by the caller. Each DNIS number in your telephone system can be programmed to route to an ACD split that is comprised of agents who are proficient in handling several types of calls.

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

Vector processing

This section includes the following topics:

- About vector processing on page 37
- Vector Directory Number on page 38
- VDN variables on page 39
- VDN Time Zone Offset on page 39
- VDN Override on page 39
- VDN Override for trunk ASAI messages on page 42
- VDN in a coverage path on page 44
- Redirect on No Answer to a VDN on page 44
- Service Observing VDNs on page 45
- Vector control flow on page 45
- Termination versus stopping on page 46

About vector processing

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

Vector processing includes the following topics:

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

Vector Directory Number

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

The VDN points to the vector, and it defines the service desired by the caller. The VDN also serves as the application number. It allows for specific call-handling and agent-handling statistical reporting within both the Basic Call Management System (BCMS) and the Avaya Call Management System (CMS) for each application that is handled by the call center.

VDNs are assigned to different vectors for different services or applications that require specific treatments. 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.

Implementation notes

The following list describes special situations due to the type of communication server implementation that cause differences in the available fields on the VDN screen.

- **Data for the Orig Annc** column appears only when **VDN of Origin Announcement** is enabled 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 one or more VDNs).

VDNs can be preassigned to incoming trunk groups, or they can be sent in digit form to the switch by a public or private network. The digits that are sent to the switch can come from the serving Central Office (CO) or toll office by way of the Direct Inward Dialing (DID) feature or DNIS. The digits can also come from another location by way of dial-repeating tie trunks, or they can be dialed by an internal caller. 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 the switch or ACD on a DID or DNIS or on a dial tie-trunk call comprise the VDN. Automatic trunks do not pass destination address digits. Instead, each such trunk always routes to a specific incoming destination that is programmed for the corresponding automatic trunk group. The destination can be an attendant queue, an extension, a hunt group number, or a VDN.

The VDN has several properties. These properties are administered on the Vector Directory Number screen.

For information about the VDN screen, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

VDN variables

VDN variables provide more opportunities for VDNs to use a smaller set of vectors. For more information about VDN variables, see VDN variables on page 151.

VDN Time Zone Offset

This feature is designed for call centers with locations in different time zones. You can program a single vector with TOD conditional steps that handle each time zone based on the *active* VDN for the call.

For more information about this feature, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

VDN Override

This section includes the following topics:

- Basic definition on page 39
- VDN parameters associated with the active VDN on page 40
- Application on page 41
- Detailed operation on page 41

Basic definition

VDN Override changes the *active* VDN for the call. The *active* VDN defines the VDN used for parameters associated with the call, such as VDN name, skills, tenant number, BSR application, VDN variables, and so on. The first VDN reached by the call becomes the *active* VDN. VDN Override allows a routed-to VDN (by a route-to number or route-to digits vector command) to become the *active* VDN.

Note:
Throughout this document the active VDN is the *active* called VDN as modified by VDN Override rules. The *latest* VDN is the most recent VDN to which the call was routed.

For more information about the **VDN Override?** field on the VDN screen, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Besides defining what VDN to use to obtain VDN screen field settings, the *active* VDN can be specified in some vector commands as a keyword. When a vector step with the keyword *active* is executed, the extension for the call's *active* VDN as defined by the VDN override rule is substituted for the keyword when processing the vector command. The keyword *active* can be used as:

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

The keyword *latest*, the last VDN routed-to, can also be assigned in these same vector commands or variable, but the *latest* VDN is not changed by VDN Override settings.

VDN parameters associated with the active VDN

VDN Override allows information about a subsequent VDN to which a call is routed to be used instead of the information about the previously-active VDN. The replacement VDN then becomes the *active* VDN for the call. The information that is associated with the *active* VDN and with the call as it progresses through the vector steps includes:

- VDN Name
- Tenant Number (TN)
- VDN of Origin Announcement Extension
- VDN skills (1st, 2nd, 3rd)
- Return Destination
- VDN Timed ACW Interval
- BSR Application
- BSR Available Strategy
- BSR Tie Strategy
- Display VDN for Route-to DAC
- Trunk ASAI Messenger - see VDN Override for trunk ASAI messages on page 42
- BSR Local Treatment
- VDN Variables
- VDN Time Zone Offset

Application

VDN Override can be used in conjunction with a vector that prompts the caller for a particular service. For example, a call is placed to an automobile dealer. Like most such dealers, this one consists of several departments, including Sales and Parts. Assume that the caller wants to talk to someone in Sales. In this case, the call comes into the Main vector (whose VDN name is *Main*) and is eventually routed to the Sales vector (whose VDN name is *Sales*). If VDN Override is assigned to the Main VDN, the Sales VDN name appears on the agent's telephone display when the call is finally connected to the agent.

Note:
When the Variables in Vectors feature is enabled, VDN override settings may change the VDN extension number value that is assigned to a `vdn` type vector variable. It is based on the *active* VDN for the call. For more information, see vdn type variable on page 127.

Detailed operation

The following table shows how the *active* VDN extension is replaced when a call is routed through a series of VDNs by **`route-to number`** or **`route-to digits`** vector steps.

The *active* VDN extension is determined by the setting of the Allow VDN Override? field for one of the previous VDNs to which the call was routed using a **`route-to`** command according to the following rules:

- If the previous VDN has the Allow VDN Override? field set to **y**, then the *active* VDN extension is overridden with the extension of the *current* VDN.
- If the previous VDN has the Allow VDN Override? field set to **n**, then the current *active* VDN extension remains the same.

The following example describes the VDN Override control of the *active* VDN extension for all calls routed to multiple VDNs by vector processing. VDN 1 is always the initial *active* VDN for the call.

Settings assigned for the Allow VDN Override field on the VDN screen								
VDN 1	y	n	n	n	y	y	y	n
VDN 2	y	y	n	n	n	n	y	y
VDN 3	y	y	y	n	y	n	n	n

***Active* VDN after the call is routed to the next VDN in the sequence**								
After call is routed to VDN2	VDN2	VDN1	VDN1	VDN1	VDN2	VDN2	VDN2	VDN1
After call is routed to VDN3	VDN3	VDN3	VDN1	VDN1	VDN2	VDN2	VDN3	VDN3

Note:

With Expert Agent Selection (EAS) enabled for the system, if the Allow VDN Override? field is set to y for the original VDN, the VDN Skills (defined on page 1 of the Vector Directory Number screen) of the new VDN are used for vector commands where the skill group can be administered as 1st, 2nd, or 3rd. If the Allow VDN Override? field is set to n on the original VDN, the VDN Skills of the original VDN are used for such vector commands.

For Best Service Routing (BSR), if the Allow VDN Override? field is set to y for the original VDN, the settings for the BSR Application and Available Agent Strategy fields (defined on page 2 of the Vector Directory Number screen) of the new VDN are used for BSR-related vector processing. If the Allow VDN Override? field is set to n for the original VDN, the settings for the BSR Application and Available Agent Strategy field settings of the original VDN are used for BSR-related vector processing.

VDN Override for trunk ASAI messages

This feature is used when a CTI application has set up an ASAI VDN or station event-notification association and the application requires the Called Number ASAI message information to be the active VDN extension associated with the incoming call rather than the Called Number digits contained in the ISDN **SETUP** message for the incoming call.

This capability is useful for a CTI application that is monitoring a call where the *active* VDN extension is changed by the following vector scenario:

1. An incoming call is routed to a VDN whose vector prompts the caller to enter one or more digits.
2. The call is then routed to a subsequent VDN by a route-to number or route-to digits vector step.

ASAI messages

The ASAI messages whose Called Number information is affected by this feature are:

- Call Offered
- Alerting
- Queued
- Connect
- Adjunct Route-Request

Important:
The VDN Override for the Trunk ASAI Messages feature is activated for an incoming call when the call is routed to a VDN that has the **VDN Override for ISDN Trunk ASAI Messages?** field on page 2 of the VDN screen set to **y**. When this feature is activated for a call, it remains in effect for the call regardless of the **VDN Override for ISDN Trunk ASAI Messages?** field setting for any subsequent VDNs to which the call is routed.

Called Number information for the ASAI messages described above is affected by the VDN Override for ISDN Trunk ASAI Messages? setting as follows:

- If set to **y**, the VDN Override for ISDN Trunk ASAI Messages feature is activated for an incoming ISDN/PRI call. The Called Number information is the *active* VDN extension associated with the call where the VDN Override feature applies to this extension.
- If set to **n**, the VDN Override for ISDN Trunk ASAI Messages feature is *not* activated for an incoming ISDN/PRI call. The Called Number information is taken from the Called Number digits sent with the incoming ISDN SETUP message for the call where the VDN override feature does not apply for this digit information.

Feature interactions

Feature interactions for the VDN Override for ISDN Trunk ASAI Messages feature are as follows:

- If an incoming ISDN/PRI call has the VDN Override for ISDN Trunk ASAI Messages feature activated, this feature is not preserved when the call is answered by an ACD agent or station user and the call is subsequently transferred to, or conferenced with, another agent or station by the Communications Manager station call-transfer or station call-conference features.
- If an incoming Central Office (CO) call is routed to a VDN that has VDN Override for ISDN Trunk ASAI Messages? activated, it has no effect on the Called Number information for the ASAI messages described above (where the Called Number is the *active* VDN extension associated with the call).

VDN in a coverage path

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

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

- Sending direct agent calls or personal calls to an agent in the 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 AUDIX at night.

For more information, see Option with the VDN as the coverage point on page 610. For information about interactions, see *Feature Description and Implementation for Avaya Communication Manager*.

Redirect on No Answer to a VDN

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

A VDN can be administered as the destination of a RONA processed call. A call that is not answered can be redirected to a VDN to receive special treatment. Enter the number of the destination VDN for a RONA call in the **Redirect to VDN** field on the Hunt Group screen. All calls that are redirected by RONA from that split are sent to the same administered VDN.

If no destination VDN is administered, but the number of rings for redirection is entered, the call redirects back to the split or skill.

Direct agent calls that are not answered follow the agent's coverage path. If no coverage path is administered, calls are redirected to the VDN that is administered as the agent's first primary skill.

For more information, see the Redirection on No Answer section in *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Service Observing VDNs

The Service Observing feature provides the option of being able to observe VDNs. With this option an observer selects a specific VDN and bridges onto calls (one call at a time) that have just started vector processing for that VDN. The observer hears all tones, announcements, music, and speech that the caller and the agent hear and say, including Call Prompting and caller dialing. Also, the observer hears VDN of Origin Announcements. Once the system makes an observing connection to a call in vector processing, it maintains the connection throughout the life of the call until the call is disconnected or until the observer hangs up. This is true even if the call is routed or transferred externally.

For more information about Service Observing VDNs, see the Service Observing section in *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Vector control flow

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

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

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

  Note:
  Any vector command that fails automatically passes control to the following step.

- Unconditional branching unconditionally passes control from the current vector step to either a preceding or succeeding vector step or to another vector. For example, **goto step 6 if unconditionally**.

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

  Note:
  Call Vectoring has an execution limit of 10,000 steps. Once a call enters vector processing, a loop counter keeps track of the number of vector steps executed. If the loop counter exceeds 10,000, a **stop** command is executed.

  Note:
  An implicit wait of 0.2 seconds is provided after every fifteen vector steps if vector processing is not suspended during any one of these steps.

Termination versus stopping

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

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

The termination of vector processing termination differs from stopping, which is caused by the **stop** command or by the execution of the final step in the vector. 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 ringback.
- If vector processing stops and the call is not queued, the call is dropped.

Programming capabilities

This section includes the following topics:

- About Call Vectoring commands on page 46
- Commands used by the Call Vectoring features on page 47

About Call Vectoring commands

Call Vectoring commands perform various call-related functions, which include:

Providing call treatments: Audible feedback, including silence, ringback, system music, or an alternate audio or music source, or a busy tone can be provided to the caller. The caller can be provided with a recorded announcement to indicate that an agent is unavailable to answer the call or to provide other information or instructions. An Audix session can also be initiated.

Vector processing can be delayed for a specific number of seconds before the next vector step is executed. The call can also be disconnected, if necessary.

Routing calls: Calls that are not immediately answered by an agent can be queued to one or more splits. A caller can also leave a recorded message if he or she chooses to do so. Finally, a call can be routed to a number programmed in the vector or to digits that are collected from the caller.

Branching or programming: Branches can be made from one vector step to another such step or to another vector. This can be done unconditionally as well as conditionally. Conditional branching is done according to a number of conditions, for example, number of available agents in a split, number of calls in a split queue, the number of the phone the call is made from, and so forth. Finally, vector processing can be stopped when necessary.

Collecting and acting on information: Optionally, touchtone digits can be collected and 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: Voice scripts on a VRU can be executed for the caller. Voice scripts provide the caller with information or instructions. The caller can often make an appropriate response to a voice script, for example, by entering touchtone digits.

Commands used by the Call Vectoring features

This section lists and describes the commands used by the Call Vectoring features. The list is meant to help familiarize the reader with these commands. The commands are also described in further detail in Call Vectoring commands on page 521.

- Adjunct Routing is available only when the CallVisor ASAI capabilities and Basic Call Vectoring are enabled on the switch. The command causes a message to be sent to an ASAI adjunct requesting routing instructions.
- 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 either connects to an agent in the split or skill or puts the call into its queue at the specified queuing priority level if the condition specified as part of the command is met. A call can be queued to up to three different splits or skills simultaneously.
- Collect Digits collects up to 16 digits that are either entered by the caller during vector processing, sent by the network, or received from an adjunct. An optional announcement can be played first when the digits are being collected directly from the caller.
- Consider Location obtains the Expected Wait Time (EWT) and agent data needed to identify the best remote location in multi-site Best Service Routing applications. One **`consider`** step must be written for each location that you want to check.

- Consider Split/Skill obtains the EWT and agent data needed to identify the best local split or skill in single-site Best Service Routing vectors. One **consider** step must be written for each split or skill that you want to check.
- Converse-on Split integrates Voice Response Units (VRUs) with the switch. Specifically, the command allows voice response scripts to be executed while the call remains in queue, and it allows the passing of data between the switch and the VRU.
- Disconnect ends treatment of a call and removes the call from the switch. The command also allows the optional assignment of an announcement that will play immediately before the disconnect.
- Goto step is a branching step that allows conditional or unconditional movement to a preceding or succeeding step in the vector. Conditional branching is determined by a number of factors. For example: the number of calls that are queued in the split, the number of staffed agents who are in the split, if the call arrives at a time of day that is in a holiday table, and so on.
- Goto Vector is a branching step that allows conditional or unconditional movement to another vector. Conditional branching is determined by a number of factors. For example: the number of calls that are queued in the split, the number of staffed agents who are in the split, if the call arrives at a time of day that is in a holiday table, and so on.
- Messaging Split allows the caller to leave a message for a specified extension or the VDN extension.
- Queue-to unconditionally queues a call to a split or skill and assigns a queuing priority level to the call in case no agents are available. A call that is sent with this command either connects to an agent in the split or skill or enters its queue.
- Queue-to attd-group queues a call to a specified attendant group and is available only for attendant vectors. A call that is sent with this command either connects to an available agent within the group or enters the queue if no agent is available.
- Queue-to attendant queues a call to a specific attendant and is available only for attendant vectors. The call only queues to the agent if the agent is a member of the TN associated with the call.
- Queue-to-hunt group queues a call to up to three hunt groups. A call that is sent with this command connects to an agent in the hunt group or enters the hunt group queue.
- Reply-best returns data to another switch in response to a status poll. **Reply-best** is only used in status poll vectors in multi-site Best Service Routing applications.
- Route-to Digits routes the call to the destination that is specified by a set of digits that are collected from the caller or VRU by the previous **collect digits** step. For more information, see Appendix I: Operation details for the route-to command on page 767.
- Route-to Number routes the call to the destination specified by the administered digit string. For more information, see Appendix I: Operation details for the route-to command on page 767.

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

Condition testing within the commands

As was mentioned in the previous section, a number of the Call Vectoring commands are implemented according to a tested condition that comprises part of the command. In other words. If the condition that is expressed in the command is true, then the command action is executed. If the condition that is expressed in the command is false, then the command action is not implemented, and the next vector step is processed.

For more information about the syntax of each condition, see Call Vectoring commands on page 521.

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

Note:
The available set of conditions is dependent upon the optional features that are enabled. For more information, see Appendix Q: Feature availability on page 851.

- The number of staffed agents in a split
- The number of available agents in a split
- The number of calls queued at a given priority to a split
- The amount of time that the oldest call has been waiting in a split
- Whether or not a call receives special holiday processing
- The Average Speed of Answer for a split or a VDN
- The Expected Wait Time for a split or for a call that has entered vector processing
- A reduction in Expected Wait Time 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 (ANI)
- The type of originating line (II-digits)
- The digits entered by the caller, sent in a message from the network (CINFO), or received from an ASAI or VRU adjunct

- The time-of-day and day of the week that the call is placed. The syntax for this condition can be illustrated as follows: **`mon 8:01 to fri 17:00`** means anytime between 8:01 a.m. Monday through 5:00 p.m. Friday, and **`all 17:00 to all 8:00`** means between 5:00 p.m. and 8:00 a.m. on any day of the week.

Depending on the condition, specific comparison operators and a threshold might 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.

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

Call Vectoring overview

This section provides the following information about basic terminology and concepts associated with Call Vectoring and summarizes its benefits.

This section includes the following topics:

- What is Call Vectoring? on page 51
- Benefits of Call Vectoring on page 54

What is Call Vectoring?

Call Vectoring is the process of defining vector programs that determine how a specific call should be routed and what call treatment that call is to be given.

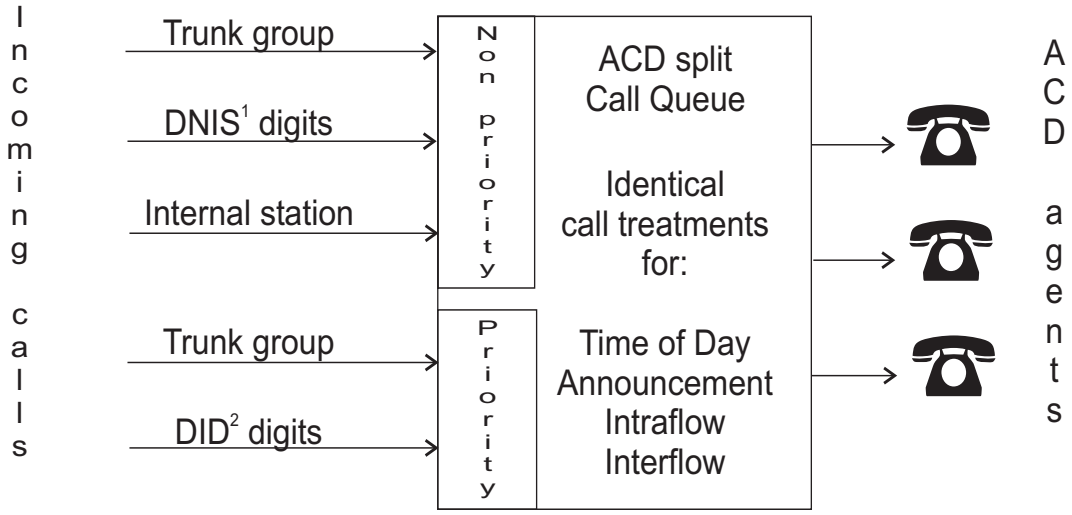
Note:
Sample vectors are provided throughout this manual to illustrate vectoring features and capabilities. Because they are simplified to clearly demonstrate specific features, they are not complete and should not be used without modification at your facility.

Call Vectoring provides a highly flexible approach for managing incoming call traffic to the switch. Using vectors, which are a series of user-defined commands, you can direct or route internal and network calls as desired in your call center and determine how these calls are processed. The processing of calls is known as call treatment. Calls can be directed to on-network or off-network destinations, to ACD agents, or to various other treatments. Call Vectoring also can be used with CallVisor Adjunct Switch Application Interface (ASAI).

Limitations of traditional ACD call processing

The traditional ACD approach is limited in the way it handles queued calls (that is, all calls within a specific queue receive identical announcements, intraflow parameters, and so forth). The following figure shows a simplified illustration of traditional ACD call processing.

Traditional ACD call processing



1. Dialed Number Identification Service
2. Direct Inward Dialing

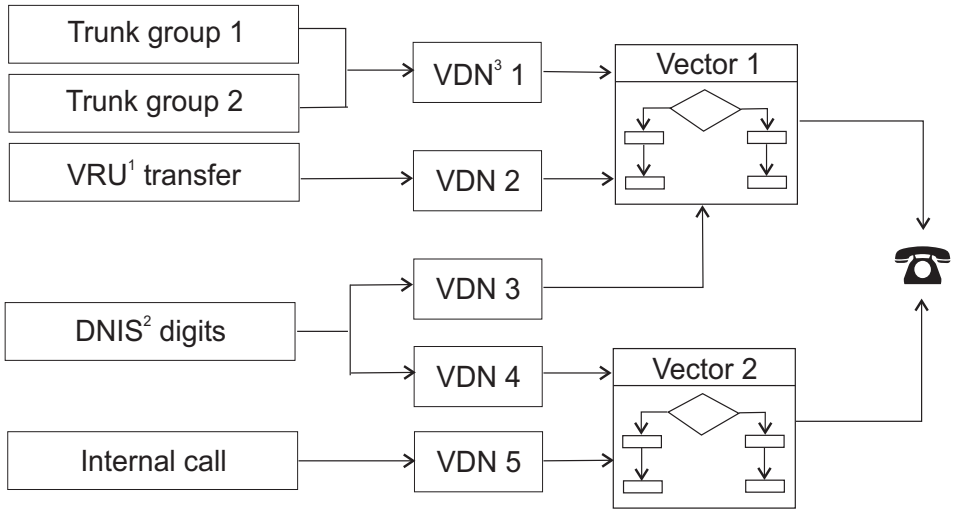
Call Vectoring, on the other hand, permits each call to be treated uniquely according to a number of factors, including the number the caller dials, the number the caller calls from, the number of calls in queue, and the time of day or day of the week. This even applies to all calls that are ultimately handled by the same agent group.

Call Vectoring is comprised of three basic components:

- Vector Directory Numbers
- Vectors
- Vector commands

Working together, these components direct incoming calls and ASAI event reports and requests to the desired answering destinations. They also specify how each call is processed. Call Vectoring may be set up as shown in the following figure.

Use of Call Vectoring for incoming calls



3. Voice Response Unit
4. Dialed Number Identification Service
5. Vector Directory Number

When a call arrives at a switch for which Call Vectoring is enabled, the call is first directed to a Vector Directory Number (VDN). A VDN is an internal telephone number that, in turn, directs the call to a specific vector. The VDN represents the call type or category, for example: billing, customer service, and so on. Thus, it defines the service that is desired by the caller. Multiple VDNs can point to the same or to different vectors, depending on whether the relevant calls are to receive the same or different treatment.

The vector is a set of commands that define the processing of a call. For example, a call can be queued and then routed to another destination.

The following screen shows an example 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. Multiple vectors can be linked together to extend processing capabilities or to process calls to the same or different answering destinations. Any number of calls can use the same multiple vectors and process steps independently.

Understanding your goals and planning your system before you begin writing vectors is crucial. A planning guide is provided in Appendix O: Setting up a call center on page 825.

Benefits of Call Vectoring

Call Vectoring enables calls to be processed at a faster rate within an intelligent, real-time system, thereby providing appreciable cost saving to the user. The following table summarizes the benefits of Call Vectoring.

Call Vectoring Benefits	Examples
Call Treatment	
Implement special treatment based on the time of day, the day of the week, and for holidays (for example, routing calls to a different vector when one location is on holiday).	Customer service center example on page 59 Conditional branching on page 591 Example application - distributed call centers on page 66
Automatically change treatment according to either how long the call has been waiting or in response to changing traffic or staffing conditions.	Automated attendant example on page 60 Example application - mutual fund company on page 62 Example application - distributed call centers on page 66 Example application - help desk on page 68 About interflow routing on page 624 Using Call Prompting to route by collected digits on page 242 Using Call Prompting to branch by collected digits on page 243
Provide appropriate caller feedback during waiting. For example, music or announcements during heavy calling periods.	Delay announcement on page 540 Forced announcement example on page 541 Information announcement example on page 541 Call delay with audible feedback on page 647 Multiple music sources on hold on page 648 Call delay with continuous audible feedback on page 648
Provide multiple or recurring informational 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 example on page 59 Leaving recorded messages (VDN as the coverage point option) on page 611 About interflow routing on page 624
Provide 24 hour or day, 7 day or week automated information announcements.	Information announcement example on page 541 Call delay with audible feedback on page 647

Call Vectoring Benefits	Examples
Remove selected calls (by providing busy or disconnect).	busy command on page 545 Call disconnect example on page 580 Leaving recorded message on page 602 Unconditional branching example on page 591
Set up and test, in advance, special call treatments for events such as sales, advertising campaigns, holidays, snow days, and so on.	Information announcement example on page 541 Setting up a Holiday Table on page 360 Changing vector processing for holidays on page 362 (both examples)
Provide the caller with a menu of choices.	Example application - mutual fund company on page 62 Example application - help desk on page 68 Using Call Prompting to route by collected digits on page 242 Passing digits to an adjunct on page 247 Dial-ahead digit vector examples on page 250
Queue calls to up to three splits simultaneously, consequently improving the average speed of answer and agent productivity.	Customer service center example on page 59 Example application - distributed call centers on page 66 Multiple split queuing on page 609
Implement routing to local or distant destinations.	Customer service center example on page 59 Example application - mutual fund company on page 62 Example application - distributed call centers on page 66 Example application - help desk on page 68 About interflow routing on page 624 Using Call Prompting to route by collected digits on page 242 Using Call Prompting to branch by collected digits on page 243
Connect callers to a voice-mail or messaging system either automatically or per caller request.	Example application - mutual fund company on page 62 Leaving recorded messages (VDN as the coverage point option) on page 611 Leaving recorded message on page 602
Call Routing	
Reduce call transfers by accurately routing callers to the desired destination.	Example application - mutual fund company on page 62 Using Call Prompting to route by collected digits on page 242 Using Call Prompting to branch by collected digits on page 243
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 example on page 59 Example application - mutual fund company on page 62 Example application - distributed call centers on page 66

Call Vectoring Benefits	Examples
Reduce agent or attendant staffing requirements by: (1) automating some tasks; (2) reducing caller hold time; (3) having agents in one split service multiple call types.	Example application - mutual fund company on page 62 Information announcement example on page 541 Call delay with audible feedback on page 647 Using Call Prompting to route by collected digits on page 242 Dial-ahead digit vector examples on page 250
Information Collection	
Provide customized or personalized call treatment using information collection and messaging.	Automated attendant example on page 60 Example application - mutual fund company on page 62 Example application - help desk on page 68 Using Call Prompting to route by collected digits on page 242 Treating digits as a destination on page 242 Dial-ahead digit vector examples on page 250
Collect information for use by an adjunct or by agent display.	Example application - help desk on page 68 Passing digits to an adjunct on page 247
Collect caller-entered or customer database-provided CINFO digits from the network.	CINFO vector example on page 183

Call Vectoring applications

This section provides example applications of the Call Vectoring feature and includes the following topics:

- List of example applications on page 57
- Customer service center example on page 59
- Automated attendant example on page 60
- Data in/voice answer and data/message collection example on page 61
- Distributed call centers example on page 65
- Help desk example on page 67
- Insurance agency/service agency example on page 69
- Warranty service (with EAS) example on page 72
- Resort reservation service (with EAS) example on page 76
- Attendant routing example on page 80
- QSIG CAS example on page 83
- Night station service example on page 85
- Holiday Vectoring example on page 86
- Network Call Redirection example on page 88
- BSR using EWT and agent adjustments example on page 90
- Dial by Name on page 93
- Vectors exercises on page 96

List of example applications

Example applications and the primary feature that is illustrated are listed in the following table.

Example	Features used
Customer service center example on page 59	Basic Call Vectoring
Automated attendant example on page 60	Call Prompting

Example	Features used
Data in/voice answer and data/message collection example on page 61	Call Prompting, Basic Call Vectoring
Distributed call centers example on page 65	Look-Ahead Interflow, Basic Call Vectoring
Help desk example on page 67	Adjunct Routing, Call Prompting, Basic Call Vectoring
Insurance agency/service agency example on page 69	Basic Call Vectoring, Call Prompting, Rolling ASA, EWT, VDN Calls, and ANI Routing
Warranty service (with EAS) example on page 72	Basic Call Vectoring, EAS
Resort reservation service (with EAS) example on page 76	Basic Call Vectoring, Adjunct Routing, Call Prompting, EAS
Local attendant group access code on page 82	Attendant Vectoring
Incoming trunk calls to attendant group on page 82	Attendant Vectoring
Incoming LDN calls on page 83	Attendant Vectoring
QSIG CAS example on page 83	Attendant Vectoring
Night station service example on page 85	Attendant Vectoring
Holiday Vectoring example on page 86	Holiday Vectoring
Network Call Redirection example on page 88	BSR multi-site, NCR
BSR using EWT and agent adjustments example on page 90	BSR multi-site
Dial by Name on page 93	Basic Call Vectoring, Call Prompting

Customer service center example

The example scenario involves a customer service center that is open weekdays from 8 a.m. until 5 p.m. The center provides two separate telephone numbers. One number is for regular customers, while the other number is for 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 to the correct number, vector 22 is accessed. The first two steps of this vector 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 vector 29 is accessed, the caller is given the appropriate announcement twice (skills 1 and 3) and is then disconnected (step 3).

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 the queue, control is sent to step 12, which routes the call to the attendant. If less than 10 calls are in the due, the call is queued to the main split (step 4). If the call is not answered immediately, an appropriate announcement is provided (step 5), followed by a wait period (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.

A call that is placed by a non priority customer is processed by vector 21. 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 when too many calls are queued, but non priority calls route to a busy signal.

Automated attendant example

This example scenario shows the use of Automated Attendant, which is one of the applications that can be supported by the Call Prompting feature. Automated Attendant allows the caller to enter the extension of the party that the caller wants to reach. Depending on the parameters established, the user can enter up to 16 digits from a touchtone telephone.

Automated Attendant is usually used by call centers that do not have DID trunks and whose callers know the extension of the people they are calling. Because it reduces the need for live attendants, Automated Attendant reduces call center costs.

The following example shows an example of a vector that implements Automated Attendant.

Example application - automated attendant

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

Step 1 of this vector contains the **wait-time** command, which is placed before the **collect digits** command in step 2 to provide the caller with ringback in the event that a 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 he or she wants to reach (step 2). The **collect digits** command in step 2 collects the digits. Thereafter, the **route-to digits** command in step 3 attempts to route the call to the destination.

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

Data in/voice answer and data/message collection example

This example 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 also uses Call Prompting to determine the desired service. Three services are offered:

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

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

This example uses three other applications that can be supported by the Call Prompting feature:

- Data In/Voice Answer (DIVA) allows a caller to receive information on a topic that he selects at the prompt. The caller selects the desired topic by entering the appropriate digits.

- Data Collection provides a method of collecting digits from a caller. The requested digits comprise an official number of some sort. For example, a Social Security Number, and they help the system process the call more efficiently.
- Message Collection allows the caller to leave a recorded message instead of waiting for the call to be answered.

The four vectors shown below illustrate how the mutual fund company handles telephone calls. Typically, the vector should be 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're sorry.  All of our operators are busy at
         the moment. If you'd like to leave your name and
         telephone number so that we can get back to you,
         dial 1.]
      6. goto step 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're sorry, we cannot take
         your message at this time. You may continue to hold, or
         you can call back later.]
     12. goto step 4 if unconditionally
```

DIVA and data/message collection vector examples (continued)

```
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're sorry. All of our operators are busy at
         the moment. If you'd like to leave your name and
         telephone number so that we can get back to you,
         dial 1.]
      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're sorry, we cannot take
         your message at this time. You may continue to hold, or
         you can 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 upon the service desired. If the caller enters a digit other than 1, 2, or 3 mentioned, or if the caller fails to enter any digits within 10 seconds, then the command fails and the call is routed to the attendant (step 6). If the caller enters 1, 2, or 3 within 10 seconds, then the call is routed to the vector specified in the appropriate **route-to number** command, which appears in steps 3, 4, and 5.

For instance, assume that, when prompted, the caller enters 3 because he or she wants to learn about the net asset values of the company's funds. In such a case, the **route-to number** commands in step 3 and in step 4 fail, because in each case, the digit 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 for matches the one entered by the caller. Accordingly, the call is routed to VDN extension 1033, and vector processing continues in vector 13.

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

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

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

The **collect digits** command in step 2 of vector 12 first requests the caller to enter his or her 6-digit account number. The command then collects the digits that are entered by the caller. Whether or not the caller correctly enters the digits, the **queue-to split** command in step 4 queues the call. If an agent does not immediately answer the call, the standard announcement is provided in step 5 and, if necessary, a delay is provided in step 6. The announcement in step 7 provides the caller with the option of leaving a message instead of having his or her call wait in queue. The caller is instructed to enter 1 if he or she wants 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 message center split so that the caller can leave a message. If the connection is made, the caller first hears ringback and can then leave a message. If the connection is not made, the step is unsuccessful, and step 13 provides an announcement that indicates that a connection could not be made. Thereafter, the **goto step** command in step 14 sends call control back to step 6, which leads the caller back into the steps to leave a message.

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

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

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

This example involves two customer 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 10 or fewer queued calls in Denver, the LAI call attempt is accepted and serviced there. Otherwise, the call is denied and remains in queue in New York until an agent becomes available. The two vectors shown below illustrate the process.

Note:

For other examples of LAI, see Look-Ahead Interflow (LAI) on page 273. To learn how to integrate distributed call centers using multi-site Best Service Routing, see Best Service Routing (BSR) on page 297.

Example application - distributed call centers

```
SENDING SWITCH:
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 of our agents
          are busy. Please hold and you will be answered
          by the first available agent.]
       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 of our agents
          are still busy. Please hold and you will be
          serviced by the first available agent.]
       9. wait-time 60 seconds hearing music
      10. goto step 5 if unconditionally
      11. busy
RECEIVING SWITCH:
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 and you will be
          serviced by the first available agent.]
       5. wait-time 60 seconds hearing music
       6. goto step 5 if unconditionally
       7. disconnect after announcement none
```

In this example, vector 80 is on the sending switch from a call center in New York, while vector 81 is on the receiving switch at a call center in Denver.

In the sending switch, 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, an announcement (step 3) and music (step 4) are provided. If the call is still not answered at this point, step 5 places a LAI call attempt to the receiving switch, on which vector 81 resides.

Step 1 in the receiving switch 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 cannot service the call. In such a case, control is passed to step 7, which rejects the Look-Ahead Interflow call attempt. However, if the test in step 1 succeeds, the call is queued by the receiving switch 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 control is passed to the Denver switch, where vector processing continues at step 4.

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

Note:

To avoid confusing the caller, the treatment provided at the receiving switch should be consistent with the treatment that is provided at the sending switch. In the distributed call centers example, note that the caller hears music (and never ringback or silence) at the sending switch. Accordingly, music should be (and, in our example, is) featured at the receiving switch.

Help desk example

This example 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. For this application, the information that is provided in the Adjunct Switch Application Interface (ASAI) route request, that is, calling party number, called number, collected digits, is used to route the call to the most appropriate agent. Such an agent might be the one who last serviced the caller, or it might be the next available agent for the specific caller. Also, based on switch traffic conditions and the caller-entered digit, the call can be diverted to other destinations, such as other ACD splits, announcements, or switches.

The following vector shows the help desk application.

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.
    If you have a question about hardware, please dial 1.
    If you have a question about software, please dial 2.
    If you have a general question, please 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 this 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 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 he or she wants to reach (step 2). In step 2 of this vector, the caller is instructed to enter 1, 2, or 3, depending upon the service (hardware, software, general) that he or she desires. Thereafter, the **adjunct routing link** command in step 3 instructs the switch to send a Route request to the adjunct processor, which is connected to extension 2400. 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 (see Adjunct (ASAI) Routing on page 201). If 1, 2, or 3 is not entered, and if the adjunct does not return a route, the call is eventually routed to the 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. Let's assume the caller enters 1 and the **adjunct routing link** command succeeds. In such a case, if the caller is judged to be a prime hardware customer, the call might be routed to one of a handful of specific agents who are assigned to handle such customers. On the other hand, if the caller is judged to be a casual hardware customer, the call might be routed to a larger group of ACD agents before it is queued, or to an appropriate announcement.

Finally, assume that the caller enters 1 and that the **adjunct routing link** command fails. In such a case, the call is routed by the **route-to number** command in step 5, probably to a vector that queues the call or provides an appropriate announcement.

Insurance agency/service agency example

This example involves an insurance company call center. It handles calls from independent field agents, policy holders with claims, policy holders needing customer service, and several general service agency type 800 number client accounts. Each different type of call has its own 800 number that routes the calls to associated VDNs.

The following list describes the call center requirements.

- The independent field agents require fast service. They call the company to find out the latest rates for specific clients, to set up policies, to make adjustments, and so on. Often their clients are waiting as they call. Therefore the insurance company wants to maintain an Average Speed of Answer (rolling-ASA) of 30 seconds or less for field agent calls. These are the most important calls and are given high priority in queues.
- The calls to claims must be separated by area code. The claims agents receive different training 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 do not need to be tested individually and can be grouped in vector routing tables.
- The insurance company wants to give customer service callers an announcement indicating how long that they can expect to wait for service.
- The insurance agency is also selling 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 30 seconds or less for field agent calls. ANI Routing is used to partition calls based on area code and route the calls to the appropriate claims agents. EWT Routing is used to notify customer service callers of their expected wait time if it is longer 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 that are 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 more clearly demonstrate the features described in this example, the sample vectors do not include tests for unstaffed or full queues, out-of-hours operation and so forth.

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 of the backup splits and goes directly to the announcement in step 6. The assumption is that the call will be 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 greater than 30 seconds, steps 3, 4, and 5 check backup splits. The call is queued to any of these splits that have a rolling ASA of 30 seconds or less. If the call still is not answered by the time vector processing reaches step 8, then the backup splits are checked again.

The following vector example could be used 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 vector routing table referenced in the example shown above 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. It also provides access to an agent when ANI is not available and the caller did not enter an area code when prompted.

The following vector example notifies customer service callers of their expected wait time unless they will not have long to wait.

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 would wait more than 10 minutes are routed to a call back later announcement. step 4 routes callers to a VRU to be given the expected wait time announcement while they hold their 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 Vectors 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 of the example vectors shown above, 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) example

This example involves a major appliance company that offers one year warranties and extended warranties on its major appliances, such as dishwashers, refrigerators, washers, and dryers. The warranties are printed in English and Spanish to accommodate customers who speak and understand these languages. Naturally, callers need to speak with someone who is familiar with the appliances they have bought and who speaks the appropriate language. Accordingly, 800 numbers are provided for calling both English-speaking agents and Spanish-speaking agents. Bilingual agents with Spanish-speaking skills are hired so that they can back up the groups of English-speaking agents. Agents are trained first on all appliance models of a certain type and then on all appliance models for a room, such as the kitchen, the laundry room, and so forth.

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

Skill table 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 table 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

VDN skill table for the warranty service call center (continued)

VDN skill preference	Appliance	VDN	First	Second	Third
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.

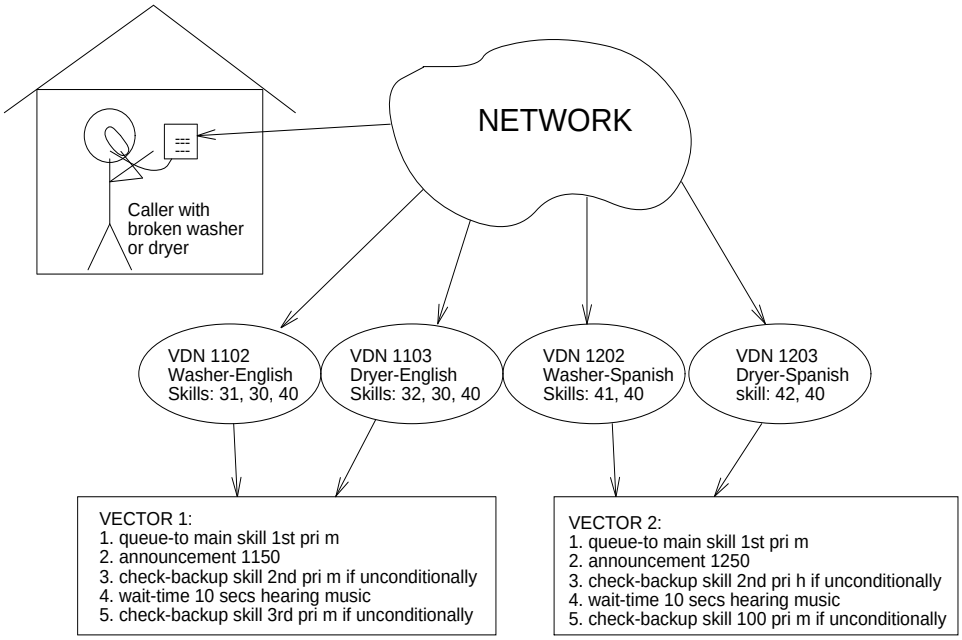
The goal of the warranty service call center is to answer 80% of the incoming calls within 20 seconds. Accordingly, if a call that is directed to a vector is not answered by the time the announcement finishes, a second group of agents is viewed, thus enlarging the agent pool. If the call is not answered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

Since the call center has only a few bilingual agents, the center's management wants to reserve these agents for Spanish-speaking callers. 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.

Warranty service call center (part 1) and Warranty service call center (part 2 show 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.

The following figure shows how the call comes into the network and is then directed to the appropriate VDN, which in turn points to the appropriate vector. For each VDN, the corresponding VDN skills are indicated.

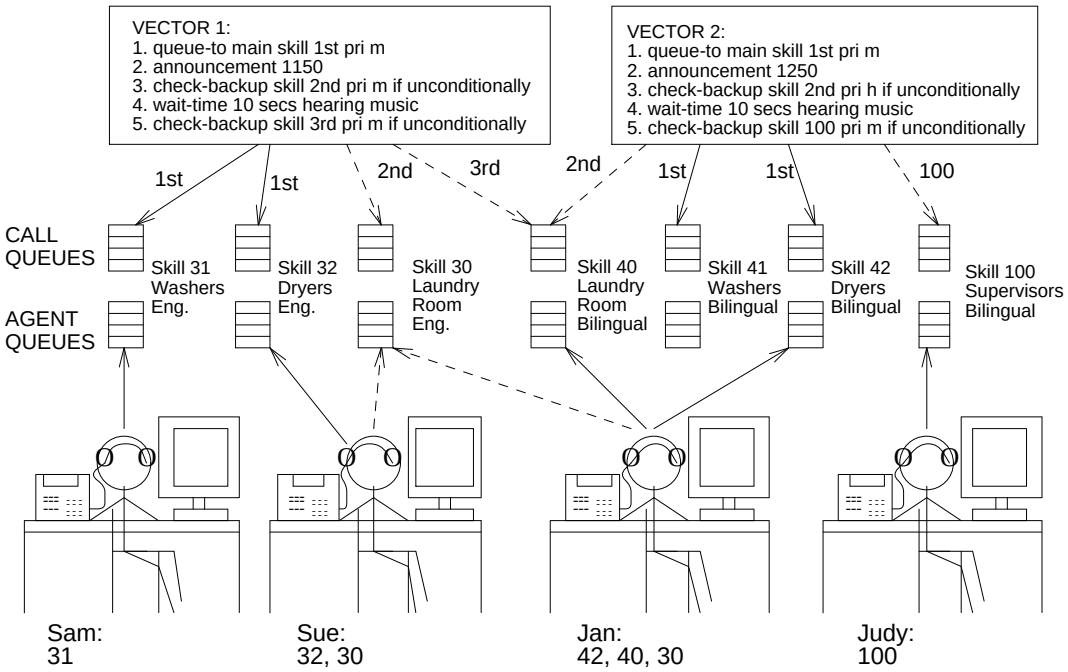
Warranty service call center (part 1)



The next figure shows 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 each agent's name. Dashed lines indicate backup or secondary skills.

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

Warranty service call center (part 2
)



Assume that a Spanish-speaking caller has a broken dryer and decides to call the warranty service call center. The caller dials the appropriate number. The call then enters the switch and is directed to VDN 1203, which points to Vector 2. As illustrated earlier, VDN skill preferences 42 (dryers) and 40 (laundry appliances) are administered as the 1st and 2nd skill preferences, respectively, for VDN 1203.

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 (Jan, for example) is available, this agent answers the call. If such an agent is not available, the appropriate delay announcement in step 2 is played. 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 (Jan, for example), that particular 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.

Resort reservation service (with EAS) example

This section includes the following topics:

- About resort the reservation service example on page 76
- Placing the reservation on page 76
- Specific number dialing on page 77
- General number dialing on page 78
- Call-back provisions on page 79

About resort the reservation service example

This example involves a resort company that places a variety of advertisements in magazines for information on a particular resort or state. Callers respond to these advertisements dial one of several numbers provided in the advertisement. A call center makes the reservations for the resort company. To satisfy the request of many callers to the service, an effort is made to have callers connected to an agent who has visited the resort they are interested in visiting. Also, the resort company has determined that it is easier to sell additional sightseeing packages if the agent has a regional accent.

Placing the reservation

To respond to an advertisement, the caller can dial a number that directly routes him or her 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 the Call Prompting feature. The following sections discuss these methods.

Specific number dialing

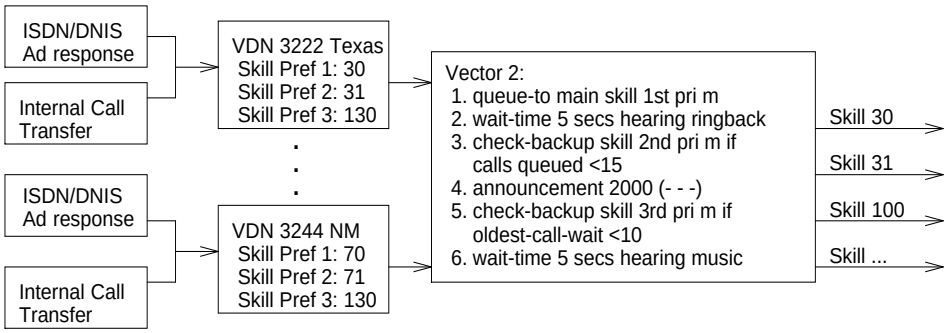
The call center is set up in such a way that a VDN with an accompanying set of VDN Skill Preferences is assigned to each state that has a resort. For example, the following table shows how Skill Preferences are assigned to Texas VDN 3222.

VDN 3222 skill preferences assignments for the resort reservation service

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 take a reservation

The following figure shows how a call to VDN 3222 can be processed by Call Vectoring.

Process involving specific number dialing



For this process, a single VDN for each state is assigned to Vector 2. Accordingly, the figure shown above shows the VDN and the associated VDN skills for two states, Texas and New Mexico.

Assume that a caller wants information on resorts in Texas and dials the appropriate number, for example, 615-3222. In this case, the call enters the switch and is directed to VDN 3222, which points to Vector 2.

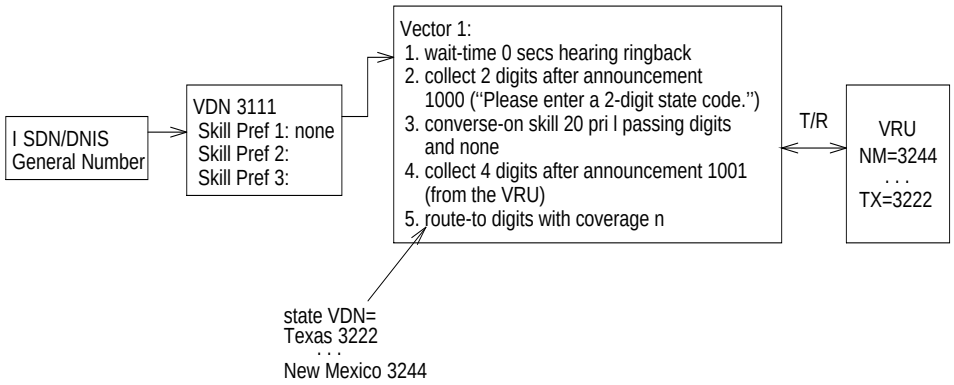
Once vector processing starts, the **queue-to skill** command in step 1 queues the call to the skill group that corresponds to the 1st VDN skill (30-Agent with a Texas accent who has visited resorts in Texas). If an agent with skill 30 is available, this agent answers the call. If such an agent is not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

This option allows the caller to dial the general number provided, for example, 615-3111. The caller is then serviced in part using the Call Prompting feature.

The following figure shows how a call to VDN 3111 can be processed using Call Vectoring.

Process involving general number dialing



After the number is dialed, the call is directed to VDN 3111, which points to Vector 1. Note there are no skill preferences assigned to VDN 3111. Also, VDN 3111 is the only VDN that is administered to point to Vector 1. Therefore, this VDN 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. Assume that the caller enters the correct code for Texas, which is 05. In this case, 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.

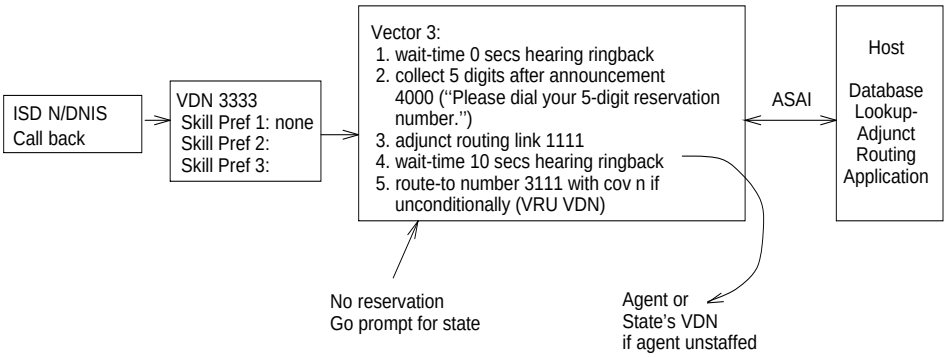
For more information about the **converse-on** command, see Basic Call Vectoring on page 103.

When the VRU port responds, the step then 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 the switch. 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. This process is discussed in the General number dialing section.

Call-back provisions

After a caller makes a reservation for a resort site, the caller is given a call-back number. Such a number is helpful if the caller needs more information or wants to check on some arrangement that was previously made. The following figure shows one approach for enabling call-back provisions.

Example 8: Call-back 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, this VDN is used for calls from all states.

The **collect digits** command in step 2 of the previous vector first requests the caller to enter his or her 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 the switch to send the collected digits (along with other information) to the host in the ASAI adjunct routing request. The host then uses these 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. Once this happens, information on the relevant reservation is displayed at the agent’s data terminal, thus providing quicker and more personal service. 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. This process is discussed in the General number dialing section.

Attendant routing example

This section includes the following topics:

- About attendant routing on page 80
- Vector administration on page 81
- Local attendant group access code on page 82
- Incoming trunk calls to attendant group on page 82
- Incoming LDN calls on page 83

About attendant routing

The following example shows how the Attendant Vectoring commands can be used to route calls in an attendant environment. For the attendant vectors, consider the following vectors and vector administration.

Note:
For the following vector examples, Tenant Partitioning is turned on:

Attendant Vectoring vectors

VDN 1999 vector 1	VDN 2999 vector 2	VDN 3999 vector 3
1. wait-time 0 secs hearing ringback 2. goto step 6 if time-of-day is all 12:00 to 13:00 3. queue-to attd-group 4. goto step 7 if queue-fail 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	1. wait-time 0 secs hearing ringback 2. queue-to attd-group 3. goto step 6 if queue-fail 4. announcement 9000 5. wait 999 seconds hearing music 6. disconnect after announcement 9001 7. queue-to hunt-group 1 8. goto step10 if queue-fail 9.wait 999 secs hearing ringback 10. busy 11. route-to number 93035381000 with cov y if unconditionally	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 7 if queue-fail 5. announcement 9000 6. wait 15 seconds hearing music 7. goto step 4 if unconditionally 8. queue-to attendant 6000 9. goto step 10 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 VDNs 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.

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 terminated to the attendant and vector processing terminates. Otherwise, the call is queued to the attendant group and the caller hears music from the 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, in the example above, if it is lunch time the call will route to the hunt group and all hunt group processing will then apply. 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 call is received on a trunk that has the attendant group assigned as the incoming destination or when the call is addressed to the attendant group, 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 queueing 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 cannot be queued, the call is routed to the remote location and vector processing terminates.

Note:

The main difference from the example shown in Local attendant group access code on page 82 is queueing the call to the hunt group rather than routing the call there. In this example, the call will not follow the hunt group's coverage path or night service destination.

Incoming LDN calls

When a call is received for an 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 into 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 attendant 6000's queue, 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 terminated 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 TN 3 every 15 seconds. If the call cannot be queued, it is sent to attendant 6000.

Note:
Vector 3 attempts to queue the call to attendant 6000. A **`route-to`** command could also be used, but care should be taken since an attendant cannot be assigned a coverage path.

QSIG CAS example

This example shows how you can use Attendant Vectoring with CAS.

This section includes the following topics:

- CAS branch on page 83
- CAS main on page 84

CAS branch

Suppose the call center always wants to play an announcement at a QSIG CAS branch before routing the call to the QSIG CAS main. In this case, assume that an attendant VDN needs to 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.

QSIG CAS vector main

```
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 TN of the incoming trunk.

Night station service example

This example shows how you can use the Attendant Vectoring features 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 example is a vector that is directing calls to special processing because of a holiday or special event. Holiday Vectoring is an enhancement that simplifies vector writing for holidays. It is designed for customers who need to reroute or provide special handling for date-related calls on a regular basis.

In this example, a commercial bank that is headquartered in Germany has branches in Europe. The bank recently established a U.S. presence by opening branches in the New York City metropolitan area. The bank's 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 European centers are bilingual and assigned to splits that handle calls from both English and German customers. The same is true for the agents in the New York call center. Because the New York call center is open 24 hours a day, it often takes calls that are routed from the Irish and German call centers after those centers close at 6:00 p.m. local time.

Due to the large number of bank holidays per year in Europe (up to 30 days), the Holiday Vectoring feature can be used to create vectors that distribute calls automatically on holidays. The call center administrator recommended this feature to the systems administrator to save the cost of time spent on writing vectors for date-related processing, and to save business that would be lost to abandoned calls if vectors are not readministered for holidays.

The following figure 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 had to 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              12    31               Christmas
    01    01  00   00     01    02  06   00       New Year’s Day
```

After submitting the Holiday Tables screen, the next step is to modify the vector processing for these holidays. On the Call Vector screen, enter the new goto conditional for the holidays.

Modifying a vector to route according to a holiday table

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

    Number: 3        Name: In Ireland
Multimedia? n     Attendant Vectoring? 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
```

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

Network Call Redirection example

This section includes the following topics:

- About the NCR example on page 88
- Primary Vector on page 89
- Status poll vector on page 89
- Interflow Vector on page 90

About the NCR example

This example shows the primary, status poll, and interflow vectors that redirect calls on the public network using the NCR feature.

Note:
This example assumes knowledge of multi-site BSR applications. For information about BSR, see Best Service Routing (BSR) on page 297 For information about NCR, see Network Call Redirection on page 379.

The e-Commerce company used in this example has three call centers. In an effort to reduce costs, the company has implemented Network Call Redirection (NCR) to redirect calls on the public network and reduce the trunking costs between the three switches. BSR is also implemented on the switches in order to increase the efficiency of agent utilization.

The e-Commerce 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, the switch is required to set up the second leg of the call and then 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 the switch. The benefit is that the switch retains control over the call and can redirect the call using the trunk-to-trunk method should the NCT invocation fail.

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 then drops both the original call and the second leg of the call at the redirecting switch.

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

The e-Commerce company has set up IP trunking to emulate ISDN PRI and will use this capability to poll remote sites for possible NCR. For information on setting up IP trunking to emulate ISDN PRI, see the *Administration for Network Connectivity for Avaya Communication Manager*.

The following sections give examples of how the vectors must be set up at each site to use the public network with NCR (as opposed to IP trunking) to route a call from one site to another. For information about administering BSR polling over IP, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Primary Vector

A call arrives at eCommerce location 1 and is processed by the primary vector. This vector begins the BSR process by considering the specified resources. The following Call Vector example shows the primary vector for incoming call processing at eCommerce location 1.

Primary vector

```
1.wait time 0 secs hearing ringback
2.consider split1 pri m adjust-by 0
3.consider location 2 adjust-by 30
4.consider location 3adjust by 10
5.queue-to-best
```

For this example, assume that location 2 returned the lowest EWT, so the call will be routed to that site.

Status poll vector

To collect information from the remote switch, 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 the switch at location 2. The following example provides an example status poll vector on the remote switch.

Status poll vector

```
1.consider split2 pri m adjust-by 0
2.consider split 11pri madjust-by 0
3.reply-best
```

The status poll only obtains information and returns it to the origin switch; the call is not connected to the status poll VDN. Once the remote switch has returned the necessary information, the **`consider`** series in the primary vector at location 1 can continue at the next vector step.

Interflow Vector

Once the switch has selected the site to which the call should be routed (location 2), the call is sent to the public network. The public network switch then sets up the second leg of the call and passes the codeset 0 UUI information in the SETUP message if this is supported. Next, the Avaya switch tells the public switch to transfer the call over the public network. The Avaya switch knows to do this because Net Redir for location 1, location 2, and location 3 was set to y on the BSR Application screen.

For incoming 800 number calls from MCI DMS-250 network switches, the vector reached by the second leg call placed by the switch must immediately be answered (and send an ISDN CONNect message). This can be accomplished with a **wait 0 secs hearing music** or an **announcement** step as the first step in the receiving interflow vector. The following example shows an example interflow vector for eCommerce location 2.

BSR example of interflow vector on remote switch

```
1. announcement83345
2. consider split 2 pri m adjust-by 0
3. consider split 11 pri madjust-by 0
4. queue-to best
```

The public network then merges the second leg of the call to the second site and drops the trunk to the Avaya switch.

BSR using EWT and agent adjustments example

This section includes the following topics:

- About the BSR using EWT and agent adjustments example on page 91
- Primary Vector on page 92
- Status poll vector on page 92
- Interflow Vector on page 93

About the BSR using EWT and agent adjustments example

In this 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 will use the **adjust-by** option in the **consider** vector step to select the best agent at any site to receive a call.

The catalog company uses the **adjust-by** command to consider delivery of calls based on adjusted idle times for the agents, so that a remote site is not selected when agent idle time differences are not significant.

To activate the BSR Available Agent Adjustment option, the administrator sets the **BSR Available Agent Adjustments** field on the Feature-Related System Parameters screen to y.

To use the option, the switch Administrator changes 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% for the second call center, and 20% for the third call center. If there is an agent surplus at two or more of the call centers, then the adjustment will apply. The adjustment makes sites more or less desirable, based on decreasing the idle time of available agents by the percentage assigned for the site.

Note:
If the actual agent idle time is 100 or more seconds, then the idle time is decreased by the assigned percentage. If the actual agent idle time is less than 100 seconds, then the idle time is decreased by the adjustment in seconds.

The following table summarizes how the above adjustment can affect the idle times for each site.

Idle time adjustment calculations

	Agent idle time	Adjust by xx%	Calculation	Adjusted idle time
incoming split 1 at location 1	40	0[1]	0	40
location 2	50	20	50 - 20 secs	30
location 3	100	20	100 - 20 secs (20% of 100)	80

1. Since the adjust-by value in this consider step is set to zero, no adjustment is made.

Primary Vector

An incoming call arrives at location 1 and is processed by the primary vector. This vector begins the BSR process by considering 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 split1 pri m adjust-by 0
3.consider location 2 adjust-by 20
4.consider location 3adjust-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 as shown in table Idle time adjustment calculations on page 91. Step 5 queues the call to the best location found.

Status poll vector

To collect information from the remote switch, the command **consider location 2 adjust-by 20** in the primary vector places a call (a status poll) to the status poll vector on the switch at location 2. The example status poll vector is shown below.

Status poll vector

```
1.consider split2 pri m adjust-by 0
2.consider split 11pri madjust-by 0
3.reply-best
```

The status poll only obtains information and returns it to the origin switch; the call is not connected to the status poll VDN.

This vector compares splits 2 and 11, identifies the better of the two, and sends this information back to switch 1 with the **reply-best** command. Notice that the **adjust-by** command could be used on the remote switch to adjust the EWT or agent idle time that is returned by either of the splits. When adjustments are applied at both the origin and remote switches, the two adjustments are added at the origin switch.

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 switch 1 contains the information from the best split considered at location 2. Once the remote switch has returned the necessary information, the consider series in the primary vector on switch 1 can continue at the next vector step.

Interflow Vector

Based on the values derived in table Idle time adjustment calculations on page 91, at each site, location 2 is the best site based on the adjusted agent idle time. The **queue-to best** command in the primary vector interflows the call to the interflow vector at location 2. The example interflow vector is shown below.

Interflow vector on remote switch

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

The interflow vector reconsiders the status of both splits to get the most current information and queues or delivers the call to the best split. Notice 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, it is removed from any queues at the origin switch and any audible feedback at the origin switch is terminated.

Dial by Name

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

Note:
The Dial by Name feature must be enabled 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 a Listed Directory Number), a vector routes the call to an announcement that says, *Hello. You have reached A1 Hotel. Please press 0 for the operator; press 1 for the front desk; press 2 if you know the guest's extension; press 3 if you know the guest's name; press 4 if you want to choose from a list of extensions; or press 5 if you wish to hear these options again.*
- When the caller selects 3, the caller is then instructed to enter the person's name.
- As soon as a single match is found, the call is placed to that person.

You can assign several vectors that define how calls will be handled as users select the different prompts. 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 ringback for 2 seconds.
2. Announcement 381 plays. This announcement asks the caller to do one of the following:
   - Press 0 if the caller wants the operator; if the caller presses 0 or waits for the timeout, the call is routed to the operator.
   - Press 1 if the caller wants the 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 person's extension; if the caller presses 2, the call is routed to announcement 382, which instructs the caller to dial the person's extension.
   - Press 3 if the caller knows the person's name; if the caller presses 3, the following sub-procedure occurs:
     1. Announcement 661 plays requesting that the caller enter the first four characters of the person's last name.
        - If there is a single match, the call is redirected.
        - If there are multiple matches, continue with 2.
        - If there is no match, go to 4.
     2. Announcement 662 plays requesting that the caller enter the rest of the person's last name, followed by the **#** key.
        - If there is a single match, the call is redirected.
        - If there are multiple matches, continue with 3.
        - If there is no match, go to 4.
     3. Announcement 663 plays requesting that the caller enter the first two characters of the person's first name.
        - If there is a single match, the call is redirected.
        - If there is no match, continue with 4.
     4. Since there are still no matches, announcement 660 plays telling the caller that he or she can press 1 to try again, or press 0 to get an operator.
   - Press 4 if the caller knows the department (such as housekeeping) that he or she 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 over again; if the caller presses 5, the caller hears announcement 381, which repeats all of the options.
   - If the caller dials anything else, the call is routed to the operator.

Vectors exercises

This section presents several typical business scenarios that involve telephone use. One or more vectors are provided that show how to handle each of these scenarios.

The vectors presented here are intended to be suggested solutions. Individual call centers must consider their own unique requirements and budget in selecting and writing vectors.

This section includes the following topics:

- Emergency and routine service on page 96
- Late Caller Treatment on page 99
- Messaging option on page 101

Emergency and routine service

Write a vector that does the following:

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

Suggested solution 1

Call Vectoring option

```
 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're sorry.  All of
  our operators are busy. Please hold.]
 8.wait-time 10 seconds hearing music
 9.goto step 7 if unconditionally
10.disconnect after announcement 4200 [We're sorry.
  All of our operators are busy at the moment. Please call back at
  your convenience.]
```

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

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

Suggested solution 2

Note:
This example uses the Call Prompting feature. For more information about Call Prompting, see Call Prompting on page 237.

Call Vectoring and Call Prompting option

```
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.
         Otherwise, please hang up now.]
      3. route-to number 1031 with cov y if digit = 1
      4. announcement 3100 [Entry not understood.  Please
         try again.]
      5. goto step 2 if unconditionally

VDN (extension=1031   name="Service"   vector=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're sorry.  All of
         our operators are busy. Please hold.]
      7. wait-time 10 seconds hearing music
      8. goto step 6 if unconditionally
      9. disconnect after announcement 4200 [We're
         sorry. All of our operators are busy at the moment.
         Please call back at your convenience.]
```

Suggested Solution 2 involves both Call Vectoring and Call Prompting. Also, it involves two vectors instead of just one vector, and it assumes the that caller is calling from a touchtone telephone. The announcement portion of the **`collect digits after announcement`** command in step 2 of Vector 30 first provides the caller with the appropriate emergency information. It then invites the caller to press 1 if the caller is calling for some other reason. If this is not the case, it finally suggests that the caller hang up.

Assume that the caller wants to hold the line but enters the incorrect touchtone digit (2, for example). In such a case, the **route-to number** command in step 3 attempts to route the call to VDN extension 1031 according to the entered digit. However, because a number other than 1 was entered, the call is not routed to the VDN extension. Instead, control is passed to step 4, where the **announcement** command first informs the caller of the input error and then invites the caller to try again. Thereafter, the **goto step** command in step 5 unconditionally sends control back to step 2, where the **collect digits** command ultimately collects the digit 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 Caller Treatment

The call center is staffed by union agents who work under a contract that stipulates that agents are free to leave promptly at 5:00 p.m. However, you are concerned about the callers who will call shortly before 5:00 p.m. on any given day and find themselves waiting in queue the at the top of the hour.

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

Suggested solution:

Late caller 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 [It is almost closing time.
   We will try to service you before we close for the day.
   However, if we are unable to do so, please call back
   at your convenience between 8:00 A.M. and 5:00 P.M.,
   Monday through Friday.]
11.wait-time 30 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're sorry, our office is now closed.
   Please call back at your convenience between 8:00 A.M.     and 5:00 P.M.,
   Monday through Friday.]
15.disconnect after announcement 400 [We're sorry, our office is     closed.
   Please call back at your convenience between 8:00 A.M.     and 5:00 P.M.,
   Monday through Friday.]
16.disconnect after announcement 500 [We're sorry, we cannot service your
   call at this time. Please call back at your convenience between
   8:00 A.M. and 5:00 P.M., Monday through Friday.]
```

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

The **goto step** command in step 1 checks whether the call is 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 whether 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 whether the current time is any time 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 the appropriate hold message, and the **wait** command in step 8 provides the caller with 998 seconds of music. Finally, the **stop** command in step 9 halts vector processing, and the call remains in queue until either the agent answers the call or the caller hangs up.

If the current time is 4:45 p.m. to 5:00 p.m. Step 5 executes a branch to step 10, where the appropriate late caller announcement is provided to the caller. Thereafter, the **wait-time** command in step 11 provides the caller with 30 seconds of music. Control is then passed to step 12, where the **goto step** command checks whether the time 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 at least a second after 5:10 P.M., control is passed to step 14 as described previously.

Messaging option

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 (if possible)
  - Delivers to the caller the following personalized messaging system statement: *All of our MegaSports agents are busy...Please leave your name and telephone number.*
- Plays 30 seconds of ringback for the caller
- After the ringback, plays an announcement for the caller that is followed by music

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're sorry, all of our
  MegaSports agents are busy. If you'd like to leave a
  message, please do so after the tone. Otherwise, please
  call back between 8:00 A.M. and 5:00 P.M, Monday through
  Friday. Thank you.]
 9.messaging split 20 for extension 4000
10.disconnect after announcement 2050 [We're sorry, we are unable
  to take your message at this time. Please call back
  between 8:00 A.M. and 5:00 P.M., Monday through Friday.
  Thank you.]
```

The **goto step** command in step 1 of the example shown above checks whether the oldest call waiting in split 50 has been waiting for 75 seconds or more. If so, control is passed to step 8, where the **announcement** command first informs the caller that all of the agents are busy and then invites the caller to either call back at the appropriate time or leave a recorded message for the agent. If the caller 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 that he or she 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 that provides the caller with the appropriate apology and subsequent directives. If the caller is successfully connected to AUDIX, vector processing terminates, and a message 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 for 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.

Basic Call Vectoring

The vector commands that are available to you as part of the Basic Call Vectoring feature set are the simplest and most common commands that are used to program call vectors.

This section includes the following topics:

- Command set on page 103
- General considerations for Basic Call Vectoring on page 105
- Types of Basic Call Vectoring commands on page 104

Command set

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

Treatment steps	Command
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**

Types of Basic Call Vectoring commands

This section includes the following topics:

- Treatment commands on page 104
- Routing commands on page 105
- Branching or programming commands on page 105

Treatment commands

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

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

For more information about these commands, see Call Vectoring commands on page 521.

Routing commands

Basic Call Vectoring includes routing commands that enable you to various destinations and treatments. Basic Call Vectoring includes the following routing commands:

- queue-to-split and check split commands
- messaging split or skill command

For more information about these commands, see Call Vectoring commands on page 521.

Branching or programming commands

Basic Call Vectoring provides programming methods that can be used within a vector either to create branching patterns in call processing flows, or stop vector processing. Branching or programming commands include:

- **goto step** and **goto vector** commands
- **return** command
- **set** command
- **stop** command

For more information about these commands, see Call Vectoring commands on page 521.

General considerations for Basic Call Vectoring

You should understand the following items when you use Basic Call Vectoring:

- For ease-of-use purposes, each specific vector function or operation should be included in a separate vector and linked with one or more **goto vector** commands.
- To keep down service costs, vector commands should be designed so that answer supervision is delayed as long as possible.
- Always provide callers with initial feedback, such as ringback.
- Direct agent calls deserve careful attention because they can affect call queuing. Queue slots occupied by direct agent calls are not always counted in Avaya CMS and BCMS reports. For example, a direct agent call is never counted toward the total of queued calls within a split, and the **calls-queued** test condition has no effect on this type of call. For more information, see Appendix H: Call Vectoring/EAS and BCMS/CMS interactions on page 751.

- You can 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. For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Variables in Vectors

This section includes the following topics:

- About VIV on page 107
- Variable definition parameters on page 108
- Implementing vector variables on page 109
- VIV job aid on page 110
- Command syntax for vector variables on page 111
- VIV requirements on page 119
- Understanding local and global variables on page 119
- System-assigned vector variable types on page 121
- User-assigned vector variable types on page 130
- VIV interactions and considerations on page 135
- VIV administration on page 135
- VIV vector examples on page 139
- Troubleshooting vector variables on page 149

About VIV

Variables in Vectors (VIV) is a Call Vectoring feature introduced in Avaya Communication Manager 2.0. The VIV feature allows you to create variables that can be used in vector commands to:

- Improve the general efficiency of vector administration.
- Provide increased manager and application control over call treatments.
- Allow you to create more flexible vectors that better serve the needs of your customer and call center operations.

The vector variables are defined in a central variable administration table. Values assigned to some types of variables can also be quickly changed by means of special vectors, VDNs or FACs (Feature Access Codes) that you administer specifically for that purpose.

Different types of variables are available to meet 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, an administered variable can be reused in many vectors.

Variable definition parameters

Variables in Vectors enhance Call Vectoring to allow letters (A to Z and AA to ZZ) to be used as either conditionals or thresholds or both in many commands. These letters are variables that can be defined by the customer for customizing vector programming.

You administer vector variables in a centralized administration table in which the variables are given alphabetical designations that can range from A to Z and AA to ZZ (up to 702 variables). Each variable can have only one definition. Once defined, the letters have the same type and assignment characteristics for every vector in which they are used. Depending on the variable type, you specify some or all of the following parameters when you create a new vector variable:

Variable type: VIV provides a number of different variable types that you use for different purposes. The kinds of information that are associated with a variable can be directly call-related, such as the active vdn for the call, asaii user information data, or the time of day at which the call is received.

Other types of variables allow you to assign your own user-defined values and use them as signals to impose high-level control over call processing operations. For example, you can use a single-digit value variable to test for operational states that are specific to your call center operations. For more information about the different types of variables, see System-assigned vector variable types on page 121.

Scope: The scope of a variable indicates how variable values are assigned and used in vectors in which the variable appears. Variable scopes can be either *local* or *global*. Local variables use data associated with a call and only apply within the vector. Global variables are system wide and apply to all vectors in which they are used. For more information, see Understanding local and global variables on page 119.

Length: Some variables require you to 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 Length specification, you also need to specify a Start position that specifies the beginning digit position of the digit string to be assigned to the variable. This along with the Length specification allows assigning only a portion of the data available to the variable.

Assignment: If you use a variable that has a user-defined value, you provide the value in the **Assignment** field of the variables administration table.

Variable Access Code (VAC): When you define a *value* variable, you have the ability to set up a Feature Access Code (FAC) that is associated with the variable so that you can dial into the FAC and set or reset the variable assignment. For more information about this capability, see VIV interactions and considerations on page 135.

Implementing vector variables

Administering variables and implementing them in your vectors is a relatively simple process:

Define the variable application: Determine how you intend to use the new variable and identify its defining characteristics. Use that information to identify a variable type that meets your needs. For a quick overview of variable types and purposes, see VIV job aid on page 110 and for more detailed descriptions, see System-assigned vector variable types on page 121.

Administer the variable: From the system administration terminal enter a **`change variables`** command to bring up the Variable for Vectors administration table. Look in the table and select any unused letter between A - Z. This letter is used to represent the variable in vector steps. In the table row for the letter that you have selected, enter the following information in the specified fields:

1. **Description** - A short description of your variable.
2. **Type** - The variable type.
3. The **scope** (local or global), **length** of the digit string, digit **start** position and **assignment**.

   Note:
   Depending on the variable type that your choose, some of these fields may be predefined or not applicable.

4. **VAC** - (optional, value variables only). If you administer a value variable type and want to be able to use a dial procedure (within the local switch only) to change the variable assignment using a Feature Access Code (FAC), do the following:

   a. From the system administration terminal, use the **`change feature-access-codes`** command to go to page 6 of the Feature Access Code screen and do the following.

      1. Select an unused FAC and note the Vector Variable feature access code number (VVx) that is associated with the FAC. Possible VVX values range from VV1 to VV9.
      2. In the **Code** field, provide the digits that you want to dial when you access the FAC.

   b. 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.

For more detailed information, see VIV administration on page 135.

Program vectors: Program one or more vectors 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 on page 111.

Change variable assignments: Some variables, such as the asaiuui 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 the communication server.

However, other variable types allow you to change the variable assignment as necessary, even as calls are being processed. For example, if you use a collect variable in a vector step, a caller changes the value assigned for the variable when they are prompted by an announcement and enter new digits.

Note:

When collect variables are provided specifically for supervisor or manager use, the collect variable usually has a global scope, and the variable is applied in a special vector intended for the supervisor or manager. For more information about this strategy, see the example at collect command with vector variables on page 113.

For descriptions of a few basic ways that you can apply variables in your call vectors, see VIV vector examples on page 139.

VIV job aid

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

Items in bold are default values that cannot be changed.

Variable type	Description	Scope	Specification	Max digit length	Assigns
ani	Tests the caller's phone number	**L**	Start digit position and Length	16	Incoming call data
asaiuui	Processes call-specific user data associated with the call	**L**	Start digit position and Length	16 out of a total of 96	Incoming call or ASAI application data

Variable type	Description	Scope	Specification	Max digit length	Assigns
collect	Processes collected digits for user-defined control, routing, or treatment	L or G	Start digit position and Length	16	The for parameter of the collected digits command or assignment in the variables table
tod	Holds the current time of day in 24-hour time for processing	**G**	None	Always 4	The main server system clock - for example, 0219 = 2:19 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	Counts the number 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	7	Routing for a call
vdntime	Provides the time in seconds that a call has been in vector processing by the call center	**L**	None	Always 4	Time in vector processing including prior processing for a call routed by BSR/LAI

Command syntax for vector variables

This section includes the following topics:

- announcement commands with vector variables on page 112

- collect command with vector variables on page 113
- converse-on command with vector variables on page 113
- disconnect command with vector variables on page 114
- goto commands with vector variables on page 115
- route-to command with vector variables on page 117
- set command with vector variables on page 118
- wait command with vector variables on page 118

announcement commands with vector variables

Variable syntax for these commands is supported beginning with Communication Manager 3.0. You can enter a vector variable between A-Z and AA-ZZ (up to 702 variables) as an announcement extension in all commands that use an announcement in the extension field.

The following syntax rules apply when vector variables are used with **announcement** commands.

```
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 for using vector variables in announcements

The requirements for using vector variables after the announcement extension are:

- 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 based on the type assignment administered in the Variables for Vectors table and used as the announcement extension number.
- The number must be a valid announcement extension assigned on the Audio/ Announcement screen.

See also:

- announcement command on page 538
- announcement commands with VDN variables on page 152

collect command with vector variables

Vector variable syntax for this command is fully supported for Communication Manager 3.0 or later.

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:

- When **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 are 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 = 1 or more, 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.

For information about using variables after an announcement extension, see announcement commands with vector variables on page 112.

converse-on command with vector variables

Variable syntax for this command is supported for Communication Manager 2.1 or later.

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

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

1. 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.

Requirements and considerations

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

- You can use a variable as a data type in both passing fields. This results in out-pulsing 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. For details, see Data passing on page 808.
- 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 out-pulsed 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.

See also:

- converse-on command on page 566
- converse-on command with VDN variables on page 153

disconnect command with vector variables

Variable syntax for this command is supported beginning with Communication Manager 3.0. You can use vector variables with the disconnect command after the announcement extension. For more information about using vector variables after an announcement extension, see announcement commands with vector variables on page 112.

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

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

See also:

- disconnect command on page 579
- disconnect command with VDN variables on page 154

goto commands with vector variables

The following syntax rules apply when using vector variables with **goto** commands.

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

goto step *1-99* **if** **or** **goto vector** *1-2000* **@step** *1-99* **if**						
ii-digits	**>,>=,<>,=,<,<=**	*A-Z, AA-ZZ*				
	in table	*A-Z, AA-ZZ*				
	not-in table					
interflow-qpos	**>,>=,<>,=,<,<=**	*A-Z, AA-ZZ*				
oldest-call-wait	**in skill**	*hunt group*, skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**>** **>=** **<>** **=** **<** **<=**	*A-Z, AA-ZZ*
rolling-asa	**for skill**	*hunt group*, skills for VDN: **1st**, **2nd**, **3rd**	**>** **>=** **<>** **=** **<** **<=**	*A-Z, AA-ZZ*		
staffed-agents	**in skill**	*hunt group*, skills for VDN: **1st**, **2nd**, **3rd**	**>** **>=** **<>** **=** **<** **<=**	*A-Z, AA-ZZ*		
V1-V9	**>,<,=,<>,>=,<=**	*A-Z, AA-ZZ*				
	in table	*A-Z, AA-ZZ*				
	not in table					
wait-improved for	**best**	**>,>=,<>,=,<,<=**				*A-Z, AA-ZZ*
	skill	*hunt group*, skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**>** **>=** **<>** **=** **<** **<=**	
	split	*hunt group*				

Requirements and considerations

The requirements and considerations for using vector variables with the **goto** commands are:

- A vector step that uses variable parameters could display command syntax like the following example, which tests the current number of counted calls for the active vdn to user-defined variable **G**:

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

- Depending on the type of variable 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.

See also:

- goto step and goto vector commands on page 583
- goto commands with VDN variables on page 154

route-to command with vector variables

Variable syntax for the **route-to** command is supported for Communication Manager 2.1 or later.

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

route-to number	[*A-Z, AA-ZZ*] **~r**[*A-Z, AA-ZZ*][1]	**with cov**	**y, n**	**if**	**digit**	**>, >=, <>, =<, <=**	*0-9*, **#**
					interflow-qpos	**<, =, <=**	*1-9*
					unconditionally		

1. When the specified number is preceeded by ~r, Network Call Redirection is attempted. For more information, seeUsing route-to number ~r vector step to activate NCR on page 393 and Using vector/VDN variables with route-to number ~r to activate NCR on page 394.

Requirements and considerations

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

- 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.
- If the variable is not defined, 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 obtained from the string of digits of the variable's current value 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.

See also:

- route-to command on page 619
- route-to command with VDN variables on page 156

set command with vector variables

This command is supported beginning with Communication Manager 3.0.

The following syntax rules apply when vector variables are used 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 and AA-ZZ collect vector variable. The collect variable type can be either global or local. Only global and Local Collect variables can be assigned. The others can be used as the operands but not assigned.
Operand1 Operand2	• User-assigned A-Z and AA-ZZ collect vector variable. The collect variable type can be either global or local. • System-assigned A-Z and AA-ZZ vector variables, such as: ani, asaiuui, doy, and so on.

See also:

- set command on page 632
- set command with VDN variables on page 157
- System-assigned vector variable types on page 121
- User-assigned vector variable types on page 130

wait command with vector variables

Variable syntax for this command is supported beginning with Communication Manager 3.0. You can use vectoring variables with this command. For more information about using vector variables after an announcement extension, see announcement commands with vector variables on page 112.

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

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

See also:

- wait-time command on page 645
- wait command with VDN variables on page 158

VIV requirements

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

The MultiVantage G3 Version field 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**.

Understanding local and global variables

This section includes the following topics:

- Definition of local and global variables on page 119
- About local variables on page 120
- About global variables on page 120

Definition of local and global variables

Variable conditionals can be either *local* or *global* in terms of the scope of their functionality in vectors. Depending on the variable type, the scope is global only, local only, or either local or global.

Local scope: When a variable has a local scope, its value is assigned on the basis of call-specific information and applies only in the vector that is currently processing 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 assumes the unique ASAI user data value for each new call that is processed by that vector.

Global scope: Global variables have system-wide values that apply to all vectors in which they are used. For example, the value specified for a *tod* (time of day) variable is provided by the system clock. Though this value changes each minute, the value provided at any given moment is identical in all vectors in which the variable appears.

For other variables that can have a global scope, such as *collect* or *value* variables, the value for the variable is user-defined by a call center supervisor or administrator. In this case, the user-defined value applies to all vectors in which the global variable may appear. The ability to administer vector variables with user-defined values that can be applied in a system-wide manner gives call center supervisors the ability to control call center resources and operations in a manner that is more precise and flexible than would otherwise be possible.

About local variables

You should understand the following items about local variables:

- When a variable is administered with a local scope, the value assigned to the variable is provided from information that is specific to that call. This value can be provided by:
  - Digits collected from the caller
  - The call VDN
  - ASAI data

  Note:
  ASAI data for a call can be modified by a CTI adjunct when a route-to adjunct command is used. For more information, see asaiuui type variable on page 122.
- When a value for a local variable is assigned to a call, that value remains with the call through all subsequent vector steps until the call is terminated or modified by another vector command or CTI adjunct. This rule applies to vector steps that include **goto** commands and to vector steps in chained vectors that may be included in the call processing sequence.

About global variables

You should understand the following items about global variables:

- Some types of global variables require you to assign values to them. The value that you assign applies to all vector steps in which the variable is applied, and when you change the value, the change is instantly propagated to all vector steps in which that variable is applied. This rule applies to all global variable types that allows input in the **Assignment** field in the Variables for Vectors administration screen. For more information, see Required variable administration entries on page 137.

  Note:
  Some variable types allow you to use a FAC or VDN to change the specified value. When you use either of those methods to change a variable value, the **Assignment** field in the Variables for Vectors administration screen is immediately updated to reflect the new variable value.

- Other types of global variables use dynamic values for which you cannot assign specific values. This rule applies to any global variable type that does not allow input in the **Assignment** field of the Variables for Vectors administration screen, such as the time of day and day of week variable types. For more information, see Required variable administration entries on page 137.

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 have administered.

The different system-assigned variable vector types are described in the following sections:

- ani type variable on page 122
- asaiuui type variable on page 122
- dow type variable on page 124
- doy type variable on page 125
- stepcnt type variable on page 125
- tod type variable on page 127
- vdn type variable on page 127
- vdntime type variable on page 129

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:

- The switch clock
- The data associated with the call - such as asaiuui, ani, and so on
- The processing of the call - such as stepcnt and vdntime

ani type variable

This variable provides expanded testing of the caller's phone number. When customers know who called, they can reroute the call based on the caller's area code, prefix, or suffix.

Scope

The scope for the ani variable is only 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	Where the number is routed.	collect	L	10	1

Variable A concatenates the incoming call to an area code. For example, if the calling ANI = 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, it can be modified or changed by an adjunct after an adjunct-route vector step. 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

You should also understand the following items about the asaiuui variable:

- A start position must be specified for the asaiuui variable.
- A length value must be administered 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, than any digits that extend the specified length are not included in the assigned value.

Example 1

The following example shows a vector step that compares an administered asaiuui variable **D** to a four digit segment of the ASAI user information string that should receive 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 2

The following vector example shows how an asaiuui variable can be used to provide selective customer treatment based on call-specific information.

In this example, a business wants to identify platinum member customers and provide them with special call treatment by queuing them at a higher level of priority. In this scenario, ANI data and other digits dialed by the caller are used by a CTI adjunct application 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.

Based on the account number constraints described above, the specifications that you would provide in the for Variables for Vectors screen for the asaiuui variable are shown in the following table:

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

The following example shows how the administered asaiuui variable can be applied 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 shown above, 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 it 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 the communication server. For information about setting and maintaining the system clock, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Scope

The scope for the dow variable is global only.

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 this variable are obtained from the system clock on the communication server. For information about setting and maintaining the system clock, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Important:
You should also understand the following items about leap years and doy variables:

Leap years include an extra day (February 29). Therefore, any 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 such doy variables are included in vectors that are initially set up in leap years, they 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 will fail.

Scope

The scope for the doy variable is only global.

Example

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

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

This 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 step count (stepcnt) variable tracks the number of vector steps. Before the number of vector steps reaches its maximum number, the call can be rerouted instead of dropped. The stepcnt variable can also be used as a loop-control variable. By monitoring the number of vector steps, customers can:

- Reroute calls before the calls have reached the maximum limit for their system and prevent calls from getting dropped.

- Reroute calls after an action has reached a pre-determined 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 this variable type anywhere other vector variables or VDN variables are used.
- Use this variable type as a threshold, conditional, or destination or data number, where supported.

Scope

The scope for the stepcnt variable is only local.

Example

The following vector example shows how you can 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 990 or more vector steps, an announcement is played 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 24-hour time. The assigned value, which can range from 0000 to 2359, is obtained from the communication server clock. The values assigned to this variable are obtained from the system clock on the communication server. For information about setting and maintaining the system clock, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

The communication server 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. If you want to check when the tod variable is after 12:30AM, you should compare to 0030, not 30.

Scope

The scope for the tod variable is only global.

Example

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

```
goto step 32 if D >= 1655
```

This 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 could be an announcement step indicating that the call center has closed.

vdn type variable

The vdn variable applies call-specific VDN information in a way that allows you to create vectors that are more versatile and reusable. When a vdn variable is used in a goto step, the extension number value that is 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 that is currently in control of the call process, and the active value represents the extension number of the current VDN, as it is defined by VDN override settings.

You specify whether the active or latest value should be applied to vdn variables when you administer the variable. For more information, see VIV administration on page 135.

Scope

The scope for the vdn variable is only local.

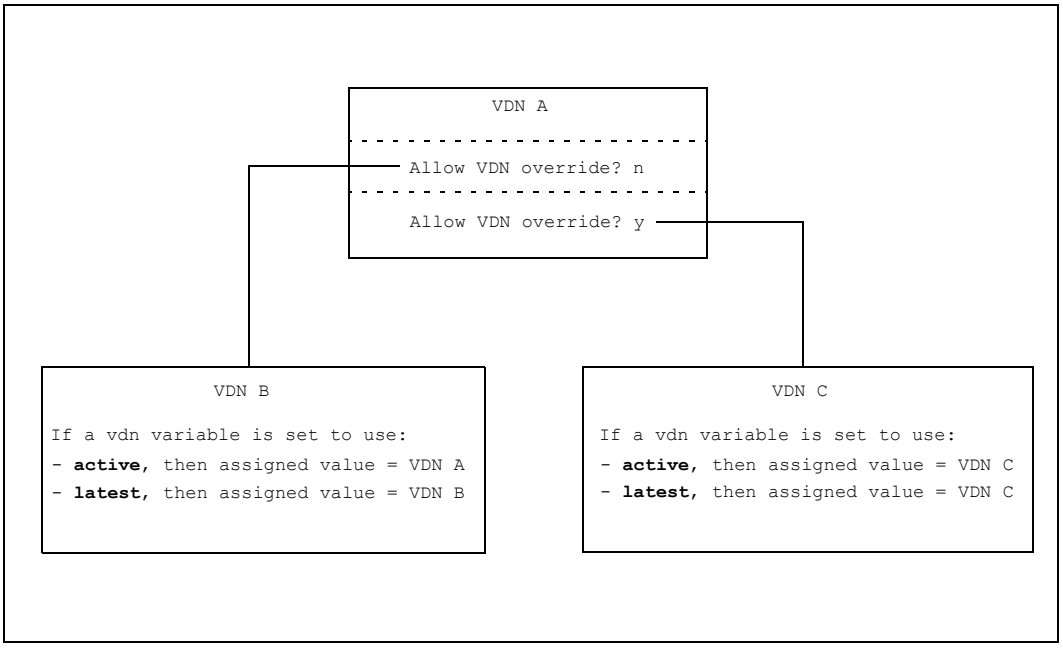
Additional information

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 administration 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.

Interactions between vdn variable assignments and VDN override settings



For more information about VDN Override settings, see VDN Override on page 39.

Example

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

```
goto step 5 if G=4561
```

vdntime type variable

The vdntime variable tests the time a call has been processed by the call center including any prior time spent in a remote Communication Manager system. Administrators can 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 Look Ahead Interflow or Best Service Routing, the time spent in the prior system is included.

Scope

The scope for the vdntime variable is only local.

Example 1

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	Time the call has been processed.	vdntime	L

In step 5, if the T variable is greater than 300 seconds, or 5 minutes, this 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 2

You can use this same approach in Example 1 with BSR Local Treatment vectors to break out of 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 on .... can be expanded for 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.

The different user-assigned variable vector types are described in the following sections:

- collect type variable on page 131
- value type variable on page 133

User-assigned definition

This section describes the user-assigned vector variable types. You can change the value of user-assigned vector variables. By contrast, the values for system-assigned vector variables are defined by the system clock, data about the incoming call, or by the processing of the call.

collect type variable

The collect type variable is used with 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 any collected data is assigned.

Syntax

The basic syntax 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 variable administration table.

Note:
Use of variables with collect commands is not required. The default entry that follows the **for** parameter is **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
```

For a complete description of the collect variable syntax used with the **collect** command, see collect digits command on page 556. For vector examples that show how the collect variable can be used, see Example on page 132.

Scope

The scope of collect variables can be either local or global. The following rules apply:

- If the scope is local, the assigned value is null until a value is provided by the call (or an adjunct) with a **collect digits/ced/cpd for** *[A-Z, AA-ZZ]* vector step. The assigned value is retained through all further call processing steps, including any chained vectors and route-to VDN commands, until the call is terminated or a new value is reassigned by subsequent **collect digits/ced/cpd for** *[A-Z, AA-ZZ]* vector steps.

- 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. For more information about how to set up a VDN and vector to facilitate changes in Global Collect variable values, see the example at collect command with vector variables on page 113 and Example application using time and day variables on page 139.

Additional information about the collect variable

You should also understand the following items about the collect variable:

- When collected data 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.

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 give-away campaign so that extra resources are shifted to more profitable sales campaigns during peak call volume hours.

Note:
For a different application of a collect variable in a vector application, see Example application using time and day variables on page 139.

In this example, a collect variable is used as a threshold to specify the number of calls allowed for the give-away 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 would provide in the for 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 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 give-away 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.]
```

value type variable

The value variable type gives you the ability to quickly change vector applications from one operational mode to another. To implement value variables, you need to do the following:

- Administer a value variable in the variable administration table. For more information, see VIV administration on page 135.
- You can administer a Feature Access Code and associate it with a Variable Access Code (VAC) if you want to use a dial code procedure to change a variable value assignment. VAC designations VV1 through VV9 are provided for this purpose on the FAC screen. For more information about how to set up a FAC to use with a value variable, see Optional FAC administration for value variables on page 138.
- 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. Otherwise, if the variable is not associated with a FAC, you must change the variable assignment in the Variables for Vector administration screen.

Scope

The scope of value variables is only global.

Reason to use the value variable

Before the value vector was available, call centers used a dummy agent logged into a dummy skill to detect the status of a call center - such as a disaster event or a closure. Call centers can now use value variables instead of the dummy agent and skill. Call centers have more options because values can be set from 0 to 9. Value variables can also be set remotely using a FAC that allows for quicker exits in disaster scenarios.

Additional information

You should also understand the following items about the value variable:

- Association of a value variable with a FAC allows you to use the phone to access a FAC and change the assigned variable value quickly and easily. If you do not create a 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 screen.
- If you set up a FAC to change a value variable assignment, a station user must use a physical phone that has the required console permissions.
- 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 you would use 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 you use to represent an operational mode or condition to which you want to be able to respond as needed in your call applications. For more information, see Example application using a value variable on page 143.

VIV interactions and considerations

Avaya CMS interactions: 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 Avaya Communication Manager by use of the Variables for Vectors administration table.

Also, if the CMS release is earlier than Release 12, an attempt to administer a vector that includes one or more vector variables generates an error message.

Note:

The specific commands for which variables are supported, depends on the CMS and Communication Manager release. For more information, see Command syntax for vector variables on page 111.

Variable failure conditions: When the variable conditional that is tested is not defined in the variable administration table, a **goto** test fails, a 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, all the information is cleared. It is important to 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, vdntime, and so on), and the value stored in “digits” can be used in a subsequent routed-to vector or vector steps.

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. Thus the value determined by or assigned through the use of 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 directly to the ASAI UUI string using the "asaiuui" vector variable type.

VIV administration

This section lists the administration screens and settings that are required to administer the VIV feature.

Note:
For most of the variable types, administration is done solely in the variables administration table. However, a FAC administration step is also required if you want to use a FAC to change assignments for value variables.

This section includes the following topics:

- Example Variables for Vectors screen on page 136
- Required variable administration entries on page 137
- Optional FAC administration for value variables on page 138

Example Variables for Vectors screen

You use the following screen to administer vector variables. For a description of the entries required for each variable type, see Required variable administration entries on page 137.

```
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 administration screen for the different types of variables.

Variable type	Scope	Length	Start	Assignment	VAC (Variable Access Code)
ani	Local only (L)	1 to 16 digits (required)	start position from 1 to 16 (required)	Not applicable	Not applicable
asaiuui	Local only (L)	1 to 16 digits (required)	start position from 1 to 96 (required)	Not applicable	Not applicable
collect	Local or Global (L or G, required)	1 to 16 digits (required)	start position from 1 to 16 (required)	Local - not applicable Global - 1 to 16 digits	Not applicable
dow	Global only (G)	Not applicable	Not applicable	Not applicable	Not applicable
doy	Global only (G)	Not applicable	Not applicable	Not applicable	Not applicable
stepcnt	Local only (L)	Not applicable	Not applicable	Not applicable	Not applicable
tod	Global only (G)	Not applicable	Not applicable	Not applicable	Not applicable
value	Global only (G)	1	Not applicable	1 digit (0 to 9, optional)[1]	VV*x* (optional)[2]
vdn	Local only (L)	Not applicable	Not applicable	active or latest	Not applicable
vdntime	Local only (L)	Not applicable	Not applicable	Not applicable	Not applicable

1. If you do not assign a value in this field, a null value is specified. However, if you administer a FAC to set the variable assignment, any value that you assign by dial code procedure is subsequently displayed in this field. For more information, see Optional FAC administration for value variables on page 138.

2. You must enter a VAC value if you want to be able to use a FAC to change the variable assignment. The format for the VAC value is VV*x*, 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 administration screen after you set up the FAC. In the FAC screen, the VV*x* value is displayed on the same line as the FAC code, as described in Optional FAC administration for value variables on page 138. If you do not specify a VV*x* value when you administer the variable, you receive an intercept tone when you attempt to dial the FAC.

Optional FAC administration for value variables

This section describes the administration steps that you need to do if you use value variables in your vectors and want to be able to use a FAC to change the variable assignments.

Use the following screen to administer a FAC that you can use to change value variable assignments.

```
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 a FAC that you can use to change variable values:

1. On the Call Vector/Call Prompting Features page of the Feature Access Codes screen, enter a 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 # or * character can be substituted for a numeral at the first digit position.
2. Note the VV*x* 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 that you want to associate with the FAC. For more information, see Required variable administration entries on page 137.

VIV vector examples

This section provides more simple examples that show how vector variables can be used to help improve call processing operations. Other examples are provided in System-assigned vector variable types on page 121.

This section includes the following topics:

- Example application using time and day variables on page 139
- Example application using a value variable on page 143
- Example applications using Global Collect variables on page 144
- Example applications using vdn type variables on page 146
- Example application using a variable in other commands on page 147
- Example application using a variable in the converse-on command on page 148

Example application using time and day variables

The VIV feature provides time and day variables that you can use to enhance vector functionality and efficiency in many different ways. The following example shows how:

- You can use time of day (tod) and day of week (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 that give you the ability to reassign variable values for opening and closing time whenever necessary, such as when a change in daylight savings time occurs. The new variable values are instantly propagated to any number of vectors in which they are used.

Details for the example scenario and the steps required to implement the solution are provided in the following sections:

- Scenario details on page 140
- Administering the variables on page 140
- Creating a vector to use the time and day variables on page 141
- Creating a vector to reassign call center hours of operation on page 142

Scenario details

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

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

Administering the variables

The specifications that you would provide in the for Variables for Vectors screen for the variables used in this example are shown in the following table:

Variable	Description	Type	Scope	Length	Start	Assignment
Time of day or day of week 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						
O	opening time, all days of week	collect	G	4	1	0700
L[1]	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

1. In the current example, the 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 **L**atest possible closing time.

Creating a vector to use the time and day variables

The following vector example shows how the tod and dow variables can tested against call center business hours so that call processing is controlled in an appropriate manner.

```
 1. goto step 30 if T < O     [if tod is earlier than 0700 hours, go to 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 out of hours treatment]
 4. goto step 30 if D = 7     [if dow is Saturday, go to 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 preceding
                               step), and dow is Friday, go to out of hours treatment]
 7. goto step 30 if T > L     [if tod is later than 2300, go to out of hours treatment]
 8. goto step 31 if holiday in table 8  [based on 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 preceding vector example, the tod, dow and Global Collect variables control the flow of the call process by testing 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 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, 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.

Creating a vector to reassign call center hours of operation

As described in Creating a vector to use the time and day variables on page 141, tod and dow variables can be tested against collect variables that specify call center opening and closing times for different days of the week. Because Global Collect variables are used to specify these 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 would need to create other vectors like this one for each of the Global Collect variables that you use to set call center opening and closing times.

```
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 of the potential uses for value variables is to allow multiple call applications to be quickly switched from one operational mode to another. Such a rapid switchover capability can be useful for businesses whose operations may be impacted by unpredictable events. For example, a public utility might desire a switchover 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 switchover need, you could administer both a value variable and an associated FAC, as described in the following sections:

- Administering a FAC code to use with a value variable on page 143
- Administering the value variable on page 144
- Using the value variable in multiple vectors on page 144

Administering a FAC code 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 the FAC.

Note:
When you administer the FAC for the variable, note the VV*x* 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 on page 144.

```
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: ____
```

Administering the value variable

After you set up a FAC to use with the value variable, you need to administer the Variables for Vectors screen to set up the value variable associated with the FAC.

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

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

Using 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 switchover conditions.

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

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

In this example, the default value for the switchover variable is administered with a value assignment of *1*, to denote normal operational modes. When a switchover due to blizzard conditions is required, the call center administrator dials *23 to access the FAC and enters the digit *2* to indicate that switchover 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 a Feature Access Code 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 a Feature Access Code.

Global Collect variables offer many advantages over the Value variable, including:

- Global Collect variables are not limited to nine Feature Access Codes.
- The ability to add a Security Code to prevent malicious or accidental changes to the call flows.
- Creation of a VDN/vector menu that could prompt for the variable to change (multiple variables via one VDN).

- Call flow routing can be changed remotely by calling the VDN rather than via Remote Access to dial an FAC.
- Additional features such as feedback announcements and confirmation.

The general functionality of the vectors used in the following examples is available with CM2.0 or later. The vector comment steps require CM4.0 or later. The VDN variables (for example, V1 –V5) are available with CM3.0 or later.

Verifying a password and then changing a value

For this example, assume that Global Collect variable “A” is used in Call Center vectors to control the flow that a call might experience. The variable is assigned on the **change variable** form and is given a type of **Collect** and a scope of **G** for Global. The other settings depend on how the variable will be used. Typically the length will be 1 to 16 with a start position of **1**.

The vector prompts the user for a 16 digit password. This 16 digit password is compared to the expected value that is contained on the Active VDN as VDN variable “V2”. Step 8 prompts the user to enter new value for Variable A. The announcement could be recorded to list the expected values along with their use.

This example illustrates one approach to accomplishing these tasks. The vector flow could 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
```

Adding 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

The vdn variable type can be used to reduce the number of vectors required to provide differential treatment to DNIS VDNs. The following examples show different ways to use vdn 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 would provide in the Variables for Vectors screen for the vdn variables 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

The first example shows how the administered vdn variable can be used in a single vector to provide multiple announcement treatments based on call identity. Vector processing 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 variable is compared to one of several possible administered VDN values (**Y <> 1001**). If the vdn variable value 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 next example shows another way that the vdn variable can be applied in a vector to implement selective call treatment. In this example, the vdn 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 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 variable in other commands

A variable can be used in the route-to number command to route a call to a destination provided indirectly though 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.

The following shows the specifications that you would provide in the Variables for Vectors screen for variable R.

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 should 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 variable in the converse-on command

Including a variable in the converse-on command as a data item to pass (out-pulse) to the VRU or IVR allows the 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 collected digits, 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. For this example, variable D forwards numerical account code data of up to 6-digits provided by an ASAI application.

The following shows the specifications that you would provide in the Variables for Vectors screen for variable D.

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. See Call flow and specifications for converse - VRI calls on page 807 for details on data passing and data return functions.

Troubleshooting vector variables

This section includes information which may assist you to troubleshoot Variables in Vectors implementations.

Useful commands: You can use the following commands to help analyze vector variable operations:

- **`list trace vector/vdn xx`** - When you use a list trace command to analyze vector operations, the current values assigned to the variables used in vector steps are displayed in the report.
- **`list usage variables [x]`** – This command provides a list of all vectors that use variables and specifies which administered variable is used in each vector. You can optionally filter the list if you include a specific (A-Z, AA-ZZ) administered variable.

Variable-related vector events : The following vector events are associated with vector operations:

- Event type 37: collect digits for variable error
- Event type 38: variable not defined
- Event type 213: No digits in variable

For more information, see Vector events on page 709.

VDN variables

This section includes the following topics:

- Description of VDN variables on page 151
- VDN variable fields on page 152
- Where to use VDN variables on page 152
- Case studies on page 158

Description of VDN variables

VDN variables provide more opportunities for VDNs to use a smaller set of vectors.

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 vector 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

Reason to use

You can create general-purpose vectors that support multiple applications with call-wait treatments that are tailored to the application.

Call centers have many vectors that use the same basic call flow but are unique because each require unique announcements, route-to destinations, holiday tables, vector routing table indexes, and conditional limits. The VDN variables allow you to create a generic call flow vector. The unique items are now designated on the VDN screen using VDN variables. VDN variables can drastically reduce the number of vectors needed, ensure common flows, and ease administration during crisis times when the flows need to change due to an unforeseen event. Unforeseen events can include problems with trunking, staffing, or messaging.

VDN variable fields

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

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

The description field allows users to describe the VDN variable using up to 15 characters.

The assignment field assigns an up to 16-digit unvalidated integer number to the VDN variable. Each digit entry can be:

- 0 - 9
- Left blank

Where to use VDN variables

You can use the VDN variables in all vector commands that support vector variables except as a **for** parameter with the **collect-digits** command. This consists of the following commands:

- announcement commands with VDN variables on page 152
- converse-on command with VDN variables on page 153
- disconnect command with VDN variables on page 154
- goto commands with VDN variables on page 154
- route-to command with VDN variables on page 156
- set command with VDN variables on page 157
- wait command with VDN variables on page 158

announcement commands with VDN variables

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 **announcement** commands.

```
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 for using VDN variables in announcements

The requirements for using VDN variables after the announcement extension are:

- 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 call's active VDN and used as the announcement extension number.
- The number must be a valid announcement extension assigned on the Audio/ Announcement screen.

See also:

- announcement command on page 538
- announcement commands with vector variables on page 112

converse-on command with VDN variables

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

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

1. A valid hunt group is a vector-controlled ACD split or skill assigned on a Hunt Group screen.
2. You can use a VDN variable only in data1, only in data2, or in both.

See also:

- converse-on command on page 566
- converse-on command with vector variables on page 113

disconnect command with VDN variables

You can use VDN variables with the disconnect command after an announcement extension. For more information about using VDN variables after an announcement extension, see announcement commands with VDN variables on page 152.

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

disconnect after announcement [*V1-V9*]

See also:

- disconnect command on page 579
- disconnect command with vector variables on page 114

goto commands with VDN variables

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

goto step *1-99* **if** **or** **goto vector** *1-2000* **@step** *1-99* **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** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**>,>=,<>,=, <,<=**	V1-V9
counted-calls	**to vdn**	**vdn extension**, **latest**, **active**	**>,>=,<>,=, <,<=**		*V1-V9*	

goto step *1-99* **if** **or** **goto vector** *1-2000* **@step** *1-99* **if**						
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 = l**ow **m = m**edium **h = h**igh **t = t**op	**>,>=,<>,=,<, <=**	*V1-V9*
	for skill	*hunt group*, skills for VDN: **1st**, **2nd**, **3rd**				
holiday	**in table**	*V1-V9*				
	not-in table					
ii-digits	**>,>=,<>,=,<,<=**	*V1-V9*				
	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 = l**ow **m = m**edium **h = h**igh **t = t**op	**>,>=,<>,=<,<=**	*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*			

goto step *1-99* **if** **or** **goto vector** *1-2000* **@step** *1-99* **if**						
wait-improved for	**best**	**>,>=,<>,=,<,<=**				*V1-V9*
	skill	*hunt group*, skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**>,>=,<>,=<,<=**	
	split	*hunt group*				

See also:

- goto step and goto vector commands on page 583
- goto commands with vector variables on page 115

route-to command with VDN variables

Variable syntax for the **route-to** command is supported for Communication Manager 2.1 or later.

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

route-to	**number**	*V1-V9*, **~r[***V1-V9***]**[1]	**with cov**	**y**, **n** y = yes n = no	**if**	**digit**	**>**, **>=**, **<>**, **=<**, **<=**	*0-9*, **#**
						interflow-qpos	**<**, **=**, **<=**	*1-9*
						unconditionally		

1. When the specified number is preceeded by ~r, Network Call Redirection is attempted.

Requirements and considerations

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

- A variable can be used 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, or assignment, of up to 16 digits is used for the destination.
- If the variable is not defined, 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 obtained from the string of digits of the variable's current value 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.

See also:

- route-to command on page 619
- route-to command with vector variables on page 117

set command with VDN variables

The following fields allow VDN variables with the **set** command.

set [*variables*[1], **digits**] **=** [*operand1*] [*operator*] [*operand2*]

1. A user-assignable vector variable only.

You can use VDN variables in the following fields:

- Operand1
- Operand2

See also:

- set command on page 632.
- set command with vector variables on page 118.

wait command with VDN variables

You can use VDN variables with the **wait** command as an announcement extension. For more information about using variables after an announcement extension, see announcement commands with VDN variables on page 152.

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**]

See also:

- wait-time command on page 645
- wait command with vector variables on page 118

Case studies

This section includes the following topics:

- Using one vector for different announcements on page 158
- Combining values in VDN variables to expand capacity on page 159

Using one vector for different announcements

In this case study, agents working for the Alpha service bureau handle calls for three different companies - ABC Company, XYZ Company, and JYK Company. 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 to be played
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. …

Combining values in VDN variables to expand capacity

This section includes the following topics:

- Business case on page 160
- Current configuration on page 160
- Assigning parameters on page 161
- Grouping parameters on page 161
- Assigning digit strings on page 162
- Separating the parameters and assigning them to vector variables on page 163
- Defining the vector variables on page 163
- Separating each VDN variable on page 164
- Summary on page 164

Business case

In this case study, the XYZ company has a separate vector for every application handled in their call center. Using VDN variables, they can consolidate similar vectors that are each reached by a different VDN, into one vector. They plan to use the newly-freed vectors for other applications. They have a problem in that the number of different parameters or values they need assigned to the VDNs as VDN variables exceeds the limit of five variables.

This 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 their component parts, and assigned to vector variables in the common vector for each of the vector commands when needed.

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
```

Assigning parameters

These are the parameters that need to be assigned for three VDNs. The parameters appear in the vector in the same order as described in this 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 (1st)	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 (2nd)	210	310	410
messaging mailbox extension	5001	5002	5003

Grouping parameters

One way to combine the parameters is to group them 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

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

Assigning 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.

Separating the parameters and assigning them 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 them 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
```

Defining the 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

Separating each VDN variable

Vector 2 is the subroutine vector. Vector 2 separates each of the VDN variables 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

This case study described how to use the V1 through V3 VDN variables to support eight parameters. It also described how to use V4 for another parameter that also needed to be passed with the active VDN for the call. This approach supported nine parameters with four VDN variables while keeping V5 as a spare. This approach can be expanded to handle even more parameters when needed. Since the A through H vector variables are local variables, they can be reused in other vectors applications that have similar string lengths.

Vector subroutines

This section includes the following topics:

- Overview of vector subroutines
- The goto vector command and subroutines
- The @step parameter
- Example 1: Test for working hours

Overview of vector subroutines

You can create vector subroutines beginning with the 3.0 release. 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 S8500 and S8700 platforms
- 400 for S8300 platforms

Reason to use

Vector subroutines allow you to 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 would have to repeat the step. Subroutines also:

- Free up more steps per vector by removing duplication
- Allow unused steps at the end of vectors to be used for subroutines, thus expanding vector capacity
- Reduces administration - you can make changes to only one vector subroutine that is referenced by many vectors, such as changing office hours or wait treatment

The goto vector command and subroutines

The **goto vector** command allows branching 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 that 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. Any changes or additions stay with the call when the call returns to the calling vector.

The @step parameter

Use the @step parameter with all supported **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:

- 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 XYZ Retail Stores call center has a large number of vectors that check whether 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.

This call center now needs to make a change in only one place instead of in every vector, saving them changes to possibly hundreds of vectors.

Incoming call processing vector

The following example provides a typical incoming call processing vector.

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 subroutine vector

The following example shows a subroutine vector that checks working hours.

```
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.*]

ANI /II-digits routing and Caller Information Forwarding (CINFO)

The ANI (Automatic Number Identification) and II-digits (Information Indicator Digits) Call Vectoring features help you to make vector routing decisions based on caller identity and the of originating line. Caller Information Forwarding (CINFO) makes it possible for you to collect caller entered digits (ced) and customer database provided digits (cdpd) for a call from the network.

When ANI and II-digits are provided with an incoming call to a VDN, they are sent to Avaya Call Management System (CMS) when vector processing starts. ANI, II, and CINFO digits are forwarded with interflowed calls. ANI and II-digits are also passed over the Adjunct Switch Application Interface (ASAI) in event reports.

This section includes the following topics:

- Command sets on page 169
- ANI routing on page 170
- II-digits routing on page 174
- Caller Information Forwarding on page 180

Command sets

The following table lists the commands that are used by ANI, II-digits, and CINFO digits.

Command category	Action taken	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`

Command category	Action taken	Command
Information Collection		
	Pass ANI to a Voice Response Unit. Pass ced and cdpd to a Voice Response Unit (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 adjunct that is based on ced or cdpd.	`adjunct-routing`

ANI routing

ANI provides information about the caller 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, and is supported with ISDN or SIP trunks.

This section includes the following topics:

- ANI basics on page 170
- ANI routing example on page 172
- Using ANI with vector routing tables on page 172

ANI basics

Calling Party Number (CPN) and Billing Number : ANI is based on the Calling Party Number (CPN). It is not always identical to the Billing Number. For example, if the call is placed by a user from a switch, the CPN can be either the switch-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 North America contains only 10 digits.

Call types that use ANI: The following call types have ANI values associated with them:

- Incoming ISDN-PRI calls that send ANI
- Incoming SIP calls that send SIP contact headers
- Incoming R2-MFC signaling calls that send ANI
- 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 that call.

Use of wildcards: The ANI value that is specified for a goto step can include the + and/or the ? wildcards. The + represents a group of zero or more digits and can be used only as the first or last character of the string. The ? represents a single digit. Any number of the wildcard can be used at any position in the digit string.

Use with vector routing tables: ANI data can be tested against ANI numbers provided in vector routing tables. For more information, see Using ANI with vector routing tables on page 172.

EAS agent calls: When an EAS agent makes a call to a VDN, the agent's login ID is used as the ANI instead of the number of the physical terminal.

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

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



Tip:
To ensure that the originator's ANI 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.

ANI routing example

The following vector example shows several applications of ANI Routing.

```
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 ANI associated with them are routed to an operator. Step 3 routes calls from a specific telephone to a specified extension. Step 4 routes local calls, which are calls with 7 or fewer digits, to a different vector. Steps 5 and 6 route calls from area codes 212 and 202 to a different vector. Calls that are not rerouted by the previous steps are then queued.

Using ANI with vector routing tables

You can test ANI against entries in a Vector Routing Table. Vector Routing Tables 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 example Vector Routing Table shown below 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 Vector Routing Table shown above.

```
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 shown above, if no ANI is available for the call, it is routed to an operator. If the first three numbers match an area code from table 6, the call is routed to vector 8. All other calls are queued.

II-digits routing

II-digits provide information about the originating line for a call. This information can be used for a variety of purposes, such as:

- Help detect fraudulent orders for catalog sales, travel reservations, money transfers, traveler's checks, and so forth
- Assign priority or special treatment to calls that are placed from pay telephones, cellular telephones, motel telephones, and so forth. For example, special priority could be given by an automobile emergency road service to calls that are placed from pay telephones
- Detect calls placed from pay telephones when it is the intention of the caller to avoid 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

This section includes the following topics:

- II-digits basics on page 174
- II-digits codes on page 175
- II-digits routing example on page 180

II-digits basics

String description: II-digits is a 2-digit string that is provided for an incoming call by ISDN PRI. 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 about 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 will not match the digit string 2 in a vector step.

Use with a vector routing table: 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 II-digits values associated with them:

- 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 (DCS over PRI) does.

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 is completed before the call reaches the II-digits conditional, the II-digits value of the originator of the call is used.
- If the transfer is completed after the call reaches the II-digits conditional, the II-digits value of the terminal that is 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's 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

The following table lists the current assignments for II-digits.

Note:
II-digit assignments are maintained by the North American Numbering Plan Administration (NANPA). To obtain the most current II digit assignments and descriptions, go to:

http://www.nanpa.com/number_resource_info/ani_ii_assignments.html

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 will cause 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 Operator Services System, the calling number may be verbally obtained and manually recorded. If manual operator identification is not available, the receiving switching system (e.g., an interLATA carrier without operator capabilities) may reject the call.
03-05	Unassigned
06	Station Level Rating - The 06 digit pair is used when the customer has subscribed to a class of service in order to be provided with real time billing information. For example, hotel/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, e.g., 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 and/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/account identification.
07	Special Operator Handling Required - calls generated from stations that require further operator or Operator Services System screening are accompanied by the 07 code. The code is used to route the call to an operator or Operator Services System for further screening and to determine if the station has a denied-originating class of service or special routing/billing procedures. If the call is unauthorized, the calling party will be 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.
21-22	Unassigned

II-digits assignments (continued)

II-digits	Description
23	Coin or Non-Coin - on calls using database access, e.g., 800, ANI II 23 is used to indicate that the coin/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 would otherwise indicate that the non-coin status is known, i.e., 00, or when there is ANI failure. ANI II 23 may be substituted for a valid 2-digit ANI pair on 0-800 calls. In all other cases, ANI II 23 should *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 may 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 Plain Old Telephone Service (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. Thus, Code 24 indicates that this is 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 Plain Old Telephone Service (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 will be replaced with Code 25 under the above 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/Inmate Service - the ANI II digit pair 29 is used to designate lines within a confinement/detention facility that are intended for inmate/detainee use and require outward call screening and restriction (e.g., 0+ collect only service). A confinement/detention facility may be defined as including, but not limited to, Federal, State and/or Local prisons, juvenile facilities, immigration and naturalization confinement/detention facilities, etc., which are under the administration of Federal, State, City, County, or other Governmental agencies. Prison/Inmate Service lines will be identified by the customer requesting such call screening and restriction. In those cases where private pay stations are located in confinement/detention facilities, and the same call restrictions applicable to Prison/Inmate Service required, the ANI II digit for Prison/Inmate Service will apply if the line is identified for Prison/Inmate Service by the customer.

II-digits assignments (continued)

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 should be used: • 30 - Intercept (blank) - for calls to unassigned directory number (DN) • 31 - Intercept (trouble) - for calls to directory numbers (DN) 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 may change the standard ANI digits to 34, before outpulsing the sequence to the IC, when the Telco performs all call handling functions, e.g., 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 class of service and the service area of the call party. When these calls are routed to the interexchange carrier using a combined WATS-POTS trunk group, it is necessary to 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 (i.e., a line for which there are no billing restrictions). Accordingly, if no request for alternate billing is made, the call will be billed to the calling line.
61	Cellular/Wireless PCS (Type 1) - The 61 digit pair is to be forwarded to the interexchange carrier by the local exchange carrier for traffic originating from a cellular/wireless PCS carrier over type 1 trunks. (Note: ANI information accompanying digit pair 61 identifies only the originating cellular/wireless PCS system, not the mobile directory placing the call.

II-digits assignments (continued)

II-digits	Description
62	Cellular/Wireless PCS (Type 2) - The 62 digit pair is to be forwarded to the interexchange carrier by the cellular/wireless PCS carrier when routing traffic over type 2 trunks through the local exchange carrier access tandem for delivery to the interexchange carrier. (Note: ANI information accompanying digit pair 62 identifies the mobile directory number placing the call but does not necessarily identify the true call point of origin.)
63	Cellular/Wireless PCS (Roaming) - The 63 digit pair is to be forwarded to the interexchange carrier by the cellular/wireless PCS subscriber roaming in another cellular/wireless PCS network, over type 2 trunks through the local exchange carrier access tandem for delivery to the interexchange carrier. (Note: Use of 63 signifies that the called number is used only for network routing and should not be disclosed to the cellular/wireless PCS subscriber. Also, ANI information accompanying digit pair 63 identifies the mobile directory number 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/ motel. The transport carrier can use this indication, along with other information (e.g., whether the call was dialed 1+ or 0+) to determine the appropriate billing arrangement (i.e., bill to room or alternate bill).
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 should not be allowed and additional screening, if available, should 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 coinless 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 these lines.
71-79	Unassigned
80-89	Reserved for Future Expansion to 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 shows branching calls that use II-digits to route to different VDNs.

Note:
In this example, VDN override is set to **yes** on the called VDN. In this way, the VDN name or VDN of Origin Announcement 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 shown above, 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.

Caller Information Forwarding

The Caller Information Forwarding (CINFO) feature allows you to associate Caller entered digits (ced) and customer database provided digits (cdpd) 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.

This section includes the following topics:

- CINFO basics on page 181
- CINFO vector example on page 183
- CINFO interactions on page 183

CINFO basics

This section includes the following topics:

- UEC IE storage on page 181
- Use with collect digits commands on page 181
- Use of wildcards on page 181
- String length on page 182
- Vector commands that use ced and cdpd on page 182
- Internal transfer to a VDN on page 182
- Buffer storage considerations on page 182

UEC IE storage

When an ISDN call is received from either the AT&T network or a tandemed PRI call, the communication server stores the Codeset 6 User Entered Code Information Element (UEC IE) when it contains the ced and/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, the system retrieves the ced or cdpd and places them in the collected digits buffer. Any digits that were 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, the communication server 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 telephone to display ced and cdpd information just like 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 is completed before the call reaches the CINFO conditional, the CINFO value of the originator of the call is used.
- If the transfer is completed 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 server requests information from the caller (ced) and from the call center database (cdpd). These digits are conveyed in the call ISDN message to the communication server 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 in case calls will be 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 that was received. 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

This section describes CINFO interactions with other features and applications.

ASAI: ced and cdpd can be passed to an ASAI adjunct as collected digits with the **adjunct routing link** command and other event reports. ASAI will pass a maximum of 16 digits.

If a touch-tone 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 that are 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. When CINFO digits are provided to an ASAI adjunct they are provided in the same manner as any other collected digits from a vector.

The Call Offered to (VDN) Domain Event Report will contain 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 multi-site BSR application routes the call to another communication server.

Avaya CMS: The Vectoring (CINFO) customer option is not required for ced or cdpd to be passed to CMS. Any version of the CMS will accept 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: CINFO digits are included with the call if Look-Ahead Interflow routes the call to another communication server.

Transfer: If a call is transferred from the communication server, 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 should not be collected until the transferring party has had time to complete the transfer. Therefore, when transfers are likely, an appropriate wait-time step should be included before the collect step.

Multi-national Calling Party Number (CPN) prefixes

This section includes the following topic:

- About multi-national Calling Party Number (CPN) prefixes on page 185
- Administering multi-national CPN prefix options on page 186
- CPN Prefix routing example on page 186
- Before enabling the CPN prefix option on page 187

About multi-national Calling Party Number (CPN) prefixes

The Calling Party Number (CPN) that accompanies a call can be used for a variety of purposes, including telephone display, routing, billing, and screen pop 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 these two locations will each begin with the same digits. The prefix in a multinational CPN helps differentiate between local/national and 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 these digits are typically 0 and 00. When these values are administered in CM, they are included as part of the CPN and can be displayed on an agent's telephone 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. This option can be set on a system-wide basis, and overridden, if desired, 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 pertinent language. For example, if a customer from Germany calls in 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 might be given priority over a national call to save toll charges.

Administering multi-national CPN prefix options

The following screens and fields are required to administer the multi-national CPN prefix options:

Screen	Field
Feature-Related System Parameters	National CPN Prefix International CPN Prefix Pass Prefixed CPN: VDN/Vector?
Vector Directory Number	Pass Prefixed CPN to VDN/Vector?

CPN Prefix routing example

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 ANI associated with them 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, whose country codes are listed in 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.

Before enabling the CPN prefix option

Before switching on the option to pass the CPN prefix to VDNs and vectors, examine the following administration forms:

- Public Unknown Numbering Plan
- Route Pattern
- Tandem Calling Party Number Conversion
- Trunk Group
- Uniform Dial Plan
- Vector and
- Vector Routing Table

There may be assumptions in these forms related to the presence of national (NATL) and international (INTL) codes in a CPN. You may need to make modifications to the above resources to take advantage of the feature and to avoid errors in call handling.

Information Forwarding

The Information Forwarding feature sends information with calls over public and private networks using ISDN trunks or SIP trunks. Private networks that are enabled for Information Forwarding can also be configured for QSIG or non-QSIG protocols. QSIG protocols are not supported with SIP trunks. Call data derived from the Information Forwarding feature can be used to enhance call processing, customer service and data collection.

Note:
Asynchronous Transfer Mode (ATM) trunking and IP trunking can be set up to emulate ISDN PRI. For more information, see *Administration for Network Connectivity for Avaya Communication Manager*, and *ATM Installation, Upgrades and Administration using Avaya Communication Manager*.

This section includes the following topics:

- Data handled by Information Forwarding on page 189
- Information Forwarding benefits on page 190
- ISDN network requirements on page 191
- SIP network requirements on page 192
- Information Forwarding support for BSR and LAI on page 192
- ASAI shared UUI data conversion on page 195
- Determining user information needs on page 196
- Information Forwarding troubleshooting on page 199

Data handled by Information Forwarding

The following table shows the call-related information you can send through Information Forwarding depending on your trunk type:

Incoming call-related information	Supported trunks
ANI	ISDN SIP
II-Digits	ISDN

Incoming call-related information	Supported trunks
CINFO	ISDN
Adjunct Switch Application Interface (ASAI)-provided user information	ISDN SIP[1]
Look-Ahead Interflow (LAI) information, such as the in-queue timestamp, VDN name, and network-provided caller information, including priority level and type of interflow.	ISDN
Universal Call ID (UCID) - UCID provides a unique identifier for each call that is used to track the call. For more information, see Universal Call ID in *Feature Description and Implementation for Avaya Communication Manager*.	ISDN SIP
Interflowed Collected Digits and in-VDN time data.	ISDN SIP

1. Adjunct Switch Application Interface (ASAI)-provided user information is the only information provided if the SIP trunk group is assigned as "service provider".

For information about administering information transport, see *Feature Description and Implementation for Avaya Communication Manager*. For detailed information about trunk group setting interactions with Information Forwarding, UCID, and multi-site routing, see Appendix E: Advanced multi-site routing on page 729.

Information Forwarding benefits

The following table lists Information Forwarding benefits:

Function	Benefit
Improved agent efficiency and service to call	Forwarding of original caller service requirements and entered prompted digits speeds service to the caller and saves the agent time.
Improved network-wide call tracking	Forwarding of UCID, In-VDN-Time and collected digits allows tracking as a single call and provides a network-wide view for call statistics.
Improved CTI integration	Forwarding of UCID, In-VDN-Time, and collected digits provides screen pop and database access applications across sites.

Function	Benefit
Forwarding of original call service requirements (VDN Name or DNIS)	Faster and more efficient agent handling, better service to the caller, and improved CTI integration
Transport of UCID	Improved call tracking as a single call and CTI integration
Collected Digits Transport	Better service to the caller because the caller doesn't have to repeat input of information, more information for the agent, better and faster call handling, improved call tracking because the collected digits are included with the call record, and improved CTI integration
Forwarding of In-VDN Time	Improved call tracking as a single call and end-to-end time-before-answer statistics
Support of ASAI user Information Forwarding	CTI integration

ISDN network requirements

Your network must meet the following requirements to support Information Forwarding:

- Both the private and public networks must support end-to-end transport of codeset 0 user data either as user-to-user information (UUI IE) or QSIG manufacturer specific information (MSI) in the SETUP and DISCONNECT ISDN messages. Private networks can be configured for either non-QSIG transport by way of a codeset 0 UUI IE or QSIG transport by way of MSI packaged in a codeset 0 Facility IE. Public networks do not currently support QSIG, and user data can only be transported by way of the UUI IE when supported by the network. Future public network offerings may support QSIG by way of a Virtual Private Network.
- The communication server must support the ISDN country protocol.

  Important:
  If testing has not been done to verify operation over the public networks that are involved with the preferred specific configuration, use of private ISDN trunking between the nodes should be assumed until successful testing is complete.

- The network byte limit for the user data portion of user information contents must be large enough to carry the data that is needed for the customer application.

Note:
Some public network providers may require service activation and/or fees for user information transport.

SIP network requirements

Your network must meet the following requirements to support Information Forwarding:

- The public network must support end-to-end transport of SIP user data in a user-to-user information (UUI) header in the INVITE messages.
- The communication server must support the SIP protocol and the UUI header.
- The network byte limit for the user data portion of user information contents must be large enough to carry the data that is needed for the customer application.

Information Forwarding support for BSR and LAI

When a call is interflowed to another communication server by BSR or Look-Ahead Interflow, the following data types are supported for Information Forwarding:

- Collected Digits - Any digits that are collected for the call are passed with the interflowed call, and automatically collected when the call enters vector processing at the receiving communication server.
- Elapsed in-VDN time - The elapsed time that the call has already spent at the sending communication server is passed with the interflowed call and automatically sent to the Avaya Call Management System (CMS) when the call enters vector processing at the receiving communication server.
- UCID - Universal Call ID.

The following sections describe handling and transport of Information Forwarding data in interflowed calls:

- Forwarding collected digits with interflowed call on page 192
- Forwarding accumulated in-VDN time on page 193
- Transport by way of globally-supported methods over ISDN on page 193
- Transport by way of globally-supported methods over SIP on page 194
- LAI backward compatibility issues on page 195

Forwarding collected digits with interflowed call

The following list describes how forwarded collected digits are handled in interflowed calls:

- The last set of up to 16 collected digits, not including the dial-ahead digits, are forwarded with a call interflowed over ISDN or SIP facilities.
- When processing for the call at the remote location reaches the VDN, the forwarded digits are inserted in the collected digits buffer. Therefore, a TTR is not needed. The objective is to immediately provide the collected digits to the CMS in a DIGITS message and to ASAI by way of the VDN event report in the same manner as incoming ANI.
- The collected digits are available for further routing by steps in the assigned and subsequent vectors, and eventual display to the answering agent.
- All interactions with the collected digits are the same as digits that are collected using a collect step. For example, a subsequent collect step will clear the digits.
- If the call is further interflowed or tandemed over ISDN or SIP facilities, the collected digits are tandemed with the call. If more digits are collected at the tandem communication server, the latest collected digits are tandemed.

Forwarding accumulated in-VDN time

The following list describes how forwarded in-VDN time data is handles in interflowed calls:

- When a call is interflowed, the in-VDN time in seconds, from 0 to 9999, is included. The in-VDN time is the elapsed time starting from the VDN that was originally called until when the Information Forwarding message is created.
- If the call was interflowed to the local system and in-VDN time was received for the call, the previous in-VDN time is added to the local in-VDN time.
- If the accumulated time exceeds the largest value that can be transported, the maximum value is sent.
- The accumulated in-VDN time that is received on an incoming interflowed call is forwarded to the CMS in the DNEVENT message when the call starts VDN/vector processing at the remote location.
- In-VDN time does not pass to the Basic Call Management System (BCMS) for reporting by BCMS.

Transport by way of globally-supported methods over ISDN

Use of codeset 0 supports information transport over ISDN PRI/BRI facilities (QSIG or non-QSIG) as well as supporting operation over public networks. The following list describes information transport by way of globally-supported methods over ISDN trunks:

- When a call is LAI or BSR interflowed, the following information is forwarded with the call over public or private ISDN networks using QSIG or non-QSIG protocols:
  - LAI information.

Note:
The forwarded LAI information is the same as that sent in the LAI IE: VDN name (also called LAI DNIS), put in queue time-stamp, priority level and type of interflow.

- Collected digits.
- in-VDN time data in the ISDN SETUP message.

- Other call related information, including calling party number (ANI), calling party name, II-digits and CINFO digits, that is tandemed with the interflowed call in the SETUP message is forwarded in the normal manner.

Note:
II-digits and CINFO are forwarded as codeset 6 IEs which may be a problem in some networks.

- At the remote end, the transported data is separated into its component parts for storage with the call, call vectoring, call processing and display, further interflow or tandeming, and forwarding to adjuncts. For example, the LAI info is treated as though it was received as an incoming codeset 6 LAI IE including forwarding over ASAI as a code set 6 LAI IE in event reports.
- When a status poll call is placed to the remote location, the communication server only forwards the UCID and caller information that was received from the original call.
- In response to a status poll, the communication server forwards the reply-best status data in the ISDN DISCONNECT message over public or private ISDN PRI/BRI networks. In this case, the DISCONNECT message has a cause value of 31 *Normal-Unspecified* for wider international interoperability.
- The Multi-Site Routing related data is in addition to the associated ASAI user data, which was previously sent in a non shared UUI IE, and the UCID data.

Transport by way of globally-supported methods over SIP

SIP supports information transport with the UUI header. The service provider must support SIP information transport. The following list describes information transport by way of globally-supported methods over SIP trunks:

- When a call is LAI or BSR interflowed, UUI header information is forwarded with the call over SIP networks using non-QSIG protocols.
- Other call related information, including calling party number (ANI), and calling party name is forwarded in the normal manner.
- At the remote end, the transported data is separated into its component parts for storage with the call, call vectoring, call processing and display, further interflow or tandeming, and forwarding to adjuncts.

- The Multi-Site Routing related data is in addition to the associated ASAI user data, which was previously sent in a non-shared UUI, and the UCID data.

LAI backward compatibility issues

The following list summarizes LAI backward compatibility issues:

- An ISDN trunk group option is provided in the SETUP message for LAI interflowed calls to specify whether to include an LAI IE (codeset 6 or 7). When this option is set to **y** (default), an LAI interflow (using the existing or enhanced LAI vector command) will include a codeset 6/7 LAI IE to provide inter-operability in a mixed communication server environment with pre-Definity R6 systems. The option must be set to **n** if the network does not support codeset 6/7 or this IE is not required.

  Important:
  Codeset 0 information transport by way of shared UUI is required for BSR polling calls.

- Administer the ISDN Trunk Group option: Send Codeset 6/7 LAI IE. This option is valid even if LAI at the remote site is not active for tandem situations. Use of this option for LAI does not depend on the setting of the Vectoring Best Service Routing customer option.
- If the ISDN trunk group option is set to send the LAI IE, this IE is sent in addition to the Information Forwarding by way of codeset 0 shared UUI transport when a call is LAI interflowed over a trunk in this trunk group. With shared UUI, you can set the LAI data to be excluded in the UUI IE.
- Administer the Shared UUI priorities for ISDN trunks. This is important when the network byte limit on the user data part of the UUI user information contents is not large enough to carry the data that is needed for the customer application. Note that Shared UUI priorities do not apply to QSIG. To determine customer application data sizes, see Determining user information needs on page 196. For instructions on how to administer Shared UUI, see *Feature Description and Implementation for Avaya Communication Manager*.

ASAI shared UUI data conversion

The outgoing trunk interface treatment controls whether the ASAI data format is shared or non-shared:

- If the outgoing trunk interface is non-shared, ASAI UUI data stored in shared format is converted to the non-shared (service provider) format.
- If the outgoing trunk interface is shared, ASAI UUI data stored in shared format is sent in shared format.

Determining user information needs

This section includes the following topics:

- User information rules on page 196
- Bytes length ranges for UUI user data on page 198
- Example on page 199

User information rules

The network byte limit on the user data part of the UUI user information must be large enough to carry the data that is needed for the customer application.

The UUI IE for ISDN uses 3 bytes for the IE header information and allows from 32 bytes to 128 bytes for the user data portion. For example, if the network specifies that it can transport 32 bytes of user data, the UUI IE header length is 35 bytes.

The UUI for SIP includes the same data as the ISDN UUI IE except for the first 2 bytes in the UUI IE header information. For example, if the network specifies that it can transport 32 bytes of user data, the UUI header length is 33 bytes.

The user information capacity need is determined by adding the space that is required for each data item to be transported based on the following rules.

Minimum and maximum byte lengths: A maximum of 128 bytes of user data is supported by the communication server for UUI. Some network providers might impose limitations on the amount of user data they support.

User data length: Each shared data item requires 2 bytes for the application header. The application header is the application identifier plus the application data length. The application data length depends on the configuration of the customer application, except for UCID, In-VDN time, and Other LAI. These applications have a fixed byte length. For more information, see Bytes length ranges for UUI user data on page 198.

Byte length overruns: If the administered over run for the data length is exceeded, the lowest priority items are not included until the remaining data fits. If a specific data item at a higher priority exceeds the administered **Maximum UUI Size** setting, that item is not sent, leaving room for other lower priority items.

Priority settings: If the data item priority is set to blank in the **Shared UUI Feature Priorities** page in the Trunk Group administration form, the data item is not sent and no space is allocated for it.

QSIG considerations: QSIG signaling and networks do not have user information size limits. They will support sending MSI for user data items at their maximums. Determination of space allocation and administration of priorities does not need to be done for QSIG networks.

ASAI data length considerations: If the network supports 128 bytes and 78 bytes or less of ASAI user data is required, you do not need to determine space allocation or administer priorities.

If your ASAI user data is greater than 78 bytes, you can have up to 96 bytes of ASAI user data (98 bytes with the application header). The need for other interflow shared data transport must be carefully considered when setting priorities and determining how much ASAI user data to support for the application. If the network supports the full 128 bytes and all interflow data at their maximums is transported (48 bytes), the maximum length for ASAI user data is 80 bytes (78 bytes plus the application header). If the full 96 bytes of ASAI user data is required (plus 2 bytes for the application header), then only 30 bytes is available for other interflow data.

With the service provider trunk group for UUI transport (non-shared UUI), the UUI transports only ASAI UUI. The first byte of the UUI contains the UUI codepoint, which is 0x7E, followed by the byte that is set to the length of the following bytes. The next byte is the UUI protocol discriminator which, is unusually set to 0x00, that means the following data (ASAI UUI data) is in a user specified format. Another possible setting for the protocol discriminator is 0x04, meaning that the following data is in the IA5 (ASCII) character format. With service provider non-shared UUI format the protocol discriminator setting is retained by Communication ManagerCommunication Managerand included in the ASAI UUI IE sent to a CTI adjunct.

The shared format for the trunk group UUI transport 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.

The ASAI UUI portion of the shared UUI does not contain a protocol discriminator field that is included with the non-shared UUI. When an ASAI UUI IE is created for passing to an adjunct, Communication Manager inserts a 0x00 for the protocol discriminator.

Note:

In most cases the application should not rely on or depend on the setting of the protocol discriminator since it usually is not meaningful other than to account for it as being part of the format.

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 CM, otherwise the protocol discriminator setting is not changed. In the case of shared the UUI IE protocol discriminator is set to 0x04 regardless of what else is carried by the shared UUI. The set command defines the data bytes following the protocol discriminator which 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).

Bytes length ranges for UUI user data

The following table specifies minimum and maximum byte lengths used to send user data over call center networks.

Type of user data	Total user data bytes (with 2-byte header)	Description
ASAI	2 to 98 or 0 (calculated by 1 byte per byte of ASAI user information)	Required for certain CTI applications when the CTI application sends user information and the amount of space is determined by the application. For example, 34 bytes is required if the application sends 32 bytes of data. Sending more than 78 bytes of ASAI data (80 bytes with the application header) reduces capacity for other interflow data.
UCID	10 or 0	Used by BSR to track calls across multiple sites. Trunk group setting and/or system feature settings control transport of UCID data, even when the priority is set to 1. When the data item is not included, it does not take up any space.
In-VDN Time	4	Used by BSR to determine time before answer and call tracking across sites. This data type can be eliminated when short waiting times are anticipated. If the priority field is not blank, it is always included.
VDN Name	2 to 17 (calculated by 1 byte per character in name) maximum of 15	Used by BSR, but can be eliminated if receiving sites use dedicated VDNs that display equivalent information to the answering agent. An interflowed call that is received without the originating VDN name uses the incoming VDN name. If the priority field is not blank, the 2-byte header is always included.
Collected Digits	4 to 11 or 0 (calculated by 1 byte per 2 digits plus 1) maximum of 16 digits	Requires a whole byte for an odd number of digits. For example, 1 digit requires 2 bytes (1 plus 1), 7 digits need 5 bytes (4 plus 1), and 16 digits need 9 bytes (8 plus 1).
Other LAI Info	6	Required for existing CTI applications that use any of the following obtained from the from the LAI IE: • in-queue time stamp • queue priority • interflow type

Example

Assume that your public network supports only 32 bytes of user information. Your application requires 13 bytes of ASAI user information (15 bytes of user data), UCID (10 bytes of user data), and 8 collected digits (7 bytes of user data - 4 plus 1 plus 2 for the header). It does not require Other LAI Information. Also, call time at the sending communication server is brief because calls are not queued before interflow takes place and tracking as a single call is not required.

By dedicating appropriately named VDNs at the receiving communication server, the public network can support the application. Because the needed data items require the entire 32 bytes of user data, the priority fields for the **In-VDN Time**, **VDN Name**, and **Other LAI Information** must be set to blank.

Information Forwarding troubleshooting

In some circumstances, UUI data may not be forwarded, even though you received no error messages while administering the Shared UUI feature, and all software and connections meet the minimum requirements. The following list provides items that can be evaluated to troubleshoot the problem:

Tip:
When a new application is implemented, run the `display events` command on a periodic basis for the appropriate vector. The resulting report notifies you if any UUI data could not be sent.

- If DCS is used, ensure that all ISDN trunks between communication server that are used for DCS or remote AUDIX are configured in the D-channel mode.
- For each ISDN or SIP trunk that is administered with the Shared UUI option, make sure that the UUI size does not exceed the UUI size that the network can support. For more information, see Determining user information needs on page 196.
- Verify that trunk group options are set correctly for the application and configuration.
- Applications may fail on networks supporting limited UUI transport. Administration determines which application's UUI will be transported in these cases. If a given application is failing, first check the administration to determine if the application in question has the highest priority. This applies to tandem nodes as well as to originating nodes.

- Applications that originate UUI on tandem nodes can request that assigned priorities at the tandem node be applied to the resulting UUI. Therefore, it is possible for a tandem node to erase UUI information that was received from the originator. Passing UUI through a tandem node transparently, as required for UUS Service 1, does not apply to communication server proprietary shared UUI procedures.

Adjunct (ASAI) Routing

This section includes the following major topics:

- About Adjunct Routing on page 201
- Considerations for implementing adjunct routing on page 202
- Receiving and implementing an ASAI call route on page 203
- Data sent with an ASAI call route request on page 205
- Special vector processing considerations associated with adjunct routing on page 206
- Adjunct routing-initiated path replacement on page 211
- Phantom calls on page 212
- Single-step conference on page 214
- Multiple outstanding route requests on page 215

About Adjunct Routing

Adjunct Routing provides a means for an Adjunct Switch Application Interface (ASAI) processor to specify the destination of a call when it encounters an **`adjunct routing link`** vector command during vector processing.

An adjunct is any processor that is connected to a switch that can use the ASAI protocol. The adjunct makes a routing decision according to caller information and/or agent availability, and returns a call route response to the switch.

The switch provides information in an ASAI route request message that the adjunct application uses to access a database and determine a route for the call. In a typical application, the ASAI adjunct might use the dialed number, the Calling Party Number (CPN/BN), or the digits that are collected by way of Call Prompting to access caller information and thereby determine an appropriate call route.

Adjunct Routing can be used in conjunction with the Call Prompting and Look-Ahead Interflow features. When combined with one of those features, the following rules apply:

- When combined with Call Prompting, Adjunct Routing can pass up to 16 digits that are collected from the last relevant **`collect digits`** vector command.
- When combined with Look-Ahead Interflow (LAI), Adjunct Routing can pass the LAI information element or other call center-related data (with enhanced Information Forwarding) that was passed from the originating switch in the message or associated with the call from the local switch.

Considerations for implementing adjunct routing

You should understand the following considerations before you implement a call center solution that uses the Adjunct Routing feature:

- 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 particular agent. An adjunct can also provide priority ringing, priority queuing, and specify that a route-to an agent be done as a direct agent call.
- If your specific application permits you to do so, you can include two or more consecutive **adjunct routing link** steps in a vector. This approach provides the following advantages:
  - Redundancy in case of ASAI link/application failure.
  - Simultaneous processing of multiple route requests, which distributes incoming call load more efficiently and results in faster call processing times. For more information, see Multiple outstanding route requests on page 215.
- Vector processing continues to occur while an ASAI route request is being processed. For this reason, the first step to follow one or more **adjunct routing link** steps should be either an **announcement**, or a **wait time** step that adheres to the following rules:
  - If an **announcement** step follows immediately after an **adjunct routing link** step, the announcement should not contain any information that is essential to the caller (such as further instructions), since it will immediately terminate when the switch receives a destination from the ASAI adjunct.
  - If a **wait-time** step follows immediately after an **adjunct routing link** step, it should usually specify either **ringback** or **music** (but not **silence**) as the feedback option, so that the caller is less likely to abandon the call.

  Important:
  If an ASAI link/application specified in the **adjunct routing link** step is out of service, the step is skipped. If the next step is not a **wait-time**, **announcement**, or **adjunct routing link** step, as much as six minutes may elapse before the switch determines that the adjunct application is out of service.
- The second step after the **adjunct routing link** step can, and often should, be implemented as a default treatment in case the host application or ASAI link is down. Speed of execution for the default treatment step (for example, **route-to number 0 if unconditionally**) is controlled by the following factors:
  - If the ASAI link is down, and if the first non-**adjunct routing link** step is either a **wait-time** or an **announcement** treatment, then the treatment step is skipped and the default step that follows the skipped treatment executes immediately.

  - If the host application is not down, the default step executes only if the adjunct does not provide a route within the time defined by the first non-adjunct step. For example, if the first non-adjunct step is an announcement, the default step executes only after the time defined by the length of the announcement is exceeded.

- When a vector contains an **adjunct routing link** command, and an ASAI link/application failure event occurs, special rules apply to vector processing operations that result. Adjunct Routing vectors should be designed to take these special processing operations into account. For more information, see Special vector processing considerations associated with adjunct routing on page 206.
- Since vector processing continues to occur while an ASAI call route request is processed at an adjunct, succeeding vector steps can terminate an ASAI call route request if they execute before a call route can be provided by the adjunct. Alternately, the adjunct may reject the call route request, and subsequent vector processing proceeds in a normal manner. For more information, see Vector steps that terminate an ASAI call route request on page 210.
- The **wait-time hearing i-silent** command is used in cases where it is important 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, the switch does not return an ISDN PROGress message to the originating switch. This is particularly important for Network ISDN features and the Look-Ahead Interflow feature.

Receiving and implementing an ASAI call route

A switch that receives an adjunct-supplied call route performs various checks to validate the call route before it is implemented. When the adjunct-supplied route is validated, the operations that result are similar to those in effect for a **route-to *xxxxx* with coverage=y** command. The caller hears normal call progress tones and feedback, and if the call routes to an extension with no available call appearances and no coverage path, the caller hears a busy signal.

Any other features that may be in effect at the adjunct-supplied destination, such as Send-All-Calls or Call Forwarding, interact with the routed call.

Also, Look-Ahead Interflow operations are not applied when calls are routed over trunks. Instead, ASAI-routed calls are directed to their adjunct-supplied destination without waiting for call acceptance.

The processes associated with receiving and implementing and ASAI call route are described in the following sections:

- Validation requirements for an adjunct-supplied call route on page 204
- Switch response to validated adjunct-supplied call routes on page 204
- Switch response to invalid adjunct-supplied call routes on page 204

Validation requirements for an adjunct-supplied call route

When the switch receives adjunct-supplied call route instructions, the switch validates the route according to the following process:

1. The switch verifies that the COR rules specified for the target VDN permit the call to be terminated at the adjunct-supplied destination.
2. The switch validates the following information:
   - Destination number
   - ACD split
   - TAC/AAR/ARS access code
   - Dial plan compatibility
   - Other options specified by the adjunct
3. If the ASAI adjunct specifies the direct agent call option, the destination number (agent) must be logged into the adjunct-specified ACD split.
4. If the destination for the call is external, the switch verifies that a trunk is available for the call.

Switch response to validated adjunct-supplied call routes

If the switch validates an adjunct-supplied call route, the following operations occur:

1. Vector processing in the VDN that contains the initiating **`adjunct routing link`** command terminates immediately.
2. The switch signals the ASAI adjunct that the route is accepted.
3. The switch routes the call to the destination specified by the ASAI adjunct.

Switch response to invalid adjunct-supplied call routes

If any of requirements for call route validation listed in Validation requirements for an adjunct-supplied call route on page 204 are not met, items the following operations occur:

1. The switch discards the route.
2. The switch signals the ASAI adjunct that the route is invalid.
3. Vector processing of any other default treatment steps in the VDN that contains the initiating **`adjunct routing link`** proceeds.

Data sent with an ASAI call route request

When a call encounters an **adjunct routing link** command and if the call is not queued to a split, the switch sends an ASAI message that requests a call route over the specified adjunct link. The following list identifies the contents of the message, along with a comment or a brief explanation for each item:

Calling number information : The calling party number or billing number (CPN/BN) that is provided by ISDN-PRI or R2-MFC signaling facilities. If the call originates from a local switch extension, this extension is the calling number.

Originating line information (II-digits): A two-digit code that is provided by ISDN-PRI facilities that indicates the type of originating line. This information is not provided by SIP facilities.

Called number : The originally called extension if a call is forwarded to a VDN, or the first VDN through which the call was routed if the call was not forwarded to the VDN.

If the VDN Override for the Trunk ASAI Messages feature is in effect for an incoming call, the active VDN extension (instead of the Called Number received in the SETUP or INVITE message) is sent in the Called Number for the Call Offered, Alerting, Queued, Connect, and Adjunct Route-Request ASAI Event Reports.

Routing VDN: The 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 by either Event Notification or 3rd Party Call Control. For more information on ASAI, see *Avaya Communication Manager CallVisor ASAI Technical Reference*.

Enhanced Information Forwarding (related data) and Look-Ahead Interflow information (if any) : Includes the original VDN display information, the priority level of the call at the originating switch, and the time that the call entered vector processing. For more information, see Look-Ahead Interflow (LAI) on page 273, and Information Forwarding on page 189.

Digits collected by Call Prompting or Caller Information Forwarding (CINFO) (if any; maximum of 16 digits): Digits that are collected by the most recent **collect digits** command. For more information, see Call Prompting on page 237, ANI /II-digits routing and Caller Information Forwarding (CINFO) on page 169, and Information Forwarding on page 189.

User-to-User Information (UUI): User-provided data that is associated with the call. If provided by ASAI, this data was provided in a 3rd-Party-Make-Call, Auto-Dial, or Route-Select message. If provided over an ISDN or SIP trunk, the data was in the SETUP or INVITE message that delivered the call to this switch. Calls that contain UUI specifically used by ASAI allow ASAI UUI to be propagated to the new call during a manual transfer or conference

operation. ASAI UUI is propagated to a new call during its establishment when the agent presses the transfer/conference button the first time. If the call is transferred to a remote switch, the ASAI UUI from the first call is copied into the SETUP or INVITE message sent for the second call, in which case, the alerting event message sent to an ASAI application contains the ASAI information.

Special vector processing considerations associated with adjunct routing

When you design call vectors that include one or more **adjunct routing link** commands, you must be aware of a number of special operational features. These considerations are described in the following sections:

- Effects of ASAI link/application failure on vector processing on page 206
- Simultaneous processing of vector steps and ASAI call route requests on page 210

Effects of ASAI link/application failure on vector processing

An ASAI link failure can change the manner in which subsequent **announcement** or **wait-time** treatment steps are processed.

In the following simplified vector example, the step that follows immediately after an **adjunct routing link** command is a **wait-time** command. If the **adjunct routing link** step fails at either the ASAI link or adjunct application, the wait-time step is skipped.

The second step after the **adjunct routing link** step is often implemented as a default treatment. In the example shown above, the default treatment in step 3 is a route to an attendant. If the switch recognizes that the ASAI link or adjunct application is out of service, this step executes immediately. Otherwise, the step executes only if the application does not respond with a route within 60 seconds (the wait-time assigned in the example).

Simplified example of vector processing in an ASAI link/application failure condition

```
1. adjunct routing link 11  [link/application is down]
2. wait-time 60 seconds hearing ringback  [step is skipped]
3. route-to number 0 with cov n if unconditionally [step is executed]
4. disconnect after announcement 2000
```

Vector processing with goto steps in an ASAI link/application failure condition

Processing rules for a vector that includes one or more **adjunct routing link** commands and has an ASAI link/application failure condition in effect are summarized as follows:

- An **announcement** or **wait time** treatment is skipped whenever one of the following conditions is true:
  - The treatment step follows immediately after a failed **adjunct routing link** command
  - The treatment step is the first non-goto step that follows a **goto** step that succeeds. In this context, a **goto** step is considered to succeed when the specified **goto** condition is true, and the call branches from the **goto** step to the treatment step.
  - The treatment step is the first non-goto step that follows a failed **goto** step. In this context, a **goto** step is considered to fail when the specified **goto** condition is true, the call fails to branch, and control proceeds to the treatment because it is the next step listed in the vector sequence.

Note:
The treatment step is skipped even when a failed **goto** step that precedes it is, in turn, preceded by one or more successful **goto** steps.

The rules listed above for vector processing under ASAI link/application failure conditions are further illustrated in the following examples.

Example 1 - Vector processing with goto steps in an ASAI link/application failure condition

```
VDN (extension=1040  name=''Ad Route''  vector=40)
Vector 40

 1.adjunct routing link 10  [link/application is down]
 2.wait-time 10 seconds hearing ringback [step is skipped]
 3.adjunct routing link 20 [link/application is down]
 4.goto step 7 if available-agents in split 20 < 1 [step executes and condition is false]
 5.wait-time 10 seconds hearing ringback [step is skipped]
 6.goto vector 50 @step 1 if unconditionally  [step executes, go to vector 50]
 7.goto step 10 if calls-queued in split 20 pri l > 50
 8.announcement 4001
 9.goto vector 50 @step 1 if unconditionally
10.route-to number 6000 with cov n if unconditionally
  VDN (extension=6000  name=''Message''  vector=60)
```

Based on the scenario presented in the example shown above, the following vector processing events occur:

Step 1 fails : For purposes of this example, assume that the adjunct link or application is out of service. The **adjunct routing link** command in step 1 fails.

Step 2 is skipped : Because the **wait-time** command in step 2 immediately follows an **adjunct routing link** command whose adjunct link is out of service, the **wait-time** step is skipped.

Step 3 fails : For purposes of this example, step 3 contains another **adjunct routing link** command whose adjunct link is assumed to be out of service. The step fails, and control is passed to the **goto step** command in step 4.

Step 4 executes: A **goto** step that immediately follows a failed **adjunct routing link** command is always executed. In this example, the command fails to branch because there is at least one available agent in split 20.

Step 5 is skipped: The **wait-time** step that follows the unsuccessful **goto** step (step 4) is skipped, because in an ASAI link failure condition, the first non-goto step to be processed after the first successful first goto step is always skipped if it is either **announcement** or **wait-time**. Control is passed to the **goto vector** command in step 6.

Step 6 executes: Step 6 routes the call to vector 50 (not shown), which is designed to queue the call and provide standard call treatment.

In the next example, assume that the **goto step** command in step 4 succeeds. In this context, the **goto** step succeeds when the specified condition is true (no agents are available in Split 20), and control is passed to step 7, where another **goto** step determines whether there are more than 50 calls in split 20. If the condition is true, step 7 succeeds and control is sent to step 10, where the **route-to number** command sends the call to vector 60.

The example processing events are described in the following figure.

Example 2 - Vector processing with goto steps in an ASAI link/application failure condition

```
VDN (extension=1040  name=''Ad Route''  vector=40)
Vector 40

 1.adjunct routing link 10 [link/application is down]
 2.wait-time 10 seconds hearing ringback [step is skipped]
 3.adjunct routing link 20 [link/application is down]
 4.goto step 7 if available-agents in split 20 < 1 [step executes and condition is true]
 5.wait-time 10 seconds hearing ringback
 6.goto vector 50 if unconditionally
 7.goto step 10 if calls-queued in split 20 pri l > 50 [step executes and condition is true]
 8.announcement 4001
 9.goto vector 50 if unconditionally
10.route-to number 6000 with cov n if unconditionally [step executes unconditionally]

VDN (extension=6000  name=''Message''  vector=60)
Vector 60

1.announcement 4000 [We're sorry.  We are still unable to connect you to an agent. If you'd
     like to leave a message, please do so after the tone.]
2.wait-time 6 seconds hearing silence
3.messaging split 18 for extension 1500
4.announcement 4010 [We're sorry.  We were unable to connect you to our voice mail. If
     you'd like to try to leave a message again, please do so after the tone. Otherwise,
     please call back weekdays between 8:00 A.M. and 5:00 P.M.]
5.goto step 2 if unconditionally
```

Based on the scenario presented in the example shown above, the following vector processing events occur:

Step 1 fails : For purposes of this example, the adjunct link or application is out of service. The **adjunct routing link** command in step 1 fails.

Step 2 is skipped : Because the **wait-time** command in step 2 immediately follows an **adjunct routing link** command whose adjunct link is out of service, the **wait-time** step is skipped.

Step 3 fails : For purposes of this example, step 3 contains another **adjunct routing link** command whose adjunct link or application is also out of service. The step fails, and control is passed to the **goto step** command in step 4.

Step 4 executes: A **goto** step that follows a failed **adjunct routing link** command is always executed. In this example, the command succeeds and branches to step 7, because no agents are available in split 20.

Step 7 executes: Again, a **goto** step that follows a failed **adjunct routing link** command is always executed. In this example, the command branches unconditionally to Vector 60

Step 10 executes: In this example, step 10 (**route-to number**) is the first non-goto step immediately preceded by one or more **goto** steps in an ASAI link fail condition. The step executes, because it not an **announcement** or **wait time** command.

Vector 60: Step 1 executes: The first step in this vector is an **announcement** command. In this example, this is the first step in the processing sequence to be either an announcement or wait time step. However, this step is not skipped, since it is not the first non-go to step in the processing sequence. Instead, step 10 in Vector 40 (a **route-to number** step) is the first non-goto step.

Simultaneous processing of vector steps and ASAI call route requests

When the switch sends a route request to an ASAI adjunct, vector processing continues for any vector steps that follow the **adjunct routing link** command. Therefore, non-**adjunct routing link** step that follows immediately after an **adjunct routing link** step (or multiple **adjunct routing link** steps in uninterrupted succession) can determine:

- The maximum length of time that the switch waits for a call route reply from the ASAI adjunct
- In some cases, whether or not the ASAI call route request is allowed to finish processing

If the next step is not a **wait-time**, **announcement**, or another **adjunct routing link** command, as much as six minutes may elapse before the switch determines that the adjunct application is out of service. For this reason, the recommended practice is to design vectors so that the next step to follow an **adjunct routing link** command is either a **wait-time**, or **announcement** step.

Vector steps that terminate an ASAI call route request

If an **adjunct routing link** step is followed by a **wait-time** or **annoucement** treatment, and the treatment completes before an ASAI call route request is returned by the adjunct, call processing continues for any vector steps that may follow the treatment. In this case, certain vector commands will terminate the ASAI call route request when they are executed. Vector commands that terminate an active ASAI call route request include:

- **busy**
- **check split**
- **converse-on split**
- **queue-to split**
- **collect digits**
- **disconnect**
- **messaging split**
- **route-to**

If a valid ASAI call route message is received by the switch before one of the vector commands listed above can execute, the system routes the call to the destination specified by the adjunct route. Otherwise, the ASAI route request is terminated.

Note:

The adjunct can also reject a call request by negatively acknowledging the route request that is sent by the switch. When the switch receives a a route request rejection message from the adjunct, any **announcement** or **wait-time** step that is being executed is immediately terminated. Call processing then continues with the next vector step.

Adjunct routing-initiated path replacement

Path replacement for calls in queue and vector processing, using QSIG or DCS with Reroute using SSE, is available for Avaya switch software R9.5 or later. For calls that are waiting in queue or in vector processing, even if the call is not connected to an answering user, path replacement can be attempted to find a more optimal path for this call. This results in more efficient use of the trunk facilities.

When adjunct routing is used with a call, path-replacement can be initiated when the following criteria are true:

- The inbound call is over a QSIG trunk or DCS SSE trunk
- A route-select response is received from the CTI application after the **adjunct route** vector command has been executed
- The routing destination that is contained in the route select ASAI message is to an outbound QSIG trunk or out bound DCS SSE trunk

When all three criteria are met, the trunk is then seized and used for the call.

The ability to track a measured ACD call after a path replacement has taken place is available for CMS versions r3v9ai.o or later. Starting with the r3v12ba.x release, CMS reports a path replacement as a *rename* operation rather than a path replacement. The *rename* operation properly reports scenarios where a path replacement takes place from a measured to an unmeasured trunk facility. Avaya recommends that you upgrade CMS to r3v12a.x or later and administer all trunks associated with path replacement as *measured* by CMS to ensure better CMS tracking of path-replaced calls.

Example vector for adjunct routing-implemented path replacement

The following Call Vector example shows how a vector for adjunct routing can be written to trigger path-replacement at the terminating switch.

Note:
In order for a path-replacement to be attempted, the incoming and outgoing trunks that are used for the call must be administered with the **Supplementary Service Protocol** field set to **b**.

Adjunct routing-initiated path-replacement vector

```
1. announcement 5996
2. adjunct routing link 11
3. wait 20 seconds hearing ringback
4. announcement 3111
```

At the terminating (receiving) switch, the vector that is executed by the incoming call must be programmed with an **announcement**, **wait hearing music**, or **wait hearing ringback** vector command. The use of one of these commands is what makes it possible for path-replacement to take place while the call is in vector processing.

Phantom calls

A phantom call is a call that originates from a nonphysical device by way of an ASAI application and may be placed anywhere. In general, phantom calls

- Use less resources
- Are treated like voice calls

This section includes the following topics:

- How do phantom calls work? on page 213
- How are phantom calls used? on page 213
- How do phantom calls affect Call Vectoring? on page 213
- Phantom call administration on page 214

How do phantom calls work?

First, an application requests a phantom call by sending an ASAI third_party_make_call or auto_dial capability message to the switch.

If the specific extension of a station Administration Without Hardware (AWOH) is specified as the originator, the switch places the call from that extension if the extension is available.

It is also possible to specify a hunt group extension with members that are AWOH extensions as the originator.

How are phantom calls used?

Applications use phantom calls when they need to originate a call without using a physical device and thus not use extra resources. For example, applications may need to:

Reserve a queue slot : Many call centers handle incoming requests as voice, video, data, voice messages, faxes, and e-mail. Agents who work in these call centers need to handle the mix of requests. However, a single queue needs to manage and distribute the work load for these agents.

For each non-voice request, the application can place a phantom call into the queue. When the phantom call reaches the head of the queue, it is delivered to the agent. The agent is then given the corresponding work item on the desktop, for example, the fax.

Conference control: Multiple parties (both internal and external) can be conferenced into a call. The initial call is placed as a phantom call. When answered, the call is placed on hold by the application and another phantom call is made. The two calls are then conferenced together. This process is repeated until all parties are added to the call.

Help with trunk-to-trunk transfers. Working with the Single Step Conference feature, applications can use the phantom call feature to help with trunk-to-trunk transfers, that is, transferring a trunk-to-trunk call to another trunk. For information about single step conferences, see *Avaya Communication Manager CallVisor ASAI Technical Reference*.

Alerts (wake-up, maintenance, and security): Applications can use phantom calls to alert users of various conditions such as wake-up, maintenance, or security.

How do phantom calls affect Call Vectoring?

Because phantom calls can be directed anywhere, you must properly configure the application and the switch to ensure that the vector commands that are executed for these calls make sense. For more information, see *Avaya Communication Manager CallVisor ASAI Technical Reference*.

The switch does not block phantom calls from executing any vector commands because phantom calls follow the same vector processing as regular voice calls. However, it might not make sense to have phantom calls enter certain vector steps such as:

Announcements : Because there is nobody listening to an announcement that is made to a phantom call, there is no sense in playing one.

collect steps: In a phantom call, the **collect** step fails because it can not connect a tone receiver to a station Administration Without Hardware (AWOH); it times out because there is nobody to put in the expected digits.

The **busy** step provides a busy signal to the caller. In a phantom call, the **busy** step disconnects the call because the switch clears a phantom call when the call cannot terminate at a specific local destination.

Phantom call administration

There are no administration screens that are specific to phantom calls, but the following criteria must be met in order for the feature to work:

- Some stations AWOH must be administered.
- If a hunt group is specified as originator, a non-ACD hunt group with AWOH members must also be administered.
- It is recommended that meaningful names are assigned for the stations AWOH that are used by phantom calls if the calling party name will appear on the agent's or Service Observer's display.

Single-step conference

The Single-Step Conference (SSC) feature is available for Avaya switch software R6.3 or later. SSC allows an application to:

- Add a device into an existing call, for example, to play announcements or make voice recordings
- Facilitate application-initiated transfers and conferences

Stations that are AWOH are eligible for single-step conference. The party may be added to a call in listen only mode (no visibility) or with listen and talk capability (visibility).

Single-step conference is only available through an ASAI link. For more information about single-step conference, see *Avaya Communication Manager CallVisor ASAI Technical Reference*.

How does SSC work with Call Vectoring?

The call to which an extension is to be single-step conferenced is not allowed to be in vector processing unless the visibility option with the single-step conference request indicates no visibility.

To be transferred to a VDN, a conference call must not have more than two parties.

Note:
Invisible (listen-only) single-step-conference parties are not counted in the two-party limit for a conference call transfer to a VDN.

Multiple outstanding route requests

This feature allows multiple ASAI route requests for the same call to be simultaneously active. The route requests can be over the same or over different ASAI links.

Route requests are all made from the same vector. They must be specified without intermediate (**`wait-time`**, **`announcement`**, **`goto`**, or **`stop`**) steps. If the **`adjunct routing link`** commands are not specified back-to-back, standard adjunct routing functionality applies and previous outstanding route requests are cancelled when an **`adjunct routing link`** vector step is executed.

The first route select response that is received by the switch is used as the route for the call and all other outstanding route requests for the call are canceled.

With multiple outstanding route requests, multiple adjuncts can process the route call request without waiting for the first route attempt to fail. An application can make use of this feature to distribute the incoming call load evenly across adjuncts based on the adjunct's current CPU load.

Note:
Each link has a unique extension number, even in a configuration where there might be multiple links to the same adjunct.

Multiple call route request example

The following example shows a typical vector where multiple adjunct route requests to multiple links are active at the same time. The first adjunct to route the call is the active adjunct and it specifies which VDN the call should be routed to at that point.

Sample adjunct routing link vector with redundancy

```
1. wait-time 0 seconds hearing ringback
2. adjunct routing link 11
3. adjunct routing link 12
4. adjunct routing link 13
5. wait-time 6 seconds hearing ringback
6. route-to number 1847 with cov n if unconditionally
```

Creating and editing call vectors

This section gives you a practical start writing vectors. You will learn the basic information that you need to write a representative vector and enter it online.

This section includes the following topics:

- Methods for entering a vector online on page 217
- Call Vector screen - basic administration on page 218
- Displaying vector variable information on page 221
- Inserting a vector step on page 224
- Deleting a vector step on page 224
- Entering a comment out indication to an existing vector step on page 225
- Removing a comment out indication on page 226
- Creating and constructing a vector on page 226
- Duplicating Vectors on page 234

Methods for entering a vector online

A vector can be entered online using basic screen administration on the system administration terminal by any of the following three methods:

- Basic screen administration on the system administration terminal
- Avaya Call Management System (CMS)
- Avaya Visual Vectors

The following section discusses the basic screen administration method for entering a vector online at the switch system administration terminal. For instructions on creating a vector using the CMS interface, see *Avaya CMS Administration*. For instructions on creating a vector with Visual Vectors, see *Avaya Visual Vectors User Guide.*

Call Vector screen - basic administration

A vector is entered online using basic screen administration by completing the Call Vector screen. An example the first page of this screen is shown in the following screen example.

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 you can enter a vector online using basic screen administration.

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

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

2. Assign a name to the vector by completing the blank next to the **Name** field. The vector name can contain up to 27 alphanumeric characters.

   Note:
   The vector number, which appears next to the **Number** field, is automatically assigned by the system.

3. In the **Multimedia?** field, indicate whether the vector should receive early answer treatment for multimedia calls. Valid values are y or n.

   Note:
   This only applies if Multimedia Call Handling is enabled.

- If you expect this vector to receive multimedia calls, set this field to y. The call is considered to be answered at the start of vector processing, and billing for the call starts at that time.
- If you do not expect the vector to receive multimedia calls, set this field to n.

4. In the **Attendant Vectoring** field enter a **y** if the vector will be used as an attendant vector. Attendant Vectoring can be used only when enabled on the Customer Options screen.
5. In the **Meet-me Conf** field enter a **y** if the vector will be used for the Meet-me Conference feature. Meet-me Conference can be used only when enabled on the Customer Options screen.

Note:
Both Attendant Vectoring and Meet-me Conference cannot be enabled for a vector at the same time.

6. In the **Lock** field, indicate whether you will allow this vector to be displayed on and edited from a client application such as Visual Vectors.
   - If you enter **y**, the vector is locked and can only be displayed and modified in the switch administration software.
   - If you enter **n**, the vector is not communicated to client software such as Visual Vectors or CMS and may not be displayed and modified from these programs.
   - If Attendant Vectoring is enabled, the **Lock** field defaults to **y** and cannot be changed.

Note:
Always lock vectors that contain secure information, for example, access codes.

7. Look at the next fields and determine where a y (yes) appears. These fields indicate the Call Vectoring features and corresponding commands you can use. If an n (no) appears in one of these fields, you cannot use the corresponding feature.

Note:
The Call Vectoring features are optioned from the Customer Options screen.

Basic	You can use the Basic Call Vectoring commands. See Basic Call Vectoring on page 103 for details on using these commands.
EAS	Expert Agent Selection is enabled. See Expert Agent Selection on page 455 for information on how the EAS feature works.
G3V4 Enhanced	You can use the G3V4 Enhanced Vector Routing commands and features. See Appendix Q: Feature availability on page 851 for an explanation of which features are included with G3V4 Enhanced Vector Routing.
ANI/II-Digits	You can use the ANI and II-Digits Vector Routing commands. See ANI / II-digits routing and Caller Information Forwarding (CINFO) on page 169 for details on using these commands. ANI/II-Digits Routing requires G3V4 Enhanced Vector Routing.

ASAI Routing	You can use the Adjunct Routing command. See Adjunct (ASAI) Routing on page 201 for details on using this command.
Prompting	You can use the Call Prompting commands. See Call Prompting on page 237 for details on using these commands.
LAI	Look-Ahead Interflow is enabled. See Look-Ahead Interflow (LAI) on page 273 information on how LAI works.
G3V4 Adv Route	You can use the G3V4 Advanced Vector Routing commands. See Advanced Vector Routing - EWT and ASA on page 257 for details on using these commands.
CINFO	You can collect ced and cdpd digits with the collect digits step. See ANI / II-digits routing and Caller Information Forwarding (CINFO) on page 169 for information on collecting these digits.
BSR	Best Service Routing (BSR) is enabled, and you can use the BSR commands. The available commands vary depending on whether you are using single-site or multi-site BSR. See Best Service Routing (BSR) on page 297 for information on the application of BSR.
Holidays	You can create tables to use for special days, such as holidays and promotional days. See Holiday Vectoring on page 357 for information on how to create holiday tables and define holiday vectors.

8. Enter a maximum of 99 vector commands in the blanks next to the step numbers. See Call Vectoring commands on page 521 for a complete description of all Call Vectoring commands.

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

9. Save the vector in the system by pressing Enter.

   Note:
   After editing a vector, verify that the vector will work as intended. This is particularly important if you deleted a step that was the target of a `go-to` step.

Displaying vector variable information

You can view vector steps while using the **change vector** screen.

This section includes the following topics:

- How to view vector variable information on page 221
- Display fields on page 222
- Examples on page 223

How to view vector variable information

To display vector variable information from the Variables in Vectors table:

1. While using the **change vector** command, press **Esc f 6**.
2. After the **Edit (i/d/v/c/u)** prompt, enter a **v**, plus the variable you want to view.

   Enter an A-Z or AA-ZZ value.

   Example: **v G**
3. 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 f 6' for Vector Editing

Var G: value type VALUE G L=1 ASGN=[5] VAC=VV1
```

Display fields

The following line is displayed on the bottom of the **Call Vector** screen after you perform How to view vector variable information on page 221.

Var *letter*: *description type scope [***L=***length* **S=***start* **ASGN=***current_value* **VAC=***fac]*

Display field	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`.
type	Displays the vector variable type. For example, `VALUE`.
scope	Displays the defined scope as L (local) or G (global).
The following items included in the brackets are displayed only if the item is applicable for the vector variable type.	
L=*length*	If Length is allowed for this variable type, this field displays the defined maximum digit length for the variable.
S=*start*	If Start is allowed for this variable type, this 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=[] is displayed.
VAC=*fac*	If the value type variable is defined, this field displays the Variable Access Code, VV1 through VV9.

Examples

Refer to the values assigned in Table A when 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	1234567890123456	
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	
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]

Table B	
For	**Based on the values in Table A, the following text is displayed**
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

Inserting a vector step

To insert a vector step:

1. After entering the **change vector** command, press **Esc f 6**
2. At the command line, type **i** followed by a space and the number of the step that you want to add 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.

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

You cannot insert a new vector step if 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.

Deleting a vector step

To delete a vector step:

1. After entering the **change vector** command, press **Esc f 6**

2. At the command line, type **d** followed by a space and the number of the step you want to delete 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.

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

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

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

Entering a comment out indication to an existing vector step

The vector editing function **c** capability inserts a **#** at the beginning of existing vector commands to comment out those vector steps. Once commented out, vector steps are skipped during normal vector processing. Commented out steps can be un-commented out for normal processing. See Removing a comment out indication on page 226.

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 2000 vectors or 198,000 steps.

To comment out an existing step:

1. After entering the **change vector** command, press **Esc f 6**.
2. At the command line, type **c** followed by a space and the number of the step you want 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.

Note:
The comment out indication functions differently than the **#** command. See # command on page 527 for details.

Removing a comment out indication

Once the comment out indication is removed from a vector step, that vector step will execute during vector processeng as before. Removing a comment out indication from a vector step that contains no comments does not affect vector processing.

To remove a comment out indication:

1. After entering the **`change vector`** command, press **Esc f 6**.
2. At the command line, type **`u`** followed by a space and the number of the step where you want 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.

Note:
The comment out indication functions differently than the **#** command. See [# command on page 527] for details.

Creating and constructing a vector

This section includes the following topics:

- About creating and constructing a vector on page 226
- Step 1: Queuing a call to the main split on page 227
- Step 2: Providing feedback and delay announcement on page 228
- Step 3: Repeating delay announcement and feedback on page 229
- Step 4: Queuing a call to a backup split on page 230
- Step 5: Limiting the queue capacity on page 232
- Step 6: Checking for non business hours on page 233

About creating and constructing a vector

This section provides a logical approach for vector construction. This method uses a starting vector that consists of one step and then builds on this vector to produce a new vector that provides additional functions. As each step is presented, you are introduced to one or more new vector commands or approaches to vector processing. While it is not practical to present all such commands and approaches, those presented in this tutorial should give you a good idea of how to use Call Vectoring.

Step 1: Queuing a call to the main split

If a call cannot be immediately answered by an agent or operator, the call is usually queued until an agent becomes available. A call can be connected to an available agent or queued 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 automatically sends the call to the agent without queuing the call. However, if no agent is available, the command queues the call to the main split of agents. Once the call is sent to the main split queue, the call remains there until it is answered by an agent or some other treatment is provided.

Call Priority levels

Each call queued to a split occupies one queue slot in that split. Calls are queued sequentially as they arrive according to the assignment of the priority level. In our vector, note that the priority level low is assigned to the call. The priority level establishes the order of selection for each call that is queued. A call can be assigned one of four priority levels: top, high, medium, or low.

Within a given split (the main split, in our vector), calls are delivered to the agent sequentially as they arrive to the split queue and according to the priority level assigned. Accordingly, calls that are assigned a top priority (if any) are delivered to an agent first, calls that are assigned a high priority are delivered second, and so forth.

Step 2: Providing feedback and delay announcement

A call remains queued until an agent becomes available to answer the call. In the meantime, it is likely that the caller wants to hear some feedback assuring him or her that the call is being processed.

The vector shown in the following example provides one feedback solution. In this example, Announcement 2771 could contain this message: *We're sorry. All of 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 8-hour delay before the next vector step is processed. The time parameter can be assigned as follows:

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

In the example vector, the specified wait time is 10 seconds.

In addition to the delay period, the **wait-time** command provides the caller with feedback. In our vector, **ringback** is provided. Other types of feedback that can be provided with the **wait-time** command are: silence, system music, or an alternate music or other audio source. For more information see, wait-time command on page 645.

The **wait-time** command in the example vector provides the caller with a maximum of 10 seconds of ringback. If an agent answers the call before the **wait-time** command runs its course, the command is terminated, the delay period is ended and the accompanying feedback is stopped. In the current example, if the call is delivered to an agent after 4 seconds the caller does not hear the remaining 6 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, and it is often used to encourage the caller to stay on the telephone or to provide information to the caller. If a 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. See *Feature Description and Implementation for Avaya Communication Manager*, for more information about announcements.

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 is complete, further feedback or treatment becomes 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 of this vector provides additional feedback (music) to the caller. If the call is not answered by the time step 4 is complete, the **goto step** command in step 5 is processed.

Conditional branching

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

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

Step 4: Queuing a call to a backup split

To this point, the vector example involves a call queued to one split. However, Call Vectoring allows a call to be queued to a maximum of three splits simultaneously, which improves can improve overall call response times. Multiple split queuing is especially useful during periods of heavy call traffic.

The vector shown in the following example allows a call to be queued to two splits.

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. But if the call is not answered by the time the **wait-time** command in step 4 is complete, the **check split** command in step 5 attempts to queue the call to backup Split 7 at a medium priority. The condition expressed in the command (**if calls-queued < 5**) determines whether or not the call is to be queued to the backup split. Specifically, if the number of calls currently queued to Split 7 at a medium or higher priority is less than 5, the call is queued to the split.

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. The other conditions are **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 four priorities: **low**, **medium**, **high**, or **top**.

Elevating call priority

Note that if the call is queued to Split 7, the call priority is elevated from low to medium priority instead of a low priority, which is assigned if the call is queued by the **queue-to split** command in step 1. It is a good practice to raise the priority level in subsequent queuing steps to accommodate callers who have been holding the line for a period of time.

Step 5: Limiting the queue capacity

Starting with Communication Manager 2.1, hunt group queue slots are dynamically allocated by the system. For more information about dynamic queue slot allocation, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* . Therefore, there is no need to include vector steps to insure 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 checking or limiting the number of calls that queue to a split/skill or 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. In the example shown above, assume that only 21 queue slots are used by split 5. Accordingly, the **goto step** command tests whether 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, shown in vector step 10. The **busy** command gives the caller a busy signal 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

Instead of providing the caller with a busy tone if the **queue-to split** step cannot queue the call, the call can be queued to a backup split. 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 (step 6). Because this queuing step is included within a continuous loop of steps (steps 6 through 9), continuous attempts to queue the call are now made.

Step 6: Checking for non business hours

If a caller calls during non business hours, you can still provide the caller with some information for calling back during working hours by playing the appropriate recorded message. This strategy is illustrated in the following Call Vector example. This vector would be used for a company that was 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. Specifically, if the call arrives between 5:00 p.m. and 8:00 a.m. on any day of the week, the command passes control to step 12.

The **disconnect** command in step 12 includes and provides an announcement that first gives the caller the appropriate information and then advises him or her to call back at the appropriate time. The command then disconnects the caller.

If the call does not arrive during the specified 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. See Basic Call Vectoring on page 103 for more details.

Duplicating Vectors

You can use the Duplicate Vector command to create duplicate vectors from an existing vector and then 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.

This section includes the following topics:

- Duplicate Vector command on page 234
- Duplicate Vector screen field descriptions on page 235
- Creating duplicate vectors on page 236

Duplicate Vector command

From the System Administration Terminal (SAT), use the following command to access the Duplicate Vector screen.

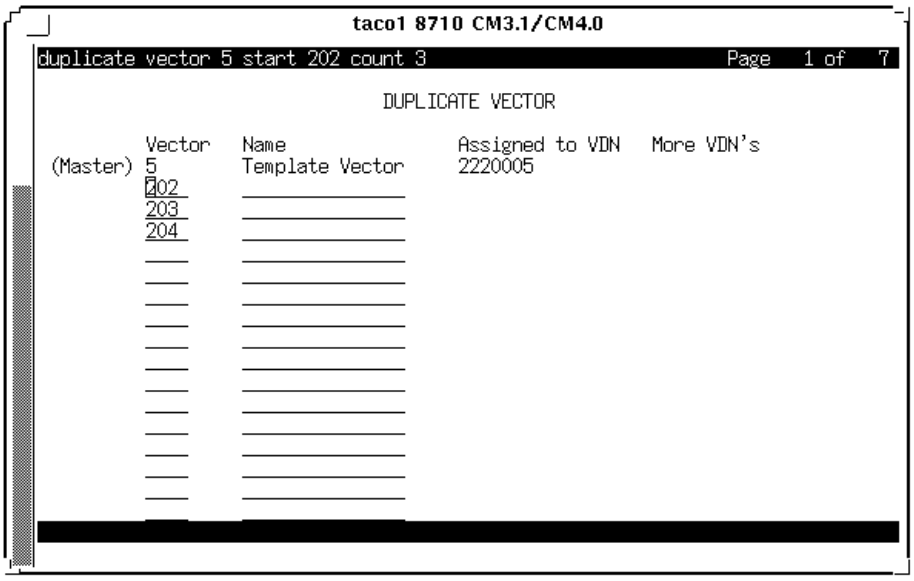
duplicate vector *master_vector* [**start *nnnn***] [**count *xx***]

Parameter	Description
duplicate vector	Creates a duplicate vector, up to 16 vectors.
master_vector	Specifies the vector number of the vector you want to duplicate or use as a template.
[**start *nnnn***]	Specifies the first vector number you want used as a duplicate. This parameter is optional. If you do not specify a start number, the software automatically selects the first available vector after the master vector number. Only one vector is selected.
[**count *xx***]	Specifies the number of duplicates you want to create from the master vector. You can enter a number from 1 to 16. This parameter is optional. If you do not specify a count number, the software automatically selects the first available vector after the master vector. Only one vector is selected.

Example

The following example creates vectors 202, 203, and 204 as exact duplicates of vector 5.

```
duplicate vector 5 start 202 count 3
```



Duplicate Vector screen field descriptions

The following fields are populated in the Duplicate Vector screen:

Count: Displays the number of duplicates created from the master vector.

More? Displays * if there is at least one more VDN assigned to the same vector. For example, if 5555 displays in the **VDN Assigned to** field and an asterick (*) displays in the **More?** field, this means that the master vector you selected is already assigned to VDN 5555 as well as to other VDNs.

Name: Displays the vector name if any of the vectors have an assigned name. The duplicated vectors can already be assigned names but they must be vectors that contain no steps. You can edit the vector name for any of the duplicated vectors.

If you specify a used or out of range vector number, an error message is displayed. You cannot move to the next field until you enter an unused number.

VDN Assigned to: Displays the VDN if a VDN was assigned to the master vector.

Vector: Displays the vector number.

Creating duplicate vectors

To administer duplicate vectors:

1. From the System Administration Terminal (SAT), enter the following command :

   duplicate vector *master_vector* [**start *nnnn***] [**count *xx***]

2. In the Duplicate Vector screen, add the new names to the duplicated vectors.
3. Edit each of the duplicate vectors to make the required changes for the different applications.

The reporting adjunct can access these vectors as normal.

Call Prompting

This section includes the following topics:

- About Call Prompting on page 237
- Command set on page 238
- Touch-tone collection requirements on page 238
- Call Prompting digit entry - collect digits command on page 239
- Functions and examples on page 241
- Dial-ahead digits - collect digits command on page 249
- ASAI-requested digit collection on page 253
- ASAI-provided dial-ahead digits - collect digits command on page 254
- Considerations on page 254

About Call Prompting

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

With Call Prompting and Vectoring enabled, the switch can collect caller entered digits (ced) and customer database provided digits (cdpd) that are supplied by the network. The system can receive Call Information Forwarding (CINFO) digits in an incoming call's ISDN message when the AT&T Network Intelligent Call Processing (ICP) service is in use. A switch can collect digits and forward those digits to other switches by way of interflow commands. For more information, see Caller Information Forwarding on page 180.

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

Call Prompting can be used in various applications so that calls can be handled with more flexibility.

Command set

The following table show the commands that are used for Call Prompting.

Call Prompting command set

Command category	Action taken	Command
Information collection	Collect information from the calling party, from the public network in an ISDN SETUP message, from a Voice Response Unit (VRU), or from CallVisor Adjunct Switch Application Interface (ASAI).	**collect digits**
Treatment	Play an announcement. Delay with audible feedback of silence, ringback, system music, or an alternate audio/music source.	**announcement** **wait-time**
Routing	Leave a message. Route the call to a number that is programmed in the vector. Route the call to digits that are supplied by the calling party.	**messaging split** **route-to number** **route-to digits**
Branching/ programming	Go to a vector step. Go to another vector. Stop vector processing.	**goto step** **goto vector** **stop**

Touch-tone collection requirements

Before the switch can accept the touch-tone digits that are entered by a caller, the switch must be equipped with a collection resource. The resource used for collecting and interpreting touch-tone digits is a unit of hardware called a Touch-Tone Receiver (TTR). These TTRs are provided on the call classifier and tone detector circuit packs, one of which is required for Call Prompting.

The number of TTRs that are required is configured according to two sources:

- Customer input to the Avaya Account Team
- Account team input to the configurator tool

For existing systems that are adding a Call Prompting application, the Account Team recommends the appropriate number of TTRs based on two factors:

- Account team input to the configurator tool
- Application review by the Avaya Design Center

The process of collecting CINFO digits does not require TTRs.

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

Note:
The Call Prompting interdigit timeout can be administered for any number of seconds from 4 to 10. This value is administered on the Feature-Related System Parameters screen.

Provisions for users of rotary telephones are illustrated in the vector scripts in this section.

Call Prompting digit entry - collect digits command

This section includes the following topics:

- About the collect digits command on page 239
- Removing incorrect digit strings on page 240
- Entering variable-length digit strings on page 240
- Entering dial-ahead digits on page 241

About the collect digits command

The touch-tone digits that are entered by a Call Prompting user are collected by the **collect digits** command. This command allows the system to collect up to 24 digits from a touch-tone telephone. Sixteen of these digits may be collected immediately, while any remaining digits are stored as dial-ahead digits, which are explained later.

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

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

The following sections explain these processes.

Removing incorrect digit strings

An announcement that requests the caller to enter digits can be included in call treatment. As an option, the announcement can instruct the caller to enter an asterisk (*) if he or she enters incorrect data.

When the caller enters a *, the following happens:

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

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

2. Digit collection is restarted.
3. The announcement is not replayed.

Once the caller enters an asterisk, the caller can reenter digits for processing.

Entering variable-length digit strings

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

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

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

Whenever the caller is permitted to enter a variable-length digit string, the announcement portion of the **collect digits** command should specify the largest possible number of digits that can be entered. Accordingly, each **collect digits** command should be administered to collect no more than the intended maximum number of digits. The caller can enter a pound sign part of a variable digit string entry either:

- At the end of each variable digit string that is entered. In this case, the pound sign should be included in the count of the number of maximum digits that can be entered.
- At the end of each such string that, not counting the pound sign, contains fewer characters than the maximum number of allowable digits. In this case, the pound sign should not be included in the count of the number of maximum digits that can be entered.

If the caller enters more digits than the maximum number specified, the additional digits are saved as dial-ahead digits for subsequent **collect digits** commands. If the vector or vectors chained to it do not contain another **collect digits** command, the extra digits are discarded.

If the caller enters fewer digits than the maximum number specified and does not complete the entry with the pound sign, a Call Prompting timeout occurs. The timeout terminates the command, and any digits collected prior to the timeout are available for subsequent vector processing. The Call Prompting timeout period is set to 10 seconds by default but can be changed to a value between 4 and 10 seconds using the Prompting Timeout field on the Feature-Related Customer-Options screen. See collect digits command on page 556 for detailed information.

A common application involving the entering of variable-length digit strings allows the user to dial either the number for the attendant or an extension to reach the desired destination. If the maximum number of digits that can be entered is administered to be 3 and the user wishes to reach the attendant, the user should dial 0#. However, if the user chooses to dial a 3-digit extension, the user should dial, for example, 748 and not 748#. Since the maximum number of digits that can be dialed in this case is three, dialing 748# would cause # to be saved as a dial-ahead digit. On the other hand, if the caller dials 748#, and if the maximum number of digits that can be entered is 4, # is not saved as a dial-ahead digit since it is the fourth of four digits that can be entered in this case.

Entering dial-ahead digits

When digit collection for the current **collect digits** command is completed, vector processing continues at the next vector step. However, the switch continues to collect any digits that the caller subsequently dials until the TTR disconnects. For more information, see Collecting Digits on the switch on page 557. These *dialed-ahead* digits are saved for processing by subsequent **collect digits** commands. Dial-Ahead Digits are explained fully in Dial-ahead digits - collect digits command.

Functions and examples

Call Prompting uses some of the functions found in Basic Call Vectoring. Call Prompting also provides some additional functions that involve digit processing. These functions are included in the following sections:

- Treating digits as a destination on page 242
- Using digits to collect branching information on page 243
- Using digits to select options on page 245
- Displaying digits on an agent set on page 245

- Passing digits to an adjunct on page 247
- Creating Service Observing vectors on page 248

Treating digits as a destination

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

The digits can represent any of the following destinations:

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

The following example shows how a call is routed by digits that are collected from a caller.

Using Call Prompting to route by collected digits

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

In this vector, the caller is prompted to enter the destination extension of the party that he or she would like to reach (step 2). The extension in this vector may contain up to 5 digits. The vector collects the digits and then routes to the destination by the **route-to digits** command in step 3.

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

Note:

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

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

Using digits to collect branching information

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

Using Call Prompting to branch by collected digits

```
1. wait-time 0 seconds hearing ringback
2. collect 5 digits after announcement 200
   [Please enter your customer number.]
3. goto vector 8 if digits = 10+
4. goto vector 9 if digits = 11+
5. goto vector 10 if digits = 12+
6. route-to number 0 with cov n if unconditionally
7. stop
```

The wildcard + indicates that the two digits can be followed by zero or any number of additional digits. Callers with a number that begins with the digits 10 are routed to vector 8, callers with a number that begins with the digits 11 are routed to vector 9, and callers with a number that begins with the digits 12 are routed to vector 10.

Vector Routing Tables

You also can test digits against entries in a Vector Routing Table.

Vector Routing Tables contain lists of numbers that can be used to test a **goto...if digits** command. Digits that are collected with the collect digits step can be tested to see if they are either in or not-in the specified table. Entries in the tables can include either the + or ? wildcard.

- The + represents a group of digits and can only be used as the first or last character of the string.
- The ? represents a single digit. Any number of them can be used at any position in the digit string.

Tables are entered on the Vector Routing Table screen. For complete instructions for creating Vector Routing Tables, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

The following Call Vector example could be used to test against the numbers provided in the Vector Routing Table.

Testing for digits in Vector Routing Table

```
1.wait-time 0 seconds hearing ringback
2.collect 7 digits after announcement 200 [Please enter your account      number.]
3.goto vector 8 if digits in table 10
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
```

If the caller enters an account number that is listed in the Vector Routing Table, the call is routed to vector 8. If the caller enters an account number that matches the wildcard entry (for example 1345987), the call is routed to vector 8.

If the caller enters an account number that is not listed in the Vector Routing Table, or if the caller does not enter an account number, the call is queued to split 5.

Suppose that, instead of containing a list of premier accounts, the Vector Routing Table contains a list of accounts with a poor payment record. The following example shows a vector that only queues calls with account numbers that are not in the table. Calls in the table route to the collection department.

Testing for digits not in Vector Routing Table

```
 1.wait-time 0 seconds hearing ringback
 2.collect 7 digits after announcement 200 [Please enter your account      number.]
 3.goto step 11 if digits = none
 4.goto step 6 if digits not-in table 10
 5.route-to number 83456 with cov y if unconditionally      [collections]
 6.queue-to split 5 pri l
 7.wait-time 10 seconds hearing ringback
 8.announcement 2771
 9.wait-time 10 seconds hearing music
10.goto step 8 if unconditionally
11.route-to number 0 with cov n if unconditionally
12.stop
```

If no digits are collected, the call is routed to the operator.

Note:

Entries in Vector Routing Tables also can be tested against the telephone number of the caller Automatic Number Identification (ANI). For more information, see ANI /II-digits routing and Caller Information Forwarding (CINFO) on page 169.

Using digits to select options

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

The following example shows how digits are used to select options.

Using Call Prompting to select options

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

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

Displaying digits on an agent set

A CALLR-INFO button can be included at the agents' display stations to help process calls that are serviced by the Call Prompting feature. However, if the agent has a two-line display set, and the display is in normal or inspect mode, the collected digits are automatically displayed on the second line. As a default, these digits remain on this line until they are overwritten, even after the call is released by the agent. As an option, administrators can decide when an agent's station display is cleared of caller information. For more information, see Clearing caller information from the station display on page 246. For other display sets, the agent must press the CALLR-INFO button to display the collected digits.

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

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

x = Info: 1234567890

where:

- *x* is a call appearance letter, for example, a, b, c, and so forth
- *1234567890* represents the digits that are collected from the caller

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

Assume that digits have been collected by Call Prompting. If the agent presses the CALLR-INFO button when the call rings at the agent station or when the station is active on a call appearance, the following events occur:

- The 10-second timer for display interval is set.
- The status lamp (if available) that is associated with the button is lit.
- The display is updated. Specifically, the incoming call identification (calling party ICI) is replaced with the collected digits in the format that was presented earlier in this section. Only those digits that were collected for the last **collect digits** command are displayed.

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

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

- The 10-second timer times out.
- The incoming call arrives at any call appearance.
- An active call changes status, for example, another caller is added to the conference.

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

- The status lamp (if available) that is associated with the button is turned off.
- The display is updated as previously described.

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

Clearing caller information from the station display

Administrators can decide when an agent's station display is cleared of caller information. Options include:

- Clearing the existing call information when the next call is received

- Clearing the existing call information when the call is released - whether the agent enters After Call Work (ACW) or not
- Clearing the existing call information when the agent leaves ACW mode or if the agent does not enter ACW, when the call is released

For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Passing digits to an adjunct

Call Prompting allows for the passing of information in the form of collected digits to an adjunct for further processing. Digits are passed to the adjunct by the ASAI Adjunct Routing capability.

An adjunct is any processor that is connected to a switch by the ASAI link. The adjunct makes a routing decision using the **adjunct routing link** command according to caller information and/or agent availability, and it returns the routing response to the switch. For example, the adjunct can indicate that the call be routed to a specific ACD agent. This is known as Direct Agent Calling (DAC).

A maximum of 16 Call Prompting digits from the last **collect digits** command can be passed to the adjunct by the **adjunct routing link** command.

The following example, shows how Call Prompting digits are passed to an adjunct.

Using Call Prompting to pass digits to an adjunct

```
1.wait-time 0 seconds hearing ringback
2.collect 10 digits after announcement 300 [Please enter your 10-digit      account
number.]
3.adjunct routing link 15
4.wait-time 10 seconds hearing music
5.route-to number 52000 with cov y if unconditionally
6.stop
```

In step 2 of this vector, the caller is asked to enter a 10-digit account number. Once the account number is entered, the adjunct receives this information from the **adjunct routing link** command in step 3. This command then makes the appropriate routing decision if it is able to do so. If the command succeeds within the specified wait time, the command routes the call to the appropriate destination, and the call leaves vector processing. If the command fails, vector processing continues at the next step.

In addition to the Adjunct Routing capability, collected digits also can be passed by way of ASAI to an adjunct by prompting for the digits in one vector and then routing the call to a VDN that is monitored by an Event Notification (VDN) association. The collected digits (up to 16) are sent to the adjunct in a Call Offered to Domain Event Report. For detailed information, see *Avaya Communication Manager CallVisor ASAI Technical Reference*.

Note:
Adjunct Routing is fully discussed in Adjunct (ASAI) Routing on page 201.

Creating Service Observing vectors

Service Observing vectors can be constructed to allow users to observe calls from a remote location or local station. When combined with Call Prompting, Service Observing vectors can route calls to a:

- Remote access Service Observing vector on page 248
- User-entered FAC and extension on page 248
- Pre-programmed FAC and extension on page 249

Remote access Service Observing vector

The following vector example connects a user to Remote Access. Once connected, the user can dial either a listen-only or listen/talk Service Observing FAC followed by the extension number to be observed. Although it is not required, Call Prompting increases security by providing passcode protection with remote service observing.

Remote access Service Observing vector

```
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 2300 [Please enter your 5-digit      security
code.]
3. goto step 5 if digits = 12345 (security code)
4. disconnect after announcement 2000
5. route-to number 5000 with cov n if unconditionally
6. stop
```

User-entered FAC and extension

The following vector example connects a user directly to the Service Observing FAC and extension based on the digits that are collected by Call Prompting.

Service Observing vector with user-entered FAC and extension

```
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 2300 [Please enter your 5-digit      security
code.]
3. goto step 5 if digits = 12345 [security code]
4. disconnect after announcement 2000
5. wait-time 0 seconds hearing ringback
6. collect 6 digits after announcement 3245 [Please enter the number 11 for
     listen-only observing or the number 12 for listen/talk observing      followed by
the number of the extension you would like to observe.]
7. route-to digits with coverage n
8. stop
```

Pre-programmed FAC and extension

The following example shows a vector that connects a user to a pre-programmed FAC and extension using Call Prompting to allow the observer to select the extension that he or she wants to observe. In this example, the observer will be Service Observing a VDN.

Service Observing vector with programmed FAC and extension

```
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 2300 [Please enter your 5-digit      security
code.]
3. goto step 5 if digits = 12345 [security code]
4. disconnect after announcement 2000
5. wait-time 0 seconds hearing ringback
6. collect 1 digits after announcement 2310 [Enter 1 to observe sales, 2 to
     observe billing.]
7. route-to number 113001 with cov n if digit = 1 [11 = listen-only observe,      3001
= Sales VDN]
8. route-to number 113002 with cov n if digit = 2 [11 = listen-only observe,      3002
= Billing VDN]
9. goto step 6 if unconditionally
```

Dial-ahead digits - collect digits command

This section includes the following topics:

- About dial-ahead digits on page 249
- Dial-ahead digit vector examples on page 250

About dial-ahead digits

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

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

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

  Note:
  Any asterisk (*) and pound sign (#) digits that are dialed ahead count toward the 24 digit limit, as do any dial-ahead digits that are entered after the asterisk or pound sign digit.

- The TTR required by the user to collect digits is disconnected. This happens whenever one of the following conditions is true:
  - A successful or unsuccessful **route-to number** step is encountered during vector processing, except where the number routed to is a VDN extension.
  - A successful or unsuccessful **route-to digits** step is encountered during vector processing, except where the number routed to is a VDN extension.
  - A successful or unsuccessful **adjunct routing link** step is encountered during vector processing.
  - A successful or unsuccessful **converse-on** step is encountered during vector processing.
  - A Call Prompting timeout occurs, during which time the caller has not dialed any additional digits, asterisks (*) or pound signs (#).
  - Vector processing stops or is terminated.
  - A successful or unsuccessful collect ced/cdpd step is encountered.

  Note:
  When the TTR is disconnected due to a **route-to number**, **route-to digits**, **converse-on**, **adjunct routing link** , or **collect ced/cdpd** step, all dial-ahead digits are discarded. This means that following a failed **route-to**, **converse**, or **adjunct routing link** step, a subsequent **collect digits** step always requires the user to enter digits.

Dial-ahead digit vector examples

The vectors shown in the following examples illustrate a situation where a caller can enter dial-ahead digits. In this case, the caller is required to have a touch-tone telephone. An alternative handling sequence should be programmed in case the caller has a rotary telephone or the caller does not dial a touch tone digit before the timeout period.

Step 2 of Vector 30 gives the caller two options, each of which provides different information. The caller is prompted to enter either 1 or 2, depending on what information he or she wants to hear. Once the caller enters a digit, the digit is collected by the **collect digits** command. Thereafter, an attempt is made by the **route-to number** command to route the call to the appropriate vector (step 3 or 4). If the caller enters a digit other than 1 or 2, the appropriate announcement is provided (step 5), and the digit entry cycle is repeated (step 6).

If the caller enters 1, Vector 31 is accessed.

Using dial-ahead digits to bypass announcements, example 1

```
VDN (extension=1030   name=''Coastal''   vector=30)
Vector 30:
1.wait-time 0 seconds hearing ringback
2. collect 1 digits after announcement 3000 [Thank you for calling Coastal    League
Baseball Hotline. You must have a touch-tone telephone to use this    service. If you
wish to hear the scores of yesterday's games, please press 1.  If you wish to hear
today's schedule of games, please press 2.]
3. route-to number 1031 with cov y if digit = 1
4. route to number 1032 with cov y if digit = 2
5. announcement 301 [Entry not understood. Please try again.]
6. goto step 2 if unconditionally
```

In step 1 of Vector 31 (below), the caller is given three options that supplement the original option that was provided in Vector 30. The caller is prompted to enter either 3, 4, or 5, depending on what information he or she wants to hear. If the caller enters an incorrect digit, the customary digit correction routine is implemented (steps 5 and 6). Once an appropriate digit is entered, the call is routed, in this example by a **goto step** command (step 2, 3, or 4), to the appropriate announcement (step 7 or step 9).

In step 10 of Vector 31, the caller is prompted with the choice of returning to the main menu provided in Vector 30 or of terminating the call. If the caller selects the former option (by entering 9), the call is routed to Vector 30, and the entire process is repeated.

Using dial-ahead digits to bypass announcements, example 2

```
VDN (extension=1031 name=''Scores''   vector=31)
Vector 31:
 1.collect 1 digits after announcement 4000 [If you wish to hear scores of      games
in both divisions, please press 3.  If you wish to hear scores for      Northern
Division games only, please press 4. If you wish to hear scores for      Southern
Division games only, please press 5.]
 2. goto step 7 if digits = 3
 3. goto step 7 if digits = 4
 4. goto step 9 if digits = 5
 5. announcement 301 [Entry not understood. Please try again.]
 6. goto step 1 if unconditionally
 7. announcement 4002 [Northern Division scores]
 8. goto step 10 if digits = 4
 9. announcement 4003 [Southern Division scores]
10. collect 1 digits after announcement 4004 [If you wish to return to the main
    menu, please press 9. Otherwise, press 0.]
11. route-to number 1030 with cov n if digit = 9
12. goto step 15 if digit = 0
13. announcement 301 [Entry not understood. Please try again.]
14. goto step 10 if unconditionally
15. disconnect after announcement none
```

Vector 32 (below) is similar in design to Vector 31. The major difference is the information provided and the requested digit entries.

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

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

Dial-ahead digits, example 2

```
VDN (extension=1032   name=Schedule   vector=32)
Vector 32
 1.collect 1 digits after announcement 5000 [If you wish to hear today's      schedule
of games in both divisions, please press 6.  If you wish to hear      today's schedule
of games in the Northern Division only, please press 7.      If you wish to hear
today's schedule of games in the Southern Division      only, please press 8.]
 2.goto step 7 if digits = 6
 3.goto step 7 if digits = 7
 4.goto step 9 if digits = 8
 5.announcement 301 [Entry not understood. Please try again.]
 6.goto step 1 if unconditionally
 7.announcement 5002 [Northern Division schedule]
 8.goto step 10 if digits = 7
 9.announcement 5003 [Southern Division schedule]
10.collect 1 digits after announcement 4004 [If you wish to return to the      main
menu, please press 9.  Otherwise, press 0.]
11.route-to number 1030 with cov n if digit = 9
12.goto step 15 if digits = 0
13.announcement 301 [Entry not understood. Please try again.]
14.goto step 10 if unconditionally
15.disconnect after announcement none
```

ASAI-requested digit collection

The ASAI-requested digit collection feature gives an adjunct the ability to request that a DTMF tone detector be connected for the purpose of detecting user-entered digits. The digits that are collected as a result of this feature are passed to ASAI monitoring and/or controlling adjuncts for action. The switch handles these digits as if they were dial-ahead digits. This feature allows the caller to request Sequence Dialing after the call has been routed to the final destination and has resulted in an unanswered call, that is busy, no answer, and so forth.

These digits are not necessarily collected while the call is in vector processing. They are sent to an ASAI adjunct, or they may be used by Call Prompting features, or both.

ASAI Adjunct Routing and Call Prompting features must be enabled on the switch for this feature to work.

ASAI-provided dial-ahead digits - collect digits command

The ASAI-provided digits feature allows an adjunct to include digits in a Route Select capability. These digits are treated as dial-ahead digits for the call. Dial-ahead digits are stored in a dial-ahead digit buffer and can be collected (one at a time or in groups) using the **collect digits** command(s). Although the adjunct may send more than 24 digits in a Route Select, only the first 24 digits (or 24-*x*, where *x* is the number of digits that are collected by vector processing prior to executing the **adjunct routing link** vector command) are retained as dial-ahead digits. An application can use this capability to specify the digits that the switch should pass to the VRU as part of the **converse-on** vector step.

Note:

The maximum number of dial-ahead digits that can be stored in the buffer is dependent on the number of digits that were already collected for the call by a previous **collect digits** vector command. If *x* digits were collected by vector processing prior to executing an **adjunct routing link** vector command, the *x* digits collected reduces the maximum number of digits that can be stored as dial-ahead digits as a result of a Route Select. The rest are discarded.

Considerations

You should keep the following considerations in mind when working with Call Prompting:

- To enter the digits requested using a **collect digits** command, outside callers must have a touch-tone telephone. For such callers using rotary dialing, a 10 second inter-digit timeout takes effect, and the **collect digits** command is omitted. As a precaution, a default treatment (for example, **route-to attendant** command, **queue-to split** command) should always be provided in the vector script unless the script is created exclusively for users of touch-tone telephones.
- If a caller does not enter the full number of digits specified in a **collect digits** step, an administered timeout occurs. Thereafter, vector processing continues with subsequent vector steps, and an attempt is made to process the call using the digits that have been collected. If the digits entered do not represent a valid destination, and if Automated Attendant is being implemented using a **route-to digits** command, the **route-to digits** command fails, and vector processing continues at the next step, which should be a default treatment.
- It may be prudent to take steps in case a **route-to attendant** command fails, such as providing a disconnect announcement.

- From time to time, all of the system's touch-tone receivers might be in use. As a result, you should avoid starting your main vector with a **collect digits** command, since the caller on a DID or tie trunk in this case receives no audible feedback if he or she has to wait for a receiver to become available. Accordingly, it is a good practice to include some treatment (for example, a **wait-time 0 seconds hearing ringback** step) before the initial **collect digits** step. The **wait-time** step is not necessary if the collect step is collecting ced or cdpd digits.

Advanced Vector Routing - EWT and ASA

This section includes the following topics:

- About Advanced Vector Routing features on page 257
- Advanced Vector Routing command set on page 258
- When to use wait time predictions on page 258
- Expected Wait Time (EWT) on page 259
- Rolling Average Speed of Answer (ASA) on page 266
- VDN Calls on page 269

About Advanced Vector Routing features

Several Advanced Vector Routing features can be used to enhance conditional routing capabilities of Basic Call Vectoring in order to achieve additional efficiencies in call center operations. These features include:

Rolling Average Speed of Answer (ASA): Rolling ASA Routing allows routing decisions to be based on the current average time for a call to be answered in a split or VDN, so that vectors route calls to the VDN or split where it is likely to be answered most quickly.

Expected Wait Time (EWT): EWT routing allows you to make routing decisions based on the wait time in queue for a call or split. The EWT can also be passed to a VRU so that a caller can be notified of his or her expected time in queue.

VDN Calls: VDN Calls routing helps you to make routing decisions that are based on the number of incoming trunk calls that are currently active in a VDN. With the VDN Calls conditional, a vector can be used to limit the number of simultaneous calls that are made to a particular VDN. For example, if a service agency is contracted to handle 100 simultaneous calls for a client, calls in excess of that number can be routed to a **busy** step.

Advanced Vector Routing command set

The commands used in Advanced Vector Routing are listed in the following table.

Command category	Action taken	Command
Routing		
	Queue the call to a backup ACD split.	`check split`
Branching/programming		
	Go to a vector step. Go to another vector.	`goto step` `goto vector`

When to use wait time predictions

A number of factors can affect the accuracy of wait time predictions. Wait time predictions are best suited for medium-volume or high-volume call scenarios. The potential accuracy of a wait time predictor increases as the rate of removal from queue increases.

Under all conditions, EWT is the most accurate wait time predictor, but EWT is most accurate when the rate of removal from queue at a given split priority level is at least one call every 30 seconds. For more information, see Expected Wait Time (EWT) on page 259.

Predictions can be made for a split with multiple priority levels as long as the majority of calls are delivered to lower priority levels. If the majority of calls are queued at the higher-priority levels, any predictions made for the lower-priority levels may not be accurate.

The following circumstances can limit the accuracy of the wait time predictions.

System restart or new split administration: The EWT algorithm uses a combination of historical and real-time information to make predictions. When no historical information exists, such as when a new split is added or a reset system 3 or 4 is completed, there is the potential for inaccuracies.

To prevent inaccurate predictions when there is no historical information, administer the **Expected Call Handling Time** field on the Hunt Group screen. The value in this field is then used in place of the missing historical data.

If the value of this field does not accurately reflect the call handling times of the split, EWT predictions may be inaccurate until some call history is generated. The algorithm normally requires about 30 queued calls to be answered from a split priority level before it reaches its maximum accuracy.

You can change the value in the **Expected Call Handling Time** field by executing a change hunt group command. Changing the value does not disrupt EWT predictions by overwriting EWT history. The value is stored and used the next time a reset system 3 or 4 is executed.

Low call volume applications: Split priority levels where the rate of removal from the queue is very low can only be predicted with limited accuracy.

Sites with frequent staffing changes: Although EWT immediately adjusts for all types of staffing changes, since predictions may have already been made for calls that are waiting in queue, those past predictions were based on staffing information which is now out of date. Therefore, the EWT in scenarios where large staffing changes are continually happening can only be predicted with limited accuracy.

Staffed agents who rarely answer calls to a split: The EWT algorithm takes account of agents in multiple splits in its calculation. However, suppose there are many agents who are assigned to a split but spend most of their time answering calls in their other splits. If a large number of these agents are moved to or from the split, the EWT for this split may be temporarily inaccurate until it adjusts to those changes.

Applications with widely varying call handling times: If the majority of calls to a split are handled within a narrow range of times, the accuracy of any predictor will be much greater than that for a split where call handling times are widely different.

Expected Wait Time (EWT)

EWT routing allows you to make routing decisions based on the time that a caller can expect to wait in queue.

This section includes the following topics:

- How EWT is calculated on page 260
- EWT for a split on page 260
- EWT for a call on page 261
- Passing EWT to a VRU on page 262
- Notifying callers of wait time without a VRU on page 263
- Using EWT to route to the best split on page 264
- Factors that affect EWT values on page 265
- Troubleshooting EWT on page 266

How EWT is calculated

Depending on how the EWT condition is used in a vector step, the predicted wait time calculation is derived by the following rules:

- If the call is currently queued to a split, the EWT is based on the actual current position of the call in the queue at a particular priority level and the rate of service of calls from the queue at that priority level.
- If the call is not yet queued to a split, the EWT is based on the assumption that the call is placed at the end of the queue and then considers the factors listed above.

EWT also adjusts for many other factors such as multiple split queuing, call handling times, and the impact of direct agent calls on the wait time of other calls to the split. The algorithm adjusts EWT immediately for changes in staffing, such as agents logging in or taking breaks in AUX work mode.

The EWT can also be passed to a VRU so that a caller can be notified of his or her expected time in queue. The **expected-wait** condition can be used with either the **goto** or **check** commands.

Call vectoring offers several conditionals that can be used to estimate predicted wait time on a queue, including EWT, rolling ASA and Oldest Call Waiting (OCW), but EWT uses the most accurate method of prediction. EWT considers more real-time and historical information, such as priority level, position in queue, and number of working agents.

EWT is responsive to changing call center conditions. For example, EWT adjusts instantly to any staffing changes in the split, or if agents moves in or out of auxiliary work mode, the wait time predictions immediately adjust.

EWT does not include the time in a call vector before the call enters a queue. It also does not include the time that the call rings at a telephone after it is removed from the queue.

For more information about the use and accuracy of wait time predictors, see When to use wait time predictions on page 258.

EWT for a split

The EWT for a split is the time that an incoming call is expected to remain in queue if it is queued to the split at the specified priority level. It is generally used to determine if a call should be queued to the split.

The following vector example shows how to use EWT to determine if a call should be queued to a split.

```
1. goto step 3 if expected-wait for split 1 pri l < 600
2. busy
3. queue-to split 1 pri l
4. announcement 3001
5. wait-time 998 secs hearing music
```

In the example shown above, the following wait time conditions are possible:

- If there are agents available, EWT is zero.
- EWT is infinite if:
  - There are no logged-in agents.
  - All logged-in agents are in AUX work mode.
  - The split queue is full.
  - There is no split queue and all agents are busy.
  - The split queue is locked. This occurs when the last working agent in a non-vector-controlled split attempts to go into AUX work mode.

EWT for a call

EWT for a call is the remaining time that a caller can expect to wait before his or her call is serviced from the queue. If the call is queued to multiple splits, the remaining queue time for each of the splits is calculated, and the shortest calculation is used as the EWT.

For a call to have an expected wait time it must be queued to at least one split. If it is not queued, or if it is queued to splits that are not staffed, the EWT value is infinite.

The following vector example vector shows how EWT is used to determine call treatment.

```
1. queue-to split 1 pri m
2. check split 2 pri m if expected-wait < 30
3. goto step 5 if expected-wait for call < 9999
4. busy
5. announcement 3001
6. wait-time 998 secs hearing music
```

Passing EWT to a VRU

The EWT for a call can be passed to a VRU to inform callers about their expected time in queue. EWT is passed to the VRU with the **converse-on** command as wait data. The value that is outpulsed to the VRU is the expected wait time of the call in seconds. The VRU can then convert the seconds to a spoken message. The expected wait is calculated after the VRU port answers the call, so queuing to a converse split does not adversely impact the EWT value that is passed to the VRU.

No zero padding is added to the wait time that is passed to the VRU. If the EWT for the call is 128 seconds, the digits 1, 2, and 8 are outpulsed. If the EWT is 5 seconds, the digit 5 is outpulsed.

The wait time that is passed to the VRU is the most accurate prediction possible. On average, 50% of the time the actual wait time will be shorter and 50% of the time it will be longer. Avaya recommends that VRU applications be configured to make an upward adjustment of the prediction so that the majority of callers receive a predicted wait time that is either equal to, or greater than, the actual wait time.

The VRU can also announce EWT at set intervals while the call is in queue, but this strategy should be used with caution. Circumstances such as a reduction in the number of agents or a sudden influx of higher priority calls could cause the caller's EWT to increase from one announcement to the next.

If the call is not queued, or if it is queued only to splits that are unstaffed or splits where all agents are in AUX work mode, the end-of-string character # is the only data item that is outpulsed to the VRU.

The following vector example illustrates routing based on the predicted split wait time and passing wait data to the VRU. Wait time is given to the caller only if the caller is expected to wait a total of more than 60 seconds in queue. Callers who would wait more than 10 minutes are told to call back later.

```
1. goto step 3 if expected-wait for split 32 pri l < 600
2. disconnect after announcement 13976
3. queue-to split 32 pri l
4. wait-time 20 secs hearing ringback
5. goto step 7 if expected-wait for call < 40
6. converse-on split 80 pri l passing wait and none
7. announcement 11000
8. wait-time 60 secs hearing music
9. goto step 7 if unconditionally
```

Calls that have predicted wait times greater than 10 minutes fail step 1 and are disconnected after an announcement. If the expected wait time is less than 10 minutes step 1 routes the call to step 3 where it is queued to split 32 and waits 20 seconds hearing ringback. After 20 seconds if the expected wait time for the call is less than 40 seconds, step 5 routes the call to an announcement followed by a wait with music. If the expected wait time for the call is equal to or greater than 40 seconds, step 6 informs the caller of the amount of time that he or she can expect to wait before the call is answered.

Notifying callers of wait time without a VRU

You can use EWT to notify callers of their expected wait time without a VRU. This can be done using recorded announcements and by associating each recorded announcement with a time band, as shown in the following example.

```
VECTOR 101
1.queue-to split 3 pri h
2.goto step 4 if expected-wait for call <= 600
3.busy
4.wait-time 12 seconds hearing ringback
5.announcement 3001 [Thank you for calling ABC Inc. All agents
  are busy, please wait and we will get to your call as soon as
  possible]
6.goto vector 202 if unconditionally

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

VECTOR 202
1.goto step 13 if expected-wait for call > 280
2.goto step 11 if expected-wait for call > 165
3.goto step 9 if expected-wait for call > 110
4.goto step 7 if expected-wait for call > 55
5.announcement 3501 [Thank you for waiting.
  Your call should be answered within the next minute.]
6.goto step 14 if unconditionally
7.announcement 3502 [Thank you for waiting. Your call should be
  answered within approximately one to two minutes.]
8.goto step 14 if unconditionally
9. announcement 3503 [Thank you for waiting. Your call should be
  answered within approximately two to three minutes.]
10.goto step 14 if unconditionally
11.announcement 3504 [Thank you for waiting. Your call should be
  answered within approximately three to five minutes.]
12.goto step 14 if unconditionally
13.announcement 3505 [We apologize for the delay. Due to heavy
  call volume, you may have to wait longer than five minutes
  to speak to a representative. If possible, we suggest that you
  call between the hours of 8am and 10am for the fastest service.]
14.wait-time 120 secs hearing music
15.goto step 1 if unconditionally
```

In step 1 of the example shown above, the call is queued to split 3 at high priority. If the call fails to get a queue slot in split 3, if split 3 has no working agents, or if the wait time in split 3 at high priority exceeds 10 minutes, step 2 fails and the caller receives a busy signal. If step 2 succeeds, the caller hears ringback and an announcement and is then sent to vector 202.

Steps 1 through 4 of vector 202 determine tests to determine a predicted time band interval for the remaining queuing time for the call. One of five recorded announcements is then played to provide the caller with an expected wait time.

You may want to program your vectors so that few callers experience wait times that exceed the wait time of the announcement. In the example shown above, the EWT thresholds are set lower than the times that are quoted in the recorded announcements.

Using EWT to route to the best split

You may want to use EWT to change the normal queuing strategy for multiple splits to ensure that calls are answered in the shortest possible time. However, this strategy uses additional system resources and can make it more difficult to read and analyze split reports. Alternately, you can use EWT to identify the best for each call and avoid multiple split queuing.

The following vector example shows a scenario that includes a main split (1) and a backup split (2). In this example, the preference is for an agent from the main split service the call, but a 30-second maximum wait time is a competing preference.

The strategy in this vector is to use the backup split only if the backup split can answer the call within 30 seconds and the main split cannot.

```
1.goto step 5 if expected-wait for split 1 pri m <= 30
2.goto step 5 if expected-wait for split 2 pri m > 30
3.check split 2 pri m if unconditionally
4.goto step 6 if unconditionally
5.queue-to split 1 pri m
6.wait-time 12 secs hearing ringback
7.announcement 3501
8.converse-on split 18 pri m passing wait and none
9.wait-time 120 secs hearing music
10. goto step 8 if unconditionally
```

In the example shown above, step 1 branches to step 5 (queue to the main split) if the main split can answer the call within 30 seconds. If the main split cannot answer the call within 30 seconds, step 2 checks to see if the backup split can answer the call within 30 seconds. If the test fails, the call branches to step 5 and is queued to the main split. If possible, the call is queued to the backup split in step 3. At this point, the call is queued either to the main split or to the backup split, but not to both.

Steps 6 through 10 provide audible feedback to the caller while the call is in the queue. Note that in step 8, which is executed every 2 minutes, a VRU is used to provide the caller with his or her remaining wait time.

Factors that affect EWT values

Various factors can either increase or decrease the EWT value returned to the communication server.

This section includes the following topics:

- Factors that increase EWT for a split priority level on page 265
- Factors that decrease EWT for a split priority level on page 265

Factors that increase EWT for a split priority level

The most common causes for an increase in EWT for a split priority level are:

- The number of calls that are in queue increases
- Agents log out
- Agents go on break or are otherwise in the AUX work mode
- Agents are moved to another split
- Agents with multiple splits answer an increasing number of calls in other splits

Other conditions that may also cause EWT for a split priority level to increase include:

- The average talk time increases
- The number of calls at a higher priority increases
- The number of direct agent calls increases
- The number of RONA calls increases
- The number of abandoned calls decreases
- The number of calls that are queued in this split but answered in another decreases.

Factors that decrease EWT for a split priority level

The most common causes for a decrease in EWT for a split priority level are:

- The number of calls in queue decreases
- Agents log in (and start answering calls)
- Agents return from break or otherwise are no longer in the AUX work mode
- Agents are moved from another split
- Agents with multiple splits answer fewer calls in other splits

The following conditions may also cause a decrease in EWT for a split priority level:

- The average talk time decreases
- The number of calls at higher priority decreases
- The number of direct agent calls decreases
- The number of RONA calls decreases
- The number of abandoned calls increases
- The number of calls queued in this split but answered in another increases

Troubleshooting EWT

To verify that your EWT is operating as intended, use the **`list trace ewt`** command to observe processing events of all calls.For more information, see Appendix D: Troubleshooting vectors on page 689.

Note:
The **`list trace ewt`** command is blocked when the Tenant Partitioning feature is enabled.

Rolling Average Speed of Answer (ASA)

Rolling ASA Routing helps you to make routing decisions that are based on the current average time that it takes for a call to be answered in a split or VDN. In this way, a vector can route a call to the VDN or split where it is likely to be answered most quickly.

This section includes the following topics:

- Rolling ASA versus interval ASA on page 267
- Rolling ASA split calculation on page 268
- Rolling ASA VDN Calculation on page 268
- When to use rolling ASA on page 267
- Combining VDN and ASA routing on page 269

Rolling ASA versus interval ASA

The ASA calculation used for vector routing is called *rolling* ASA to differentiate it from the *interval* ASA that is recorded in Basic Call Management System (BCMS) and Avaya Call Management System (CMS) reports.

Rolling ASA is a running calculation that does not take into account the 15-minute, half-hour, or hour reporting intervals. It does not reflect interval boundaries.

The interval ASA used for BCMS and CMS reports is calculated on reporting interval boundaries and clears to zero at the start of each reporting interval.

The rolling ASA for a split or VDN is calculated based on the speed of answer for all calls recorded since system start-up, and is recalculated every time a call is answered. During each calculation, the speed of answer for the current call is given a weighted value that is greater than previous calls. Approximately 95% of the value of rolling ASA is obtained from the previous ten calls.

Note:
Calls that are not answered, such as calls that receive a forced busy, are not considered in the rolling ASA calculation.

The rolling ASA is calculated for an entire split or VDN. The calculation does not consider the priority levels of answered calls.

When to use rolling ASA

Rolling ASA is best used to test whether vector processing should queue the call to additional splits/skills when the main split/skill does not currently meet the targeted threshold.

Rolling ASA conditionals should *not* be used to prevent calls from queuing to the main split/skill or being answered in the principal VDN. If no calls are being answered in the main split/skill or VDN, the value of rolling ASA does not change. This could result in all future calls being locked out of the main split/skill or VDN unless there are other call vectors in the system that are directing calls to them.

Important:
To implement a call flow that tests whether or not to queue a call to a main split/skill, use the EWT feature.

Rolling ASA split calculation

The rolling ASA for a split is the average call answer time, as specified by the time interval that starts when call processing attempts termination to a split, and ends when the call is answered in that split. The measured interval includes both time in queue and ringing time at the agent station.

If the call is answered in another split or the call is abandoned by the caller, rolling ASA is not recorded for the call. If a call flows into a split from another split, the time queued and ring time for the previous split are not included. If a call is queued in multiple splits, only the rolling ASA for the split in which the call is answered is measured.

Rolling ASA VDN Calculation

The rolling ASA for a VDN is the average call answer time, as specified by the time interval that starts when call processing is initiated within the VDN until it is answered. The measured interval includes:

- Time elapsed in vector processing, including time in announcements.
- If the call is answered by an agent, time in queue and time ringing at the agent station.

Note:
If a call flows between VDNs, only the time elapsed in the answering VDN is used in the calculation.

Specifying VDNs

Rolling ASA follows the rules used for other Advanced Vector Routing conditionals to specify a VDN in a **goto** step:

- A VDN number.
- The value designated as *latest*. The latest VDN is the VDN currently processing the call. The latest VDN is not affected by VDN override settings.
- The value *active*. The active VDN is the VDN of record, which is the called VDN as modified by override rules. For example, if a call routes from a VDN with override set to **yes** then the new VDN is the active VDN. If a call routes from a VDN with override set to **no,** the previous VDN is the active VDN.

Combining VDN and ASA routing

The following vector example combines VDN and split ASA routing.

```
1.queue-to split 10 pri h
2.goto step 6 if rolling-asa for split 10 <= 30
3.check split 11 pri h if rolling-asa <= 30
4.check split 12 pri h if rolling-asa <= 30
5.check split 13 pri h if rolling-asa <= 30
6.announcement 10000
7.wait-time 40 secs hearing music
8.goto step 3 if unconditionally
```

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 of the backup splits and goes directly to the announcement in step 6. The assumption is that the call will be 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 greater than 30 seconds, steps 3, 4, and 5 check the backup splits. The call is queued to any of these splits that have a rolling ASA of 30 seconds or less. If the call still is not answered by the time that vector processing reaches step 8, the backup splits are checked again.

VDN Calls

VDN Calls routing allows you use the counted-calls conditional to make routing decisions on the number of incoming trunk calls that are currently active in a VDN.

This section includes the following topics:

- How VDN Call counts are calculated on page 269
- Using the counted-calls conditional on page 271

How VDN Call counts are calculated

The counted-calls conditional allows a vector to limit the number of simultaneous calls directed to a particular VDN. For example, if a service agency is contracted to handle 100 simultaneous calls for a client, calls in excess of that number can be routed to a **busy** step.

VDN Call counts follows the rules used for other Advanced Vector Routing conditionals to specify the VDN in a **goto** step:

- A VDN number.
- The value designated as *latest*. The latest VDN is the VDN currently processing the call. The latest VDN is not affected by VDN override settings.
- The value *active*. The active VDN is the VDN of record, which is the called VDN as modified by override rules. For example, if a call routes from a VDN with override set to **yes** then the new VDN is the active VDN. If a call routes from a VDN with override set to **no,** the previous VDN is the active VDN.

When Advanced Vector Routing is enabled, a count of active incoming trunk calls is kept for each VDN. The VDN counter increments each time that an incoming call is placed to the VDN and decremented each time that an incoming call is released. A call is considered active in a VDN from the time the call routes to the VDN until all parties on the call are dropped and the call is released.

Note:
The call is counted for the originally called VDN only. When a call is routed to another VDN, the call counter for the subsequent VDN does not increment, nor does the call counter for the original VDN decrement.

The VDN Call count includes the following types of calls:

- Incoming trunk calls routed directly to the VDN.
- Incoming trunk night service calls in which the VDN is the night service destination.
- Calls that cover or forward to the VDN if it is the first VDN routed to and the call is an incoming trunk call.
- Already counted calls that are conferenced with counted or not counted calls from the same VDN.

The VDN call count does not include:

- Internal calls to the VDN.
- Calls that are transferred to the VDN.
- Calls that are redirected to their VDN return destination.
- Conferenced calls that were previously counted on different VDNs.

Using the counted-calls conditional

The following vector example shows how the **counted-calls** conditional can be used to route calls.

Using VDN call counting to route calls

```
1.goto step 3 if counted-calls to vdn 1234 <= 100
2.busy
3.queue-to split 60 pri l
4.wait-time 20 seconds hearing ringback
5.announcement 27000
6.wait-time 60 seconds hearing music
7.goto step 5 unconditionally
```

If more than 100 calls are active in VDN 1234, the caller hears a busy signal and vector processing is terminated. If 100 or fewer calls are active, the call queues to split 60.

Look-Ahead Interflow (LAI)

This section includes the following topics:

- About LAI on page 273
- LAI prerequisites on page 274
- Example of a two-switch configuration on page 275
- Command set on page 275
- How traditional LAI works on page 277
- How enhanced LAI works on page 281
- LAI-initiated path replacement for calls in vector processing on page 289
- DNIS and VDN override in an LAI environment on page 290
- LAI with network ADR on page 292
- Multi-site applications for Enhanced LAI on page 293
- LAI considerations on page 294
- Troubleshooting for LAI on page 295

About LAI

Look-Ahead Interflow (LAI) enhances Call Vectoring for call centers with multiple ACD locations. LAI allows these centers to improve call-handling capability and agent productivity by intelligently routing calls among call centers to achieve an improved ACD load balance. This service is provided by ISDN D-channel messaging over QSIG or non-QSIG private networks, virtual private networks, or public networks. The receiving switch is able to accept or deny interflowed calls sent by the sending switch.

LAI has the following basic attributes:

- Produces First in First Out (FIFO) or near-FIFO call processing
- Includes enhanced information forwarding, that is, codeset 0 user information transport

LAI prerequisites

The following items are criteria for basic LAI call control operation over a virtual private network or a public switched network:

- The sending and receiving call center locations must have ISDN (PRI or BRI) trunk facilities.

  Note:
  ATM trunking and IP trunking can be set up to emulate ISDN PRI. For information on setting this up, see *Administration for Network Connectivity for Avaya Communication Manager*, and *ATM Installation, Upgrades and Administration using Avaya Communication Manager*.

- The switch must support the ISDN country protocol.
- LAI has been tested with several major carriers. To find out if these capabilities work with your carrier, check with your account team for the most current information. If testing has not been done to verify operation over the public networks that are involved with the preferred specific configuration, use of private ISDN trunking between the nodes should be assumed until successful testing is complete.
- The ISDN SETUP and DISCONNECT messages are transported between sending and receiving locations, for example, SS7 or equivalent public network connectivity.
- A receiving-end generated DISCONNECT message must transmit back to the sending the switch call center without changing the cause value.

  Conversion of the DISCONNECT message to a progress message (with a Progress Indicator Description set to 1 and a cause value other than 127 included) is a valid reject message and compatible with LAI.
- Progress messages that are generated towards the sending end by intervening network switches must have the Progress Indicator Description set to 8 so that the switch does not consider the call accepted or rejected.
- ISDN codeset 0 user information transport supports LAI information forwarding. As an alternative, LAI can use dedicated VDNs at the receiving location to provide an equivalent display of the forwarding application identity and set trunk group options to not send either the codeset 6/7 LAI IE or codeset 0 information transport.

  Note:
  Best Service Routing (BSR) cannot use these LAI alternatives. BSR must use ISDN codeset 0 user information transport.

Example of a two-switch configuration

Look-Ahead Interflow (LAI) is enabled through the use of call vectors and their associated commands. For a two-switch configuration, these vectors are included in both the sending switch, which processes vector outflow, and the receiving switch, which processes vector inflow.

Command set

LAI enhances call vectoring so that calls interflow only to those remote locations that can accept the calls.

LAI is achieved through a set of vector commands. The following table lists the call-acceptance vector commands that are used in LAI.

Call-acceptance vector commands

Command	Qualification
`announcement`	Announcement available Queued for announcement Retrying announcement
`check split`	Call terminates to agent Call queued to split
`collect digits`	Always (except for ced and cdpd digits, which are neutral)
`converse-on split`	VRU answers the call Call queued to converse split
`disconnect`	With announcement and announcement available With announcement and queued for announcement With announcement and retrying announcement
`messaging split`	Command successful Call queued
`queue-to split`	Call terminates to agent Call queued to split

Call-acceptance vector commands

Command	Qualification
route-to	Terminates to valid local destination Successfully seizes a non-PRI trunk Results in a LAI call attempt, and the call is accepted by the far-end switch
wait-time	Always (except **wait-time hearing i-silent**, which is neutral)

If the receiving switch decides it is unable to accept the LAI call, call denial is accomplished by executing one of the vector commands that are listed in the following table.

Note:
It is recommended that you use **busy** instead of **disconnect** to allow for compatibility with similar network services such as Alternate Destination Redirection (ADR).

Call-denial vector commands

Command	Qualification
busy	Always
disconnect	Without announcement With announcement but announcement unavailable
reply-best	Always; used with BSR

The vector commands that are shown in the following table are considered neutral because they do not generate either call acceptance or denial messages.

Neutral vector commands

Command	Qualification
adjunct routing link	Always
announcement	Announcement unavailable
check split	Call neither terminates nor queues
collect ced/cdpd digits	Always
consider	Always - used with BSR
converse-on split	Call neither terminates nor queues
goto step	Always

Neutral vector commands (continued)

Command	Qualification
`goto vector`	Always
`messaging split`	Command failure
`queue-to split`	Call neither terminates nor queues
`route-to`	Unsuccessful termination Trunk not seized LAI call denied by the far-end switch
`stop`	• Always
`wait-time hearing i-silent`	• Always **Note:** This command is used following an `adjunct routing link` command in applications where the adjunct decides whether to accept or reject the Look-Ahead calls.

How traditional LAI works

Traditional LAI is recommended when the preferred call flow performs LAI attempts before queuing the call.

This section includes the following topics:

- LAI commands on page 277
- Example of traditional LAI on page 279
- Receiving switch operation on page 279

LAI commands

LAI uses the commands that are included within the Basic Call Vectoring and Call Prompting features:

- `route-to number with coverage n` or `route-to digits with coverage n` command on a switch that has LAI enabled and that successfully seizes an ISDN trunk automatically results in a normal LAI call attempt being placed. The call attempt can be rejected or accepted by the remote end.

- **route-to number with coverage y** or **route-to digits with coverage y** command never results in a LAI call attempt. The sending end assumes that the call is always going to be accepted. This command always completes the call. Moreover, the command should not be used when the vector at the receiving location ends up denying the call, since the caller in this case is given a busy signal, or the call is disconnected. Use this command with coverage set to **y** only for those cases when an unconditional interflow is wanted (with LAI active) and the terminating switch is set up to handle this type of call.

When a LAI call attempt is made, Call Vectoring at the sending location checks a potential receiving location to determine whether to hold or send the call. While this is done, the call remains in queue at the sending location. As such, the call can still be connected to the sending-location agent if one becomes available before the receiving location accepts the call.

Call Vectoring at the receiving location decides whether to accept the call from the sending location or to instruct the sending location to keep the call. In the latter case, the sending location can then either keep the call, check other locations, or provide some other treatment for the call. Conditions for sending, refusing, or receiving a LAI call attempt can include a combination of any of the following:

- Expected wait time for a split
- Number of staffed or available agents
- Number of calls in queue
- Average speed of answer or the number of calls active in a VDN
- Time of day and day of week
- Any other legitimate conditional

If the call is accepted by the receiving switch, the call is removed from any queues at the sending switch, and call control is passed to the receiving switch. If the call is denied by the receiving switch, vector processing continues at the next step at the sending switch. Until the call is accepted by either switch, the caller continues to hear any tones applied by the sending switch. If the call is denied, the call vector can apply alternate treatment, such as placing another LAI call to an alternate backup switch.

Note:

The LAI operation is completely transparent to the caller. While a LAI call attempt is being made, the caller continues to hear any audible feedback that is provided by the sending switch vector. The caller also maintains his or her position in any split queues until the call is accepted at the receiving switch.

LAI passes Call Prompting digits collected in the sending switch to the receiving switch by codeset 0 user information transport. For more information, see Information Forwarding on page 189.

Example of traditional LAI

The vectors in the sending switch use the **goto** command to determine whether the call should be sent to the receiving switch. Recall that the **goto** command tests various outflow threshold conditions such as expected wait time. If the expressed condition is met, a branch is made to the appropriate **route-to** command. This command sends the call to the receiving switch, which, as already noted, can accept or deny the call.

The following example shows an outflow vector that might be included in a sending switch.

Using LAI with route-to commands to outflow calls

```
1.wait-time 0 secs hearing ringback
2.goto step 5 if expected-wait for split 3 pri m < 30
3.route-to number 5000 with cov n if unconditionally
4.route-to number 95016781234 with cov n if unconditionally
5.queue-to split 3 pri m
6.announcement 3001
7.wait-time 30 secs hearing music
8.goto step 6 if unconditionally
```

If split 3 has an expected wait time of less than 30 seconds (step 2), step 5 queues the call to the split's queue at a medium priority.

If the expected wait time is 30 seconds or more, LAI attempts are made in steps 3 and 4. If the call is accepted by one of the receiving switches call control passes to the receiving switch.

If the receiving switches deny the call, the call queues to split 3 and announcement 3001 plays. The caller then hears music (interrupted by announcement 3001 every 30 seconds).

Receiving switch operation

When the receiving switch receives the LAI request, the call first routes to a VDN. The VDN then maps the call to the receiving switch's inflow vector, and vector processing begins, starting with inflow checking. Inflow checking is enabled by conditional **goto** commands in the inflow vector. The decision to accept or deny a call can be based on checks such as any of the following:

- Expected Wait Time
- Number of staffed agents
- Number of available agents
- Time-of-day/day of the week
- Number of calls in split's queue
- Average Speed of Answer

- Active VDN Calls
- ANI
- II-Digits
- CINFO ced and/or cdpd digits
- Collected digits forwarded from the sending switch

Once inflow checking is complete, acceptance of the LAI call is accomplished by executing any of the vector commands listed in Call-acceptance vector commands on page 275.

Note:
For each of the commands listed in Call-acceptance vector commands on page 275, Neutral vector commands on page 276 and Call-denial vector commands on page 276, only one of the corresponding qualifications needs to be true for the command to effect the desired result, which is call acceptance, call denial, or no effect on such acceptance or denial.

The following example shows an inflow vector that might be used by a receiving switch.

Using inflow checking for LAI requests

```
1.goto step 6 if expected-wait 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
```

Step 1 of this inflow vector checks the inflow thresholds. The **goto step** command in step 1 checks the expected wait time in split 1. If the expected wait time is greater than 30 seconds, a branch is made to the **busy** command in step 6. If executed, the **busy** command denies the call, and the receiving switch returns a call denial message to the sending switch. The sending switch, in turn, drops the LAI call attempt and then continues vector processing at the next vector step.

If the expected wait time in split 1 is less than or equal to 30 seconds, the receiving switch returns a call acceptance message to the sending switch, and call control is passed to the receiving switch. Thereafter, the call is queued to split 1 in the receiving switch (step 2). Once queued, the caller receives the appropriate announcement in step 3 and is then provided with music until the call is answered by an agent or abandoned by the caller (steps 4 and 5). Remember that the **stop** command halts vector processing but does not drop the call.

If the sending switch does not receive a call acceptance or call denial message within 120 seconds after the LAI call request, the LAI attempt is dropped. The sending switch continues vector processing at the next step.

How enhanced LAI works

This section includes the following topics:

- About enhanced LAI on page 281
- The simple way to achieve FIFO on page 281
- Detailed information about the interflow-qpos conditional on page 282
- When does a call not interflow? on page 283
- How the minimum EWT is set on page 284
- Example of single-queue multi-site operation on page 285
- Example of maintaining FIFO processing with LAI on page 286
- Single-queue FIFO considerations on page 286
- Example of LAI in a tandem switch configuration on page 287
- Sending switch operation on page 287
- Tandem switch operation on page 288
- Far-end switch operation on page 288

About enhanced LAI

Enhanced LAI uses the same basic vectoring commands as traditional LAI, but adds the conditional **interflow-qpos**. Enhanced LAI is recommended when the preferred call flow performs LAI attempts after queuing the call.

Using enhanced LAI **interflow-qpos** conditional:

- Produces First in First Out (FIFO) or near FIFO call processing
- Uses less processing during LAI

The simple way to achieve FIFO

You can use the **interflow-qpos** conditional in a **route-to** or **goto** command to achieve FIFO results.

For example, you can use the following **route-to** command with the conditional to achieve FIFO results:

```
route-to number 9581234 with cov n if interflow-qpos=1
```

If you have a lot of remote agents, you may want to set the **route-to** command as follows:

```
route-to number 9581234 with cov n if interflow-qpos<=2
```

Detailed information about the interflow-qpos conditional

You can use this feature without understanding the differences between split queues and eligible queues or between **interflow-qpos** and queue position. There are features that are built into enhanced LAI so that when you write a step such as **route-to number 9581234 with cov n if interflow-qpos=1**, the system operates smoothly under all conditions.

The interflow-qpos conditional

The **interflow-qpos** conditional only applies interflow processes to a dynamic eligible queue and to calls that are queued locally before the **route-to** is attempted.

The eligible queue is that portion of the split/skill queue that:

- Includes only calls that are not expected to be answered locally during the interflow process at that moment relative to the call being processed
- Does not include direct agent calls because these calls are excluded from any interflow process.

The following is an example of the **interflow-qpos** conditional used in a **route-to** command:

```
route-to number _____ with cov _ if interflow-qpos CM x
```

where

- ***CM*** is the comparator. It is one of three symbols: =, <, <=
  - With **if interflow-qpos = x**, the call is interflowed if it is at the ***x*** position from the top of the eligible queue.
  - With **if interflow-qpos < x**, the call is interflowed if it is among the top ***x-1*** of the eligible queue.
  - With **if interflow-qpos <= x**, the call is interflowed if it is among the top ***x*** eligible calls.
- ***x*** indicates the call's position in the eligible queue. Valid queue positions are 1 through 9. The top queue position is 1. The eligible queue is made up of calls from the first local split/skill that the call has been queued to due to previous steps in the vector.

Note:
Calls that are likely to be serviced locally before an LAI can be completed are not eligible for interflow since they are excluded from the eligible queue. Calls that are likely to be answered are identified based on conditions of the split/skill to which the call is queued and, under certain conditions, an administered minimum EWT threshold value.

The following is an example of the **interflow-qpos** conditional used in a **goto** command:

goto step/vector ____ if interflow-qpos CM x

where

- ***CM*** is the comparator. It is one of six symbols: =, <>, <, <=, >, >=
- ***x*** indicates the call's position in the eligible queue. Valid queue positions are 1 through 9. The top queue position is 1.

Calls that are likely to be serviced locally before an LAI can be completed are not eligible for interflow since they are excluded from the eligible queue.

When does a call not interflow?

A call does not interflow under the following circumstances:

- If the **interflow-qpos** conditional is not met.

  As with other conditionals, the **route-to number... if interflow-qpos** step or the **goto step/vector** branch is executed only if the conditional is met, otherwise vector processing goes to the next step.

- If the call is not in a split/skill queue or not in the eligible portion of the queue when the conditional step is executed.

  If the call is not in queue when the **route-to number... if interflow-qpos** step is executed, a vector event is logged and vector processing continues at the next step.

  If the call is not in queue when a **goto... if interflow-qpos** step is executed, the queue position of the call is considered to be infinite in determination of the conditional.

  Note:
  A vector event is not logged if the call is in queue, but is not in the eligible portion of the queue.

- Interflow failure or LAI rejection

  Interflow failure or LAI rejection will also go to the next step. Route-to operation and feature interactions will be the same as other configurations of the route to number command, for example, **route to number ___ with cov _ if digit *CM x***.

The following table outlines what action is taken for different cases of interflow eligibility.

Actions taken for cases of interflow eligibility

Case	Action at route-to step	Action at goto step
The call not eligible for interflow.	The call is never routed.	Treat as if the interflow queue position is infinite.
The call is not in any split queue.	The call is treated as if the interflow queue position is infinite.	Treat as if interflow queue position is infinite.
The call is eligible for interflow.	Act according to the conditional.	Act according to the conditional.

How the minimum EWT is set

The minimum expected wait time (EWT) threshold that is used to help determine which calls are more likely to be answered locally is administered on the Feature-Related System Parameters screen. Minimum EWT is used when the local agents, that is, in the first split/skill to which the call is queued, are handling a significant number of the calls. If these agents are not handling a significant number of calls, the call is eligible for LAI even if its EWT is lower than the threshold.

Note:
When enhanced LAI vectors or the look-ahead EWT threshold are administered inappropriately, remote agents may experience phantom calls or a delay between becoming available and receiving an ACD call.

The instructions below assume that you use a SAT terminal or terminal emulator to administer the switch.

To set the minimum EWT threshold:

1. In the command line, type **`change system-parameters feature`** and press **Enter**.

   The system displays the Feature-Related System Parameters screen.

2. Find the page of the Feature-Related System Parameters screen that has the **Interflow-Qpos EWT Threshold** field.

   If Look-Ahead Interflow is active, the **Interflow-Qpos EWT Threshold** field can be administrated.

3. In the **Interflow-Qpos EWT Threshold** field, enter the number of seconds, as a number from 0 to 9, that you want for the EWT threshold. The default of 2 seconds is recommended.

   Note:
   When the look-ahead EWT threshold field is set too low, remote agents may experience phantom calls.

4. Press **Enter** to save your changes.

Example of single-queue multi-site operation

In this scenario, all new calls for a given customer application are routed by the public network to only one of the switches in the network, where the calls are put in the queue.

Local agents service the calls from the queue in the normal fashion; however, remote agents service calls by means of enhanced look-ahead.

The switch with the call queue does rapid enhanced look-ahead attempts to all other switches in the network that can service this call type, looking for an available agent.

Normally, the look-ahead attempts are placed only on behalf of the call that is at the head of the queue (**interflow-qpos = 1**). However, in scenarios where there are large numbers of agents at a remote switch, it may be necessary to do interflows on behalf of more than one call in order to outflow a sufficient volume of calls to keep all agents busy (**interflow-qpos <= 2**).

Vector to back up split

```
1. announcement 3501
2. wait-time 0 secs hearing music
3. queue-to skill 1 pri m
4. route-to number 93031234567 with cov n if interflow-qpos = 1
5. route-to number 99089876543 with cov n if interflow-qpos = 1
6. wait-time 5 secs hearing music
7. goto step 4 if unconditionally
```

In this example, interflow call attempts are placed on behalf of the call that is at the beginning of the queue every 5 seconds to the two other switches in the network.

If queuing times are very long, 5 minutes, for example, and the call is not near the beginning of the queue, it is wasteful to go through the vector loop from step 4 to step 7 every 5 seconds. For this reason, the on page 286 is more efficient.

Example of maintaining FIFO processing with LAI

One of the advantages of enhanced LAI is the ability to provide FIFO or near-FIFO call processing. The following example shows a vector that is used to achieve such call processing.

FIFO processing vector

```
1.announcement 3501
2.wait-time 0 secs hearing music
3.  queue-to skill 1 pri m
4.goto step 7 if interflow-qpos < 9
5.wait-time 30 secs hearing music
6.goto step 5 if interflow-qpos >= 9
7.route-to number 93031234567 with cov n if interflow-qpos = 1
8.route-to number 99089876543 with cov n if interflow-qpos = 1
9.wait-time 5 secs hearing music
10.goto step 7 if unconditionally
```

In this vector:

- The rapid look-ahead loop is only entered when the call reaches one of the top 8 positions in queue.
- The number of executed vector steps is reduced dramatically when call waiting times are long.

It is important to write vectors so that calls at the head of the queue have advanced to the rapid look-ahead loop by the time their turn to interflow has been reached. In the vector example shown above, if 8 calls can be serviced from queue in less than 30 seconds (which is the loop time on step 5), there can be a delay in outflowing calls to available agents at the remote sites.

Single-queue FIFO considerations

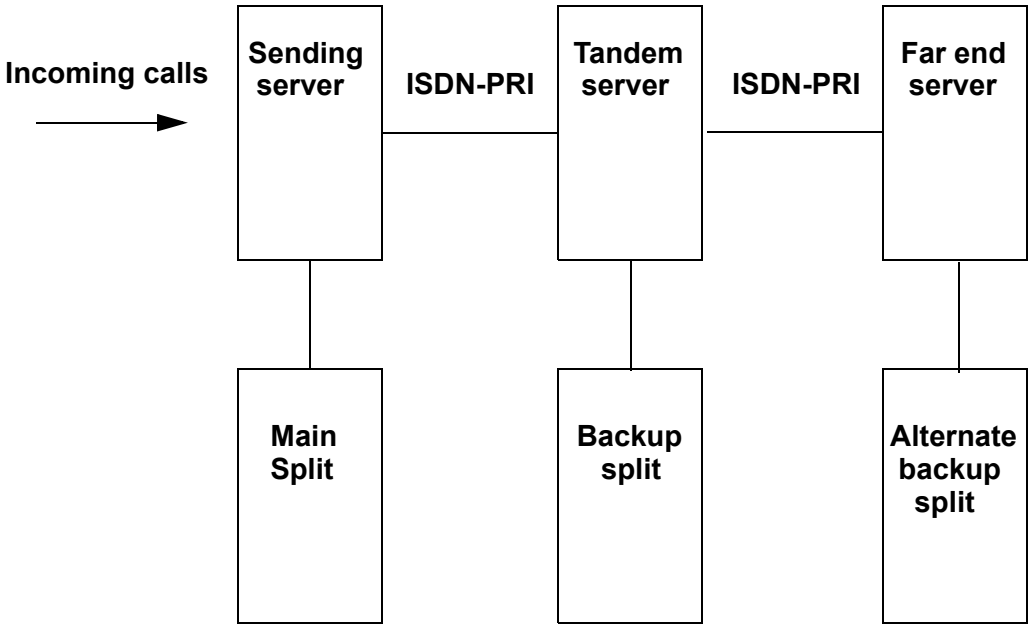
The following issues need to be taken into consideration for FIFO in a single queue:

- When there are available agents, calls are always delivered to available agents at the queuing switch before available agents at the remote switches.
- When there are calls in the queue and agents serve calls from multiple applications, the agents always service calls from the applications that are queued locally before calls from applications that are queued at another switch.
- Backup VDNs and vectors are recommended in order to provide continuous operation in the event of a failure at a queuing switch.
- EWT predictions cannot be made if the split/skill in which the calls are queued has no working agents.
- EWT predictions may be temporarily inaccurate if there are sudden, major changes in the number of working agents in the split/skill in which the calls are queued.

Example of LAI in a tandem switch configuration

Tandem LAI is implemented by using **route-to** commands that contain external destinations that route over ISDN facilities. This configuration is shown in the following figure.

LAI using a tandem switch



Sending switch operation

The sending switch is unaware that its LAI call is being tandemed to an alternate switch. The operation of the sending switch in the tandem switch configuration is the same as that in the two-switch configuration.

Tandem switch operation

If the receiving switch executes a **route-to** command that routes the call over an ISDN facility before call acceptance, the **route-to** command is performed on a look-ahead basis in the same manner as a sending switch. If the call is accepted at the far-end switch, acceptance is passed to the sending switch, and call control is passed to the far-end switch, along with tandeming of the original calling party information and the original DNIS name. If the call is denied, the next step of the tandem switch vector is executed.

The following example shows a tandem switch vector.

Tandem switch vector example

```
1. goto step 6 if expected-wait in split 30 pri h > 30
2. queue-to split 30 pri h
3. announcement 200
4. wait-time 2 seconds hearing silence
5. stop
6. route-to number 4000 with cov n if unconditionally
7. busy
```

Step 1 of this vector checks the inflow threshold. If the inflow criteria are acceptable, the vector flow drops to step 2, where the **queue-to split** command provides acceptance to the sending switch. Thereafter, steps 3 through 5 provide a typical queuing-wait scheme.

If the inflow criteria are not acceptable, a branch is made to step 6. The **route-to** command in this step checks another switch that is enabled with LAI on a look-ahead basis. If this far-end switch rejects the call, a denial message is relayed back to the tandem switch, which then drops the LAI call attempt. On the other hand, if the far-end switch accepts the call, an acceptance message is relayed all the way back to the sending switch.

No ringback is provided in this tandem switch vector. This is necessary so that an acceptance message is not returned to the sending switch. This operation is appropriate for the caller because the sending switch has already returned an announcement before a LAI attempt is made to the receiving switch.

Be sure that the sending switch is not used as a backup location for the tandem switch or for any of the far-end switches. If the sending switch is administered in this manner, all trunk facilities could be tied up by a single call.

Far-end switch operation

The far-end switch is also unaware that tandeming has taken place. The far-end switch functions in the same manner as the receiving switch within the two-switch configuration.

LAI-initiated path replacement for calls in vector processing

This section includes the following topics:

- About path replacement for calls in vector processing on page 289
- Example vector on page 289

About path replacement for calls in vector processing

Path replacement for calls in queue and vector processing can be accomplished using QSIG or DCS with Reroute using ISDN SSE. For calls that are waiting in queue or in vector processing, even if the call is not connected to an answering user, path replacement can be attempted to find a more optimal path for this call. This results in more efficient use of the trunk facilities.

The **route-to** command is used in LAI to initiate a QSIG path replacement for a call. The following scenario can take place. At the terminating communication server, if a Path Replacement Propose operation is received for a call that is in queue or vector processing, the switch can immediately initiate path replacement using the Path Replacement Extension if the **Path Replace While in Queue/Vectoring** field is set to **y** and the **Path Replacement Extension** field has a valid entry. These fields are located on the ISDN parameters page of the Feature-Related System Parameters screen.

The ability to track a measured ACD call after a path replacement has taken place is available for CMS versions r3v9ai.o or later. Starting with the r3v12ba.x release, CMS reports a path replacement as a *rename* operation rather than a path replacement. The *rename* operation properly reports scenarios where a path replacement takes place from a measured to an unmeasured trunk facility. Avaya recommends that you upgrade CMS to r3v12a.x or later and administer all trunks associated with path replacement as *measured* by CMS to ensure better CMS tracking of path-replaced calls.

Example vector

The following example shows how an LAI vector can be written to trigger path-replacement at the terminating switch.

Note:
In order for a path-replacement to be attempted, the incoming and outgoing trunks that are used for the call must be administered with the **Supplementary Service Protocol** field set to **b**.

LAI-initiated path-replacement vector

```
1.wait 0 seconds hearing music
2.queue-to skill "n" if available-agents < 6
3.route-to number "ARS number for ISDN trunk" with cov n
4.wait 999 seconds hearing ringback
```

At the receiving communication server, the vector that processes the incoming call must use an **announcement**, or **wait hearing music** vector command to enable path-replacement.

DNIS and VDN override in an LAI environment

This section includes the following topics:

- About DNIS and VDN override on page 290
- DNIS information displayed to answering agent on page 291
- Originator's display on page 292

About DNIS and VDN override

LAI handles Dialed Number Identification Service (DNIS) and VDN Override in various ways, depending on a number of different characteristics of the call. DNIS, as described in Call Vectoring fundamentals on page 31, allows any agent with a display-equipped telephone to receive visual displays that specify the name of the called VDN. VDN Override in its basic form allows the name of a subsequently routed to VDN to be displayed to the answering agent instead of the name of the originally called VDN.

The following sections discuss how LAI handles DNIS and VDN Override.

DNIS information displayed to answering agent

For LAI, the DNIS name, which is the called VDN name from the sending switch, is presented on the display for the answering agent on the receiving switch if all of the following are true:

- The LAI option is enabled.
- The call routes to a VDN.
- The DNIS name field is not blank.

The type of DNIS information that is displayed depends upon a number of different scenarios. This information is presented in the following table.

DNIS information displayed for LAI scenarios

Scenario	Information displayed
Tandem LAI call	Look-Ahead Interflow DNIS information from the original LAI call.
No redirection at the sending switch	VDN name according to Override rules at the sending switch (active VDN).
Redirection at the sending switch (VDN in coverage path)	Original VDN name, or If multiple VDNs are accessed, the name of the VDN that was last accessed by a **route-to** command.
Sending switch sends a blank DNIS Name field (that is, a name is not assigned to the sending switch called VDN) or the trunk group is administered to not send the LAI name (see Information Forwarding on page 189).	Name associated with the receiving VDN. This name can be changed according to the rules of VDN Override at the receiving switch.

Note:

VDNs that map to vectors that place LAI calls must have their ISDN Calling Party Number (CPN) prefixes administered. If an ISDN CPN prefix is not administered, the assigned VDN name is not sent. Instead, a DNIS of all blank space characters is sent and displayed on the answering agent's terminal.

Originator's display

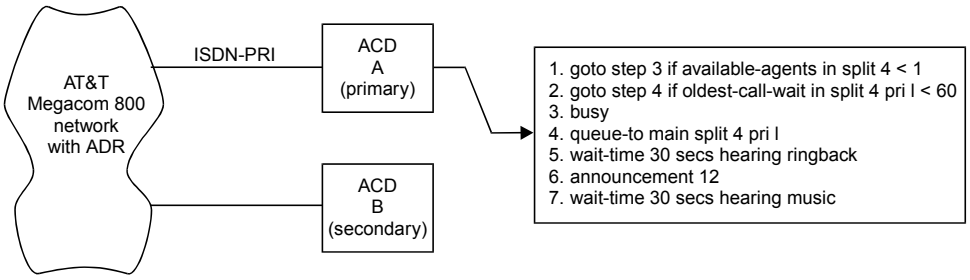
For internal calls, the originator's display contains the same information as for Basic Call Vectoring, but it is possible that the originator might receive unwanted display updates during LAI call attempts. In this case, LAI calls should go out over trunk groups that have the **Outgoing Display** field set to **n**. When the display field is set to no, internal callers who call that trunk group see the digits that they dialed on their display.

LAI with network ADR

Call Vectoring and LAI are compatible with and supplement the network services Alternate Destination Redirection (ADR) rerouting feature or equivalent service from other network providers. ADR uses ISDN-PRI connectivity with the switch in the same manner as LAI to allow the receiving system to indicate whether a call is to be accepted or rejected. The same type of vector that is used as a receiving ACD for LAI is used at the ADR-receiving ACD. If the call is accepted, it is connected to the system. If the call is rejected, the network routing number is translated to another number that routes the call to the alternate location within dialing-plan constraints. ADR allows for only one alternate location. LAI can be used at the alternate location to test other locations for less-busy conditions.

The following figure shows the configuration for a multilocation application.

ADR Example



The network requires ISDN-PRI connectivity to primary location A. Connection to secondary location B may or may not be ISDN-PRI. ADR attempts to route the call to location A over the ISDN-PRI link using a routing number that selects a VDN that is assigned to the receiving vector shown.

When the routing attempt is made, Call Vectoring starts processing the vector. The example then proceeds at location A as follows:

1. Step 1 checks for staffing of the ACD split, and branches to step 3 if it is not staffed.
2. If the ACD split is staffed, step 2 checks the oldest call waiting time in the split, and branches to step 4 if it is less than 60 seconds.
3. If the ACD split is unstaffed or if the oldest call waiting time is 60 seconds or more, step 3 rejects the call and returns a busy indication to the network.
4. If the oldest call waiting time is less than 60 seconds, step 4 accepts the call and queues it. ADR then connects the call through to the receiving system.
5. Steps 5 through 7 provide ringback, announcement, and music to the caller.

If the vector at location A rejects the call by sending a busy indication back to the network over the ISDN-PRI link, ADR reroutes the call to location B which must accept the call. If location B is closed or too busy to take the call, location B can use Call Vectoring and LAI to check other locations. If other locations exist and can take the call, location B can forward the call. If other locations do not exist or cannot take the call, location B can use Call Vectoring to route the call to location A. If location A is not open, location B can use Call Vectoring to provide an announcement or a busy tone to the caller.

Multi-site applications for Enhanced LAI

Enhanced LAI has two principal applications in a multi-site environment.

- It is possible to implement single-queue FIFO operation for any application. However, in many cases, Avaya recommends the use of BSR instead of LAI for maximum efficiency and flexibility. For more information, see Best Service Routing (BSR) on page 297.
- LAI can be used in combination with BSR for those switches in the network with extremely low call volumes.

  For more information about using BSR and LAI together, see Appendix E: Advanced multi-site routing on page 729

LAI considerations

The following are considerations for working with LAI:

- Never interflow to a remote vector that in turn might interflow back to the same local vector. This could cause a single call to use up all available trunks.
- Do not use the **oldest-call-wait** test condition in LAI vectors. OCW corresponds to the very next call to be answered and, as such, this test condition gives no information on the current state of call overload. For example, if OCW = 30 seconds, all we know from this is that the queue was overloaded 30 seconds ago. In place of **oldest-call-wait**, use the EWT conditional. For more information, see Expected Wait Time (EWT) on page 259.
- If an LAI call attempt is accepted by a step that contains a **queue-to**, **check split**, or **route-to** command, there is a small but finite interval during which the call could be answered by an agent at the sending switch before notification of acceptance is received by the sending switch. In this case, the caller is connected to the agent at the sending switch, while the agent at the receiving switch might receive a phantom call. For this reason, consider using a short **wait-time** or **announcement** step at the receiving switch to allow the call to be accepted and taken out of the queue at the sending switch. If call acceptance is to be based on available agents, use of a **wait-time > 0** seconds or an **announcement** is not recommended. A **wait-time** with 0 seconds of silence might be useful in this case.

  Note:
  For enhanced LAI operation, there are capabilities built into the feature to eliminate or reduce the occurrence of phantom calls. If phantom calls are a problem in an enhanced LAI operation, the **Interflow-Qpos EWT Threshold** field has been set too low.

- When an LAI call attempt is made, the TTR (if attached) is disconnected, and any dial-ahead digits are discarded. This implies that a subsequent **collect digits** command would require that the TTR be connected.
- Be sure that the feedback provided by the receiving switch after a successful LAI attempt is consistent with what the caller has already received.
- It is perfectly acceptable for a vector to route a call over an ISDN-PRI facility to a destination that is not a VDN. In this case, the sending switch treats the call as if it were a LAI call. Generic ISDN processing at the receiving switch causes the call to be accepted. The DNIS name is ignored.
- If a LAI call terminates to a VDN on a receiving switch where the LAI option is not enabled, intelligent interflow still results. However, any relevant DNIS information is ignored, and intelligent interflow to far-end switches is not possible.

- The LAI time-out in the sending switch occurs after 2 minutes.
- T-1 equipment might modify the ISDN D-channel that is used for LAI. If multiplexors are introduced into the ISDN-PRI circuit, bit compression and echo cancellation must be turned off for the D-channel.

Troubleshooting for LAI

The following are troubleshooting suggestions when working with LAI:

- If remote agents are experiencing a high volume of phantom calls, the **Interflow-Qpos EWT Threshold** may be set too low or too high.
- If remote agents are experiencing a delay between becoming available and receiving a call, the following may be the cause:
  - The **Interflow-Qpos EWT Threshold** may be set too low.
  - An insufficient number of LAI attempts have been made from the sending switch. In this case, change the **`interflow-qpos`** conditional at the sending switch. For example, change **`interflow-qpos=1`** to **`interflow-qpos <= 2`**.
  - An insufficient number of tie trunks are available.
- If remote agents are receiving no calls, the maximum number of vector steps that are executed at the sending switch vector may have been reached before calls reached the head of the queue. In this case, rewrite the vector on the sending switch.

Best Service Routing (BSR)

This section includes the following topics:

- About BSR on page 297
- Benefits of BSR on page 298
- Server and network requirements for BSR on page 300
- Special BSR terminology on page 302
- Single-site BSR on page 304
- Planning and administering single-site BSR on page 317
- Troubleshooting for single-site BSR on page 319
- Multi-site BSR on page 320
- Planning and administering multi-site BSR on page 342
- Local treatment for remotely queued IP and ISDN calls on page 346
- Troubleshooting for multi-site BSR on page 354
- Tips for writing BSR vectors on page 354
- BSR-initiated path replacement for calls in vector processing on page 355

About BSR

The Best Service Routing (BSR) feature compares specified splits/skills and selects the one that provides the best service to a call. To respond to changing conditions and operate more efficiently, BSR monitors the status of the specified resources and adjusts call processing appropriately.

BSR can be configured for either single-site or multi-site operation. Single-site BSR compares splits/skills on the communication server where the BSR resides to find the best resource to service a call. Multi-site BSR extends this capability across a network of communication servers, comparing local splits/skills, remote splits/skills, or both, and routing calls to the resource that provides the best service.

Benefits of BSR

Both single-site and multi-site BSR intelligently compare specific resources to find the one that can best service a call. In addition, multi-site BSR makes it possible for you to integrate a network of call centers for better load balancing and optimal agent utilization. Depending on your specific application, BSR can yield a variety of other benefits as shown in the following table.

Note:
If a call center network is heavily overloaded and a significant number of calls are being blocked or abandoned, shorter wait times may not result when BSR is used. Rather than reducing wait times, any productivity gains will allow more calls to gain access to the network.

Best Service Routing benefits

You can benefit from…	As a result of…
Increased revenue	• Better agent utilization, thus allowing more calls to be handled with a given staff level. • Lower abandonment rates - By balancing the load between resources, BSR reduces extremes in wait times across local resources or across an entire network. • In call centers with Expert Agent Selection, the ability to deliver calls to the best qualified or highest revenue generating agents.
Lower costs	• Better agent utilization. • Shorter trunk holding times. • Reductions of ineffective interflows. • Operation over ISDN-BRI trunks and public networks.

Best Service Routing benefits

You can benefit from…	As a result of…
Improved customer satisfaction	• Interflowing calls from centers with a surplus of calls to centers with a surplus of agents. You can achieve uniform service levels across your network. This means that all callers for a given application experience approximately equivalent waiting times. • Shorter wait times. • In call centers with Expert Agent Selection, the ability to deliver calls to the best qualified or highest revenue generating agents. • Robust information forwarding capabilities. Multi-site BSR can forward original service requirements and any caller-entered digits with each call, and can use both QSIG and non-QSIG information transport methods over private or public networks.
Increased performance and more efficient trunk usage	• Less messaging and processing required per call than in traditional LAI scenarios. • Eliminates phantom calls to remote agents. • Intelligent interflows that only route calls to centers with available agents.
BSR's easy configuration	• Simple vector commands. You do not need to learn complex programming languages or design comparison steps. All that you have to do is list the local and remote resources to be considered for calls and instruct the communication server to queue or deliver the call to the best resource on the list.
Improved agent productivity	• Increased efficiency. Improve your service without adding staff, or reduce staff while maintaining your current level of service. Network-wide load balancing means that agents at one location are less likely to sit idle while calls wait in queue at another location. • No call delivery delays. In contrast to approaches that queue calls at all remote centers simultaneously, with BSR there is no delay in delivering a call when an agent becomes available.

Best Service Routing benefits

You can benefit from…	As a result of…
Increased operating flexibility, easier staffing and scheduling	• Larger pool of agents available to take calls in a split/skill. Through its network-wide call distribution and information forwarding, BSR effectively converts distributed locations into a virtual call center. Thus, staffing problems do not need to be solved on a center-by-center basis. BSR can automatically react to staff shortages at one center by routing more calls to other locations. • Automatic management of sudden and unexpected increases in call volume. Large increases in call volume for a single split/skill can be distributed across other splits/skills. Spikes in call volume at a single call center can be distributed across all call centers, provided that sufficient trunk capacity is available between servers.
Improved service levels	• Lower average speed of answer (ASA).

Server and network requirements for BSR

For single-site BSR applications, your Avaya communication server must meet the requirements that are shown below. The requirements for ISDN trunks and LAI do not apply to single-site BSR applications.

To use multi-site BSR applications, all servers involved and the network connecting them must meet all of the requirements that are described in this section.

CAUTION:
To ensure that your network meets the requirements for BSR support, contact your Account Executive about BSR network certification.

This section includes the following topics:

- Server requirements on page 301
- Network requirements on page 301

Server requirements

Your Avaya communication server must meet the requirements shown in the following table to support BSR.

Requirements to use Best Service Routing

Screen	Page	Field	Must be set to…
System-Parameters Customer-Options	2	ISDN-BRI Trunks[1]	Y
		ISDN-PRI Trunks[1 2]	Y
	3	Vectoring (G3V4 Advanced Routing)	Y
		Vectoring (Best Service Routing)	Y
		Lookahead Interflow (LAI)[3]	Y
Feature-Related System Parameters	8	Adjunct CMS Release	R3V6 or higher, or left blank

1. Multi-site BSR operates over both BRI and PRI trunks. ISDN connectivity is only necessary if you want to use multi-site BSR, in which case one or both of these fields must be set to **Y**.
2. ATM trunking and IP trunking can be set up to emulate ISDN PRI. For information on setting this up, see the *Administration for Network Connectivity for Avaya Communication Manager*, and *ATM Installation, Upgrades and Administration using Avaya Communication Manager*.
3. Look-Ahead Interflow is only necessary if you want to use multi-site BSR.

Tip:
If you begin using BSR and then turn it off, you can not set **Vectoring (Best Service Routing)** to **n** until you remove all BSR commands from vectors. If you are using multi-site BSR with Look-Ahead Interflow and want to turn LAI off, you can not set **Lookahead Interflow (LAI)** to **n** until you remove all **`consider location`**, **`reply-best`**, and **`interflow-qpos`** commands from vectors.

Network requirements

To support multi-site BSR, networks must meet both the criteria for LAI call control operation over switched networks (see Look-Ahead Interflow (LAI) on page 273) and the following criteria:

- The network must support end-to-end transport of codeset 0 user data, either as a User-to-User Information Element (UUI IE) or by QSIG Manufacturer Specific Information (MSI IE), in the ISDN SETUP and DISCONNECT messages. For more information, see Determining user information needs on page 196.
- With BSR poll calls, the information is forwarded back in the DISCONNECT message. In this case, the network must support forwarding of UUI in the first call clearing message, while the call is still in the call proceeding state, prior to the active state.
- Private networks can be configured for either QSIG (using MSI packaged in codeset 0 Facility IEs) or non-QSIG (using a codeset 0 UUI IE) transport. Currently, public networks do not support QSIG and user data can only be transported by the UUI IE when supported by the network. Future public network offerings may support QSIG, possibly by Virtual Private Network.
- The switch must support the ISDN country protocol.
- The network byte limit for the user data portion of the user information contents must be large enough to carry the data needed for the customer application.

Note:
Some public network providers may require service activation, fees for user information transport, or both.

BSR, LAI, enhanced information forwarding, and UCID have been tested with several major carriers. To find out if these capabilities work with your carrier, check with your account team for the most current information.

If testing has not been done to verify operation over the public networks that are involved with the preferred specific configuration, use of private ISDN trunking between the nodes should be assumed until successful testing is complete.

Special BSR terminology

Understanding the BSR terms listed below will be helpful as you read through the material in this section. The following list contains terms pertaining to both single-site BSR and multi-site BSR.

adjusted EWT. Expected Wait Time plus a user adjustment set by a **`consider`** command.

agent selection method: The method that the communication server uses to select an agent in a hunt group when more than one agent is available to receive the next call. Possible methods are:

- UCD-MIA
- UCD-LOA
- EAD-MIA
- EAD-LOA

The agent selection method is a property of hunt groups and is set in the Group-Type field on the Hunt Group screen.

Note:
To use an EAD available agent strategy, Expert Agent Selection (EAS) must be enabled.

Alternate Selection on BSR Ties: The BSR feature compares splits or skills using a series of consider steps and selects the one that provides the best service to a call. When that comparison results in a tie (the results are equal in value), the Alternate Selection on BSR Ties determines how BSR chooses which agent, skill, or location to select.

For more information about this feature, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

application: A general term for a system in any call center that handles calls of a particular type. In relation to BSR, any specific implementation of multi-site BSR.

application plan: Used only in multi-site applications, the application plan identifies the remote switches that may be compared in consider series. The plan also specifies the information that is used to contact each switch and to interflow calls to it.

best: Includes the following conditions

- No agents available - When no agents are available in any of the specified splits/skills, the best resource is the one with the lowest adjusted EWT.
- Agent available in one resource - When an agent is available in one and only one of the splits/skills that are specified in a consider series, that agent is the best and the call is delivered to that agent. If the **BSR Available Agent Strategy** field is set to **1st-found**, BSR ignores all subsequent steps in the consider series. If any other available agent strategy is used, all remaining resources are still considered before the call is delivered.
- Agents available in two or more resources - When agents are available in two or more splits/skills, the best agent is the one that best meets the criteria that are specified in the **BSR Available Agent Strategy** field. For example, if the available agent strategy is UCD-MIA, the best agent out of those available will be the agent with the longest idle time.

Best Service Routing (BSR): A feature that is based on call vectoring and routes ACD calls to the resource that is best able to service each call. BSR can be used on a single switch, or it can be used to integrate resources across a network of switches.

BSR available agent strategy: A field that appears on the VDN screen when either version of BSR is enabled. The entry in this field is a property of the VDN and its assigned vector. Possible entries are:

- 1st-found
- UCD-MIA
- UCD-LOA
- EAD-MIA

- EAD-LOA

When the VDN is the active VDN for a call, as determined by VDN Override, this field determines how BSR commands in the vector identify the best split/skill when several have available agents.

consider series: `consider` commands are typically written in a set of two or more. This set of `consider` commands is called a consider series. A consider series in a status poll vector might have just one consider step.

consider sequence: A consider sequence is a consider series plus a `queue-to best, check-best, or reply-best` step.

Expected Wait Time (EWT): Expected Wait Time is an estimate of how long a call in the queue will have to wait before it is connected to an agent.

Intelligent polling: An automatic feature of BSR that significantly reduces the number of status polls that are executed. When a remote location cannot be the best resource at a given moment in time, the intelligent polling feature temporarily suppresses polls to that location.

interflow: The process of routing an incoming call to an external switch without answering it at the origin switch.

poll suppression: A component of BSR intelligent polling that eliminates wasteful polling of remote locations which have returned poor adjusted EWTs.

resources: An agent, split, skill, or location

status poll: A call that is placed by a `consider location` vector command to obtain status data from a remote location in a multi-site BSR application.

Single-site BSR

This section includes the following topics:

- About single-site BSR on page 305
- Command set - single site BSR on page 305
- How BSR determines the best resource on page 307
- Example of basic single-site BSR on page 310
- User adjustments in single-site BSR on page 312
- Example of single-site BSR with adjustments on page 314

About single-site BSR

Single-site BSR is a simple, logical extension of call vectoring. Like any other vector, vectors with BSR commands are assigned to one or more VDNs. Using new vector commands and command elements, you tell the communication server to compare, or consider, specific splits/skills for each call that is processed in that particular vector. Throughout the comparison, the server can remember which resource is the best based on how you define best. BSR vectors can deliver a call to the first available agent found, or they can consider all of the specified resources and deliver the call to the best split/skill. If no agents are available in any split/skill, the call is queued to the split/skill with the shortest adjusted EWT.

Command set - single site BSR

The following table shows the screens, the vectors, and the vector commands and command elements that are used in single-site BSR. The following table shows the vector commands and command elements used in single-site BSR applications.

Vector commands and usage for single-site BSR

Commands and command elements		Use this…
Screens	Vector Directory Number screen	To link a VDN to a BSR vector. To set the agent selection strategy that will be used for all calls to that VDN.
	Call Vector screen	To confirm that BSR is administered. To write vectors that use BSR commands.
Commands	**consider split/skill**	To obtain the Expected Wait Time or agent data that is needed to identify the best local resource. One **consider** step must be written for each split/skill that you want to check. Since the **consider** command is designed to compare two or more resources, **consider** commands are typically written in a series of two or more with the sequence terminating in a **queue-to best** vector step. This set of **consider** commands and a **queue-to best** step is called a consider sequence.
	queue-to	With the **best** keyword to queue calls to the best resource that is identified by the consider sequence.
	check	With the **best** keyword to queue calls to the best resource that is identified by the consider sequence if the resource meets certain conditions.

Vector commands and usage for single-site BSR (continued)

Commands and command elements		Use this…
Key word	**best**	Use the **best** keyword in **queue-to**, **check**, and **goto** commands that refer to the resource that is identified as best by a series of **consider** steps
Conditional	**wait-improved**	To prevent calls from being queued to an additional split/skill when the reduction in Expected Wait Time is not enough to be useful. *Wait improved* means that a call's EWT must be improved by a specific amount, specified in seconds, over its current EWT or the communication server does not queue the call to the additional split/skill.
User adjustment	**adjust-by**	To specify your preferences for the splits/skills that might handle the calls for a particular application, reflecting factors such as agent expertise or reducing calls to a backup split/skill. When a vector considers a local resource you can make the selection of that split/skill less desirable. The higher the setting, the less chance that resource will be selected over another with a lower setting (for example, set to 30 makes that choice 30% less desirable). With EWT returned, the setting increases the returned expected wait time for comparison with other returned EWTs. As a result, this split/skill is less likely to service the call unless its EWT is significantly less than that of any other available split/skill. Optionally, the adjust-by setting applies in the available agent case. If you are using the UCD-MIA or EAD-MIA available agent strategy, the setting decreases the returned agent idle time, making the agent appear less idle (busier). If you are using the UCD-LOA or EAD-LOA available agent strategy, the setting increases the returned agent occupancy, making the agent appear busier. In either case with EAD, the MIA or the LOA is used as a tie breaker if more than one site has an agent available with the same highest skill level.

How BSR determines the best resource

BSR determines the best resource to service a call by examining one or all of the following variables:

- The EWT of the resource
- Any user adjustments
- The availability of agents
- The selection strategy for the active VDN

Note:
The BSR available agent strategy that applies to a given call is the strategy that is assigned to the active VDN for that call, as determined by VDN override.

This section includes the following topics:

- Call surplus situations on page 307
- Agent surplus situations on page 308
- Agent selection adjustments on page 309

Call surplus situations

Every BSR application compares a set of predetermined resources (splits/skills) and selects the best resource to service the call.

In a call surplus situation when no agents are available, the best resource is the split/skill with the lowest Expected Wait Time (EWT). For purposes of calculating the best resource in a call surplus situation, BSR allows you to adjust the EWT figure for any split/skill. The actual EWT for calls in queue is not changed. Only the figure used in the calculations performed by the BSR feature is changed. You do not have to enter adjustments, but the ability to adjust the EWT for splits/skills allows you to program preferences in vectors. Because of agent expertise, for example, or the availability or cost of tie trunks, you might prefer that some resources do not service a call unless doing so significantly decreases the time in queue for the call.

It is possible for you to make adjustments to agent availability using the **`consider`** step. For more information, see Agent selection adjustments on page 309.

Agent surplus situations

In an agent surplus situation when one or more agents are available to take incoming calls, BSR delivers a new call according to the **BSR Available Agent Strategy** field that is specified on the VDN screen. The best resource is the split/skill that meets the criteria that are defined by the strategy that was administered for that VDN. BSR can use any of the five strategies shown in the following table to select an agent when agents are available.

BSR available agent strategies

If BSR Available Agent Strategy is set to…	The call will be delivered to…
1st-found	The first available agent. BSR will not consider any other resources as soon as it finds an available agent.
ucd-mia	The resource with an agent who has been idle for the longest amount of time. BSR compares all of the splits/skills that are specified in the vector before delivering the call.
ead-mia	The resource with an agent who has the highest skill level that is relevant to the call and who has been idle the longest. BSR compares all of the splits/skills that are specified in the vector before delivering the call.
ucd-loa	The resource with the least-occupied agent. BSR compares all of the splits/skills that are specified in the vector before delivering the call.
ead-loa	The resource with an agent who has the highest skill level that is relevant to the call and who is the least occupied. BSR compares all of the splits/skills that are specified in the vector before delivering the call.

For more information on LOA, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* , or *Avaya Business Advocate User Guide*. LOA is available with the Contact Center Elite package.

When agents are available in one or more of the specified resources, BSR does not consider resources (local or remote) that return an EWT (call queue/call surplus situation) in selecting the best place to send the call.

Note:
The **BSR Available Agent Strategy** that is assigned to a VDN should match the agent selection method that is used in the splits/skills considered by a BSR application.

Agent selection adjustments

An option has been provided to have the BSR adjust-by value apply in the agent surplus (agents available) situation. This adjustment provides the ability to use the **`consider`** step adjustment value to prioritize (handicap) agent resources when agents are available.

When the adjustment is used, the **`consider`** step uses the following syntax:

consider split/location adjust-by ***`x`***

The server applies the agent adjustment in the same manner as the calls in queue/call surplus (lowest EWT) situation.

To select an adjustment, think in terms of reducing the importance of a resource/site and in relative percentage — the higher the adjustment, the less desirable it is to pick that agent/site. So, if ***`x`*** = 30, then the agent/site is 30% less desirable.

The available agent adjustment applies to the UCD-MIA, UCD-LOA, EAD-MIA, and EAD-LOA call distribution methods. For the most idle agent distribution methods, the adjust-by lowers the idle time value returned by the agent/site. For the least occupied agent distribution methods, the adjust-by raises the returned occupancy level of the agent/site. In either case, with EAD, the MIA or LOA is used as a tie breaker if more than one site has an agent available with the same highest skill level.

The same adjust-by value in the **`consider`** step applies to both agent surplus and call surplus situations.

Example of basic single-site BSR

This example shows the simplest use of BSR. The central element of all single-site and multi-site BSR is a VDN/vector pair. The vector contains the commands that actually process the call, but the active VDN for the call contains information that is used by some vector steps. For single-site BSR, the active VDN for a call sets the available agent strategy that is used by the vector.

Single-site BSR example VDN screen

```
change vdn xxxxx                                                  page 1 of 3
              VECTOR DIRECTORY NUMBER

                              Extension: 5000
                                   Name: Single-site BSR
                         Vector Number: 234
                     Attendant Vectoring? n
                      Meet-me Conference? n
                      Allow VDN Override? n
                                    COR: 59
                                     TN: 1
                               Measured: internal
       Acceptable Service Level (sec): 20
              Service Objective (sec):
        VDN of Origin Annc. Extension:
                              1st Skill:
                              2nd Skill:
                              3rd Skill:
```

```
change vdn xxxxx                                                  page 2 of 3
              VECTOR DIRECTORY NUMBER

                            Audix Name:

                     Return Destination:
                 VDN Timed ACW Interval:
                        BSR Application:31
         BSR Available Agent Strategy: 1st-found
```

In the example Vector Directory Number screen shown above, the **BSR Available Agent Strategy** field is set to **1st-found**. If vector 234 uses BSR commands, as soon as a **consider** step locates a resource with an available agent any subsequent **consider** steps are skipped and the call is delivered to that resource. Resources that are specified in any subsequent **consider** commands are not checked. If no split has an available agent, the call is queued to the split with the lowest adjusted EWT.

If the **Allow VDN Override?** is set to **n** and a second VDN and vector are used to process this call, the 1st-found strategy specified in VDN 5000 will still be used.

In the preceding example, Vector Directory Number 5000 is associated with vector 234, which is shown below. In this example, vector 234 compares two splits. No adjustment is assigned to either resource, indicating that both splits are equally suited to service calls since neither is preferred to the other. In reality, such a vector would probably have additional steps after step 4, such as **announcement** or **wait-time** commands. These steps are omitted in this example for purposes of clarity.

Single-site BSR example vector

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

Notice that the **consider** commands follow each other in unbroken sequence and that the **queue-to best** command immediately follows the last **consider** command. This structure is called a *consider series*, and it is recommended that you typically write such series in uninterrupted order. A few commands, such as the **goto** command, which cause little if any delay in the execution of the **consider** steps, may be used. In general, however, do not put other commands between **consider** steps, or between a **consider** step and a **queue-to best** step. Even if BSR still works in that situation, you might seriously impair the performance of the vector.

Consider commands collect and compare information. When a call is processed in the vector above, the first consider step collects and temporarily saves the following information about split 1:

- The fact that split 1 is a local split
- The queue priority that is specified in the **consider** step
- The user adjustment that is specified in the **consider** step
- The split's
  - Split number
  - Expected Wait Time

If EWT=0, which indicates that one or more agents are available, the step also collects all of the agent information that might be needed by the BSR available agent strategy. This includes:

- Agent Idle Time (AIT)
- Agent Occupancy (AOC)
- The skill level of the agent in the split/skill who will receive the next call

In the example shown above, neither split has an available agent when the consider series executes. If one did, the call would be delivered to that split by the **queue-to best** step. Since there are no available agents in either split, the complete set of saved data now defines the best resource—for the moment. The second consider step collects the same data and compares it to the current best data. For this example, assume that the EWT for split 1 is 40 seconds and the

EWT for split 2 is 20 seconds. When the second **consider** step executes, its data will replace the best data from step 1 because its adjusted EWT is lower. The best data is essentially a placeholder. When a **queue-to best** step executes, it reads the data that is saved as the best at that moment and queues the call to that split. In this case, the best data was collected from split 2, so the call is queued to split 2 at the specified priority.

What if there are available agents in both splits?

Since the **BSR Available Agent Strategy** in this example is 1st-found, the consider series will skip any **consider** steps after step 2 and the **queue-to best** step will deliver the call to split 1, which is the first split/skill with an available agent that is found by the vector.

In any BSR vector, the order of the **consider** steps should reflect your preferences for the resources to be considered. Put the step that considers the most preferred split/skill first, the step for your second preference second, and so forth in the consider series.

What if there are several available agents in split 1? Which agent receives the call?

When more than one agent is available in a split, the BSR **consider** command collects agent data only for the agent who will receive the next call to that split. This agent is identified according to the agent selection method that is specified in the **Group-Type** field on the Hunt Group screen.

Note:
For greatest efficiency, the agent selection method used in the splits/skills considered by a BSR vector should match the **BSR Available Agent Strategy** that is assigned to the active VDN.

User adjustments in single-site BSR

You may have preferences as to which splits/skills should answer certain types of calls. In both single-site BSR and multi-site BSR, the **adjust-by** portion of the **consider** command makes it possible for you to program these preferences into your vectors.

You can assign a value of 0 to 100 in user adjustments. The units of this value are supplied by the server depending on the conditions whenever that **consider** step executes. For example, in the command **consider split 1 pri h adjust-by 20**, the server interprets **adjust-by 20** to mean *add 20% to the EWT, but add at least 20 seconds*.

Note:
If the user adjustment were defined as a number of seconds, BSR would not be efficient when EWT was high. If the user adjustment were defined as a percentage, BSR would not be efficient when EWT was low. Such efficiencies, while always important, become critical in multi-site BSR applications where issues of trunk cost and capacity are involved.

For Expected Wait Times of 1 to 100 seconds, an adjustment of 20 will therefore add 20 seconds. Above 100 seconds, the same adjustment will add 20% to the EWT for the split/skill that is specified in the **consider** step. The following table shows the results of applying a constant adjustment to a range of Expected Wait Times.

User adjustments in BSR

EWT of resource (seconds)	User adjustment	Adjustment applied by the server (seconds)	Adjusted EWT used to select resource
10	20	20	30
60		20	80
120		24	144
300		60	360

Example of single-site BSR with adjustments

The following example shows a more complex implementation of single-site BSR. Four skills in an Expert Agent Selection environment are compared. The Expected Wait Time (EWT) for some skills is adjusted to reflect the administrator's preferences

Single-site BSR example VDN screen

```
change vdn xxxxx                                              page 1 of 3
              VECTOR DIRECTORY NUMBER

                              Extension: 5001
                                   Name: Single-site BSR
                         Vector Number: 11
                     Attendant Vectoring? n
                      Meet-me Conference? n
                      Allow VDN Override? n
                                    COR: 59
                                     TN: 1
                               Measured: internal
       Acceptable Service Level (sec): 20
              Service Objective (sec):
        VDN of Origin Annc. Extension: 501
                              1st Skill:
                              2nd Skill:
                              3rd Skill:
```

```
change vdn xxxxx                                              page 2 of 3
              VECTOR DIRECTORY NUMBER

                             Audix Name:

                  Return Destination:
               VDN Timed ACW Interval:
                      BSR Application:19
         BSR Available Agent Strategy: EAD-MIA
```

In the example shown above, the **BSR Available Agent Strategy** field is set to **EAD-MIA**. If vector 11 uses BSR commands, calls are not automatically delivered to the first resource with an available agent that is found. All **consider** steps in vector 11 are executed and one of the following things happens:

If …	Then…
No skill has an available agent	The call queues to the skill with the lowest adjusted EWT.
Only one skill has an available agent	The call is delivered to that skill.

If …	Then…
Two or more skills have available agents	The call is delivered to the skill with the most expert agent.
Two or more skills have available agents with the same skill level	The call is delivered to whichever of these agents has been idle the longest.

Also note that **Allow VDN Override?** is set to **n**. If a second VDN and vector are used to process this call, the EAD-MIA strategy that is specified in VDN 5001 is used. If **Allow VDN Override?** is set to **y** and vector 11 routes some calls to another VDN, the subsequent VDN's available agent strategy governs the operation of **consider** steps in its vector.

The following example vector 11, which compares four skills.

Single-site BSR example vector

```
1. wait-time  0 secs hearing ringback
2. consider skill  1 pri l adjust-by 0
3. consider skill  2 pri l adjust-by 30
4. consider skill 11 pri l adjust-by 30
5. consider skill 12 pri l adjust-by 30
6. queue-to best
7. wait-time 10 secs hearing ringback
8. announcement 1001
9. wait-time 30 secs hearing music
10. goto step 8 unconditionally
```

For this example, assume that the Expected Wait Times of the four skills are 95, 60, 180, and 50 seconds, respectively. Notice that all **consider** steps except the first adjust the EWT returned by the specified skill. Skill 1 is the preferred skill to handle calls to VDN 5001, so its EWT is not adjusted. Skills 2, 11, and 12 can handle this call type, but they are not preferred. The adjustment of 30 means that, in call surplus situations, these skills will not handle calls to VDN 5001 unless their EWT is at least 30 seconds better than the EWT in skill 1.

The following table shows the adjustments that would be applied to each skill given its EWT and the user adjustment specified in the **consider** step. The last column shows the adjusted EWT the server will use to select a skill for the call.

User Adjustments

Skill number	User adjustment in the consider step	Actual EWT (seconds)	Adjustment applied by the server (seconds)	Adjusted EWT used in BSR calculations (seconds)
1	0	95	0	95
2	30	60	30	90

User Adjustments (continued)

Skill number	User adjustment in the consider step	Actual EWT (seconds)	Adjustment applied by the server (seconds)	Adjusted EWT used in BSR calculations (seconds)
11	30	180	54	234
12	30	50	30	80

Since the available agent strategy is not 1st-found, all four **consider** steps are executed each time that the vector processes a call. In this example, there are no available agents in any of the skills. In fact, EWT is high enough in the first three skills for the server to queue the call to skill 12.

When the **queue-to-best** step executes, the data in the best data placeholder is the data from skill 12 and so the call is queued to that skill. From this point on, if the call is not answered during the execution of step 7, a common vector loop regularly repeats an announcement for the caller while he or she waits in the queue.

User adjustments also apply to available agent situations (with a strategy other than first found) in a manner that is similar to EWT. For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

What if there is an available agent in one skill? Will user adjustments be applied?

Since the **BSR Available Agent Strategy** in this example is EAD-MIA, the entire consider series will always be executed to check all of the skills for available agents. If only one skill has available agents, the call is delivered to that skill and user adjustments are not applied.

What if there are available agents in two skills? Which skill gets the call? Will user adjustments be applied?

Since the **BSR Available Agent Strategy** for VDN 5001 (the active VDN) is EAD-MIA, the call is delivered to the skill with the most expert agent. If there are available agents in both skills with the same skill level, their user adjusted idle times are compared and the call goes to the skill with the agent who has the longest adjusted idle time.

If a split/skill has more than one available agent, remember that it is the split/skill's agent selection method that determines which agent's data is used in BSR selection of the best resource.

What if no agents are staffed in a skill? Will the server recognize this?

Yes. Under any of the following conditions, the EWT returned from a split/skill is infinite:

- No agents logged in
- No queue slots available
- All agents in AUX work mode

The server logs a vector event and goes to the next vector step without changing the data in the best placeholder. A resource with an infinite EWT is never selected as the best resource.

Can VDN skills be used in consider steps?

Yes. For example, **`consider skill 1st [2nd, 3rd] pri m adjust-by 0`** will collect data on the 1st [2nd, 3rd] skill, as defined for the active VDN.

Planning and administering single-site BSR

This section presents information that is specific to BSR. Follow existing procedures to add or change other properties of VDNs and vectors that are not discussed in this section.

First, confirm that your server meets the requirements for single-site BSR if you have not already done so. For a listing of requirements, see Server and network requirements for BSR.

This section includes the following topics:

- Planning on page 317
- Administration on page 318

Planning

To work more efficiently, you may want to record goals, VDN extensions, vector numbers, and other information on paper before you begin your administration session. To do this, complete the following:

1. Select the group of callers for which you want to use single-site BSR, and identify the VDNs and vectors that support this group.
2. Define your goals. For example, your goals in using BSR might be faster average speed of answer, or better service by routing calls to the most qualified agents.

   Different VDNs or vectors may have different goals.

3. Decide which agent selection strategy that you will assign to each VDN in order to best achieve the goals that are relevant to that VDN.
4. Decide whether you will allow VDN Override for each of the VDNs that are identified.

Administration

Use this procedure to administer single-site BSR, complete the following:

1. To go to the Vector Directory Number screen for the first VDN you identified in step 1 of Planning, type **add vdn *xxxxx*** or **change vdn *xxxxx*** at the command line prompt and press Enter, where ***xxxxx*** is a valid VDN extension as defined in the system dial plan.
2. In the **Allow VDN Override?** field, enter **y** or **n**. If the call is directed to another VDN during vector processing:
   - **y** allows the settings on the subsequent VDN, including its **BSR Available Agent Strategy**, to replace the settings on this VDN.
   - **n** allows the settings on this VDN, including its **BSR Available Agent Strategy**, to replace, or override, the settings on the subsequent VDN.
3. In the **BSR Available Agent Strategy** field, enter the identifier for the agent selection method that you want this VDN to use.

   When this VDN is the active VDN for a vector that uses BSR, the available agent strategy determines how calls are directed when one or more of the specified resources have available agents. If there is only one split/skill with available agents, calls are delivered to that resource.

If you enter…	Consider series in vectors will select the resource with…
1st-found	The first available agent. BSR does not consider any other resources as soon as it finds an available agent.
ucd-mia	The agent who has been idle the longest. BSR will compare all of the splits/skills that are specified in the vector before delivering the call.
ead-mia	The agent with the highest skill level who has been idle the longest. BSR compares all of the splits/skills that are specified in the vector before delivering the call.

If you enter…	Consider series in vectors will select the resource with…
ucd-loa	The least-occupied agent. BSR compares all of the splits/skills that are specified in the vector before delivering the call.
ead-loa	The agent with the highest skill level who is the least occupied. BSR compares all of the splits/skills that are specified in the vector before delivering the call.

4. Press **Enter** to save your changes.

You are now ready to write or modify the vector that is assigned to this VDN. For tips on using BSR commands in vectors, see Tips for writing BSR vectors on page 354.

Troubleshooting for single-site BSR

You should regularly execute a **`display events`** command for the appropriate vectors, especially if you have just implemented a new BSR application. Vector events will identify and indicate the source of common malfunctions and administration errors.

For a list of BSR vector events and definitions, see Troubleshooting vectors on page 689.

Note:
Only the most recent events are displayed when a **`display events`** command is executed. For this reason, you should periodically display vector events to help quickly identify problems.

To verify that your BSR vectors are operating as intended, use a **`list trace vdn`** or **`list trace vec`** command to observe processing of an individual call. For more information, see Clearing events on page 726.

Note:
The **`list trace vdn`** and **`list trace vec`** commands are blocked if the Tenant Partitioning feature is enabled.

Multi-site BSR

Multi-site BSR extends all of the capabilities of single-site BSR across a network of communication servers. Multi-site BSR compares local splits/skills and remote splits/skills, and route calls to the resource that provides the best service. Multi-site BSR has special features that work to ensure efficient use of processor power and network resources in your BSR applications.

This section includes the following topics:

- Multi-site BSR command set on page 320
- Multi-site BSR applications on page 323
- Example of multi-site BSR on page 326
- BSR available agent strategies on page 331
- More on status poll and interflow vectors on page 331
- User adjustments in multi-site BSR on page 332
- Example of multi-site BSR with limited trunking on page 333
- Example of multi-site BSR with slow networks on page 339
- Example for handling excessive wait times on page 342
- Selecting or administering application plans on page 343
- Administering the BSR Application Plan on page 343

Multi-site BSR command set

The following table shows the screens, the vectors, and special vector commands and command elements that you use to administer multi-site BSR applications. The table also briefly describes the purpose of each component.

Vector commands and usage for multi-site BSR

Screens	
Best Service Routing Application Plan screen	• To define the group of remote sites that will be polled by a specific application. • To assign a unique name and number to each application. • To assign routing numbers for the status poll and interflow VDNs.

Vector commands and usage for multi-site BSR (continued)

Vector Directory Number screen	• To link a VDN to a BSR application by its application number. • To link the VDN to a BSR vector. • To set the agent selection strategy that will be used for all calls to that VDN.
Call Vector screen	• To confirm that BSR is administered and to program the vector steps for BSR.
ISDN Trunk screens	• To tell the communication server whether to forward user information by Shared UUI or QSIG MSI.
List Best Service Routing Applications screen	• To display a list of all the BSR applications by name and number.
System Capacity	• To monitor the number of BSR application-location pairs that are assigned in your system.
VDNs and Vectors	
Primary VDN (the active VDN for the call at the origin, as determined by VDN override)	• To define the application plan and available agent strategy that are used by the vector that is assigned to this VDN.
Primary vector	• To control call processing at the original server and compare local and remote resources.
Status poll VDN/ vector	• To respond to status poll calls from another server. The status poll vector considers a set of local splits/skills and returns data on the best resource to the original server.
Interflow VDN/ vector	• To accept BSR calls from another server and queue them to the best of the local resources considered.
Commands	
`consider split/skill`	• To obtain the Expected Wait Time or agent data that is needed to identify the best local resource. One **`consider`** step must be written for each split/skill that you want to check. Since the **`consider`** command is designed to compare two or more resources, **`consider`** commands are typically written in a series of two or more with the sequence terminating in a **`queue-to best`** vector step. This set of **`consider`** commands and a **`queue-to best`** step is called a consider sequence.

Vector commands and usage for multi-site BSR (continued)

consider location	• To obtain the Expected Wait Time or agent data that is needed to identify the best resource at a remote server. One **consider** step must be written for each location that you want to check. Routing information is obtained from the BSR Application plan for the active VDN.
reply-best	• To return data to another server in response to a status poll
queue-to	• With the **best** keyword to queue calls to the best resource that is identified by the consider sequence.
check	• With the **best** keyword to queue calls to the best resource that is identified by the consider sequence if the resource meets certain conditions.
Key word	
best	• In **queue-to**, **check**, and **goto** commands that refer to the resource identified as best by a series of **consider** steps
Conditional	
wait-improved	• To prevent calls from being queued to an additional split/skill—local or remote—when the reduction in Expected Wait Time is not enough to be useful. *Wait improved* means that a call's EWT must be improved by a specific amount, which is a figure that you specify in seconds, over its current EWT or the server will not queue it to the additional split/skill.
User adjustment	
adjust-by	• To control long-distance costs and limit trunk usage, reflecting factors such as availability of the trunks or agent expertise at remote locations. When a vector polls a local or remote resource, you can make the selection of that site less desirable. The higher the setting, the less chance that resource will be selected over another with a lower setting. With EWT returned, the setting increases the returned expected wait time for comparison with other returned EWTs. Optionally, the adjust-by setting applies in the available agent case. If you are using the UCD-MIA or EAD-MIA available agent strategy, the setting decreases the returned agent idle time, making the agent appear less idle (busier). If you are using the UCD-LOA or EAD-LOA available agent strategy, the setting increases the returned agent occupancy, making the agent appear more occupied (busier). In either case with EAD, the MIA or the LOA is used as a tie breaker if more than one site has an agent available with the same highest skill level.

Multi-site BSR applications

You can implement BSR at a single location solely by using the BSR commands in vectors. Using BSR across a network is more complex and requires additional administration.

A series of **`consider location`** steps in a multi-site BSR vector contacts one or more remote locations. You need to define these locations, tell the server how to contact each one, and set up VDNs and vectors to handle communications between the origin server and the remote (or receiving) servers. The BSR application should support some larger application in your call center that handles calls of a particular type.

Note:
Any combination of split/skill numbers, VDN numbers, and vector numbers can be used to support a single customer application or call type across a network. For clarity and simplicity, Avaya recommends that the BSR Application Plan number and the location numbers for a given application be the same on all servers.

You also need to set up ISDN trunk groups, set the parameters for information forwarding (UUI Transport), and administer numbering plans and AAR/ARS tables.

Multi-site BSR starts with the active VDN for a call, as determined by VDN override. If you want any specific VDN/vector pair to interflow calls using multi-site BSR, you must create a specific application for it. A multi-site application must contain the elements shown in the following table.

Required elements of a multi-site BSR application

A BSR application consists of…	Which serves this purpose…
The Primary VDN	The Primary VDN is the active VDN for a call at the origin server, as defined by VDN override. Therefore, the Primary VDN in a BSR application does not have to be the VDN that originally received the incoming call. The primary VDN links its assigned vector to a BSR application plan and sets the **BSR Available Agent Strategy**.
The Primary vector that handles the incoming call on the origin server	The Primary vector contacts the specified remote servers, collects information, compares the information, and delivers or queues the call to the resource that is likely to provide the best service.
An application plan	The application plan identifies the remote servers that you can compare and specifies the information that will be used to contact each server and to route calls to it.

Required elements of a multi-site BSR application *(continued)*

A BSR application consists of…	Which serves this purpose…
Two VDN/vector pairs on each remote server: ● Status poll VDN/vector ● Interflow VDN/vector	**Status poll VDN/vector** The status poll vector compares splits at its location and replies to the origin server with information on the best of these splits. Each remote server in a given application must have a dedicated status poll VDN/ vector.
	Interflow VDN/vector When a given remote server is the best available, the origin server interflows the call to this VDN/vector on the remote server. Each remote server in a given application has to have a dedicated interflow VDN/ vector. The steps in this vector deliver or queue the call, as appropriate, to the best resource that is found by the status poll vector.

To create a multi-site BSR application, you start by creating an application plan on the origin server.

Note:
Remember that the terms *local*, *origin*, and *remote* are relative terms. In most networks that use multi-site BSR, every server can interflow calls to other servers and receive interflowed calls from other servers. Therefore, every server in the network may have all the elements described above. For clarity in the following discussions, *local* or *origin* means a server that is considering or might consider whether to interflow a call. *Remote* means any server that is polled or might be polled by this first server.

Application plans

The application plan identifies the remote servers that you can compare and specifies the information that is used to contact each server and to route calls to it.

The plan for each application is identified by the application number and a name. It specifies the remote servers that might be polled by the application and identifies each with a number called the location number. The plan also specifies the numbers for the status poll and interflow VDNs for each remote server. Whatever you would dial to reach these VDNs is what should be entered in these fields: full length numbers as well as AAR, ARS, UDP, or public network numbers will work.

You create application plans on the Best Service Routing Application screen. A plan for an application with three remote servers might look like the following example.

Sample multi-site BSR Application Plan

```
BEST SERVICE ROUTING APPLICATION PLAN

 Number: 15   Name: Customer Service  Maximum Suppression Time: 60   Lock? y

NumLocation NameSwitch Node          Status Poll  VDN Interflow VDN  Net Redir?
1 New Jersey    32084015          84115                 n
2 Denver      18  913031234015    913031234115          n
4 New York    12345912121234015    912121234115         n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
```

The maximum number of application plans may vary depending on your Avaya Communication Manager software release and platform. For more information, see *System Capacities Table for Avaya Communication Manager on Avaya Media Servers*. You can find the latest capacity tables from the Avaya support Website at:

http://www.avayadocs.com

By entering the application number from this plan on a VDN screen, you can link a given VDN on your local server to this list of locations. This VDN becomes the primary VDN for the application. For example, if the primary vector contains instructions to consider locations 1 and 2, the server places a status poll call to the status poll VDN at the New Jersey and Denver servers and compares the results. If location 2 is better than either location 1 or any splits that are considered on the originating server, the call will be interflowed to the interflow VDN that is specified in the plan for location 2.

Example of multi-site BSR

This section includes the following topics:

- Multi-site BSR on page 326
- Primary vector on page 328
- Status poll vector on page 329
- Interflow vector on page 330
- BSR call interflow with SIP on page 330
- What happens to the call if the interflow attempt fails? on page 330
- Can I adjust the AIT or AOC returned by an available resource? on page 330

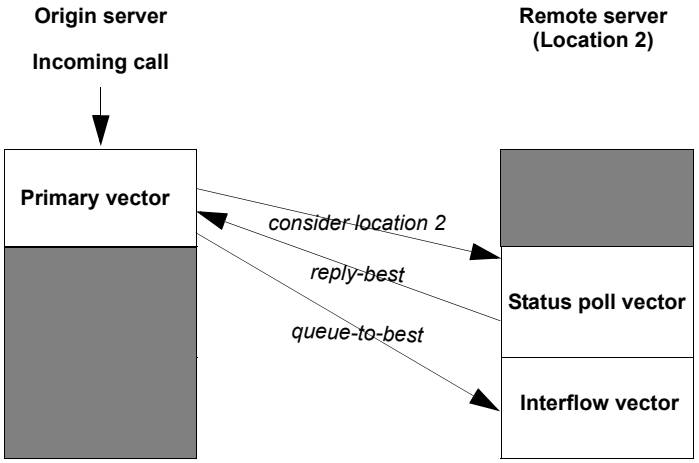
Multi-site BSR

To see how the basic elements of multi-site BSR work, consider a simple application in a two-server network. Multi-site BSR compares local and remote splits/skills and queues calls to the resource that provides the best service. Remember that each BSR application has two main parts:

- An application plan. This plan identifies the remote servers that you want to compare.
- A set of three VDN/vector pairs:
  - The primary VDN/vector. This vector on the origin server contacts the specified remote servers, collects information, compares the information, and routes the call to the server that is likely to provide the best service.
  - The status poll VDN/vector. The status poll vector on the remote server compares resources on that server and replies to the origin server with information on the best of these. Each remote server in a given application must have a dedicated status poll vector.
  - The interflow VDN/vector. When a given remote server is the best available, the origin server interflows the call to this vector on the remote server. Each remote server in a given application has to have a dedicated interflow vector.

The general operational scheme for multi-site BSR is shown in the following figure.

BSR example of origin and remote communication servers



The following example shows the primary VDN using a multi-site BSR application.

BSR example primary VDN

```
change vdn xxxxx                                              page 1 of 3
              VECTOR DIRECTORY NUMBER

                            Extension: 52222
                                 Name: Multi-site BSR
                       Vector Number: 222
                   Attendant Vectoring? n
                    Meet-me Conference? n
                    Allow VDN Override? n
                                  COR: 59
                                   TN: 1
                             Measured: internal
       Acceptable Service Level (sec): 20
              Service Objective (sec):
        VDN of Origin Annc. Extension:
                            1st Skill:
                            2nd Skill:
                            3rd Skill:
```

```
change vdn xxxxx                                              page 2 of 3
              VECTOR DIRECTORY NUMBER

                           Audix Name:

                  Return Destination:
               VDN Timed ACW Interval:
                      BSR Application:15
         BSR Available Agent Strategy: UCD-MIA
```

In the example shown above for VDN 52222, the entry in the **BSR Application** field links this VDN to BSR Application Plan 15. Also note the UCD-MIA entry in the **BSR Available Agent Strategy** field. If vector 222 uses BSR commands, calls are not automatically delivered to the first resource found with an available agent. All **consider** steps in vector 222 are executed, and one of the following things happens:

If:	Then:
There is no available agent in the local or the remote splits	The call will be queued to the split with the lowest adjusted EWT.
Only one split has an available agent	The call will be delivered to that split.
Two or more splits have available agents	The call will be delivered to the split with the most idle agent.

Also note that **Allow VDN Override?** is set to **n**. If a second VDN and vector are used to process this call, the UCD-MIA strategy and the application plan that are specified in VDN 52222 are used.

Application plan 15 (which is shown in on page 325) identifies the remote server and provides the digit strings to dial into the VDNs for both the status poll vector and the interflow vector.

Primary vector

When a call arrives at the origin server, it is processed by the primary vector. This vector begins the BSR process by considering the resources that are specified. The following example shows a primary vector used for that purpose.

BSR example of primary vector on origin communication server

```
1. wait time 0 secs hearing ringback
2. consider split1 pri m adjust-by 0
3. consider location 2 adjust-by 30
4. queue-to-best
```

In this example, the **consider** commands in steps 2 and 3 collect information to compare local split 1 with one or more splits at location 2. (Location 2 is the Denver server identified on the BSR Application Plan screen.) Step 4 queues the call to the best split that is found. As in single-site BSR, the **adjust-by** portion of the **consider** command allows you to set preferences for each resource, whether the resource is a remote location or a split/skill on the origin server. In multi-site BSR, this user adjustment enables you to control the frequency of interflows by adjusting the EWT that is returned by a particular resource on a remote server. In this example, the communication server administrator has chosen to adjust the EWT value for location 2 by 30.

Status poll vector

To collect information from the remote server, the command **consider location 2 adjust-by 30** in the primary vector places an ISDN call, known as a status poll, to the status poll vector on the server at location 2. The following example shows a status poll vector on the remote server used for that purpose.

BSR example of status poll vector on remote communication server

```
1. consider split2 pri m adjust-by 0
2. consider split 11pri m adjust-by 0
3. reply-best
```

The status poll only obtains information and returns it to the origin server; the call is not connected to the status poll VDN.

This vector compares splits 2 and 11, identifies the better of the two, and sends this information back to server 1 with the **reply-best** command. Notice that the **adjust-by** command could be used on the remote server to adjust the EWT that is returned by either of the splits. When EWT adjustments are applied at both the origin and remote servers, the two adjustments are added at the origin server. For more detail on user adjustments in multi-site applications, see User adjustments in multi-site BSR on page 332.

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 that is returned to server 1 contains the information from the best split considered at location 2. Once the remote server returns the necessary information, the consider series in the primary vector on server 1 can continue at the next vector step.

CAUTION:
It is recommended that status poll vectors not be used to poll other servers. Status poll vectors should only consider resources on the server where the vector resides. Status poll vectors must always end with a reply-best step. A busy or disconnect should never be used.

Note:
Multi-site BSR includes mechanisms that automatically limit the number of status poll calls that are placed over the network when such calls are unlikely to yield better service for the caller. For a detailed explanation of these mechanisms, see Advanced multi-site routing on page 729.

Interflow vector

In this example, assume that no agents are available and that split 11 (location 2) has the lowest adjusted EWT. The **queue-to best** command in the primary vector will interflow the call to the interflow vector at location 2. The following example shows what the interflow vector looks like.

BSR example of interflow vector on remote communication server

```
1.consider split2 pri m adjust-by 0
2.consider split 11pri m adjust-by 0
3.queue-to best
```

The interflow vector reconsiders the status of both splits to get the most current information and queues or delivers the call to the best split. Notice that the consider sequences in the interflow vector and the status poll vector are identical aside from their last step. When a call is interflowed, it is removed from any queues at the origin server and any audible feedback at the origin server is terminated.

CAUTION:
BSR will not operate correctly unless the consider series in the status poll vector and the interflow vector use the same splits/skills with the same queue priorities.

BSR call interflow with SIP

You can use SIP to interflow BSR calls, but BSR polling is not supported over SIP trunks. The polling must be done using either H.323 or ISDN trunks. Once the polling has determined the best site, the incoming call can be routed to another location by the **queue-to-best** step. The redirected call can be routed over a SIP trunk.

What happens to the call if the interflow attempt fails?

If the interflow attempt fails, for example, because there are no available trunks, the call is queued to the best local split. The call is not disconnected. The call is not dropped from vector processing on the origin server. For the call to be queued to a local split, however, that split must have been considered at some previous point in the consider series. In writing primary vectors, always consider local splits/skills before considering remote resources.

Can I adjust the AIT or AOC returned by an available resource?

To adjust the AIT or AOC returned for an available agent at a particular location, do the following tasks:

1. Go to the feature related system parameters screen.
2. Enable the **Available Agent Adjustments for BSR?** option field.

3. Go to the Vector screen and program a **consider split/skill** or **consider location** vector command specifying both the split/skill or location and the **adjust-by** parameter. The **adjust-by** parameter can be used to provide a percentage value during vector processing and can be:
   - A percentage (0 through 100)
   - A vector variable (A-Z, AA-ZZ)
   - A VDN variable (V1-V9)

Once the vector command is executed, the adjustment factor has the following result when the remote site has an available agent:

- For the MIA strategies, the adjustment reduces the agent idle time (AIT) received.
- For the LOA strategies, the adjustment increases the agent occupancy percentage (AOC) received.

Depending on the available agent strategy assigned to the VDN for the call, the adjusted AIT or the adjusted AOC is used for that local split/skill or remote location when choosing between available agents over multiple locations.

Example: You have an agent whose current AIT is 40%. You want to increase this agent's idle time to 60% to handicap sending the call to that remote location. If the strategy is ucd-loa, you can program the following vector command:

```
consider location 4 adjust-by 50
```

The occupancy used for location 4 is increased by 50% of the actual occupancy. The occupancy originally sent was 40%. A 50% adjust-by results in multiplying 40 by 50% resulting in 20. Therefore, 40 + 20 = 60%.

BSR available agent strategies

In multi-site BSR applications, the 1st-found available agent strategy results in fewer interflows and thus minimizes the load on trunking between communication servers. The communication server also has less processing to perform for each call in BSR vectors, since it may not need to compare as many resources to identify the best. If processing power and tie trunk capacity are issues in your multi-site applications, you may want to use the 1st-found strategy.

The other strategies typically result in a much greater percentage of calls being interflowed, thus optimizing load balancing across locations. For a strategy that greatly increases agent fairness across the network while limiting the number of trunks used, see Example of multi-site BSR with limited trunking on page 333.

More on status poll and interflow vectors

The following points are important to consider when you write status poll and interflow vectors.

- Since status poll vectors do not return answer supervision, call charges are not normally incurred for the status poll portion of the call flow.
- When a **consider location** step performs a status poll, it also checks for the availability of a B-channel. If no B-channel is available, the remote resource is never considered the best since the call cannot be redirected to it.
- If only one split/skill on a remote server can service the call type that is handled in a BSR application, you do not need to write a consider series in the interflow vector. You can just queue the call to the appropriate resource.
- If status poll and interflow vectors consider more than one split/skill, the VDNs for these vectors must be administered with the appropriate BSR available agent strategy.

User adjustments in multi-site BSR

User adjustments are especially important in multi-site applications, where unnecessary interflows may be costly and use trunk capacity inefficiently.

User adjustments in multi-site applications function in the same way they do in single-site BSR with one important difference: user adjustments may be applied at the remote servers in an application as well as at the origin server. Since a status poll vector uses **consider** steps to evaluate resources on the server where it resides, the **adjust-by** portion of each **consider** command allows the administrator at each server to set preferences for the splits/skills at that server. In BSR applications, any such adjustment for a split/skill is considered by the status poll vector in selecting the best resource on its server. The adjustment is then returned to the origin server along with the other data for that resource. When the server receives this adjustment from the remote server, it adds it to any adjustment that was assigned to that location in the **consider location** step. The following example assumes, of course, that no agents become available during the time these vectors are processing the call.

The following example shows a primary vector that considers one remote location, to which it assigns an adjustment of 30.

Vector with consider step for one location

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

The following example shows the status poll vector at location 2.

Status poll vector

```
1.consider split2 pri m adjust-by 0
2.consider split 11pri m adjust-by 20
3.reply-best
```

Consider split/skill commands in status poll vectors work just like they do in single-site BSR vectors. The user adjustments are applied to a single split/skill and not to the entire location. In this case, the two splits are assigned different adjustments. Say that split 11, despite having the larger adjustment, returns the lower adjusted EWT for a call. The **reply-best** command in step 3 returns the user adjustment of 20 to the primary vector on the origin server, along with the rest of the data for split 11.

In saving the data that is returned by location 2, the origin server adds the remote adjustment of 20 to the adjustment of 30 that is specified in step 3 of the primary vector. As a result, the call will not interflow to location 2 in this example unless the EWT for location 2 is more than 50 seconds better than the EWT in split 1 on the origin server.

Example of multi-site BSR with limited trunking

Multi-site BSR applications must balance improvements in wait times and agent utilization with the cost of interflows and the availability of inter-server trunking for status polls and interflows. The following example shows an application that is recommended for balancing agent workload across the network while still limiting tie trunk usage.

BSR example of Application Plan

```
BEST SERVICE ROUTING APPLICATION PLAN

 Number: 10   Name: International       Maximum Suppression Time: 60   Lock? y

NumLocation NameSwitch Node          Status Poll  VDN Interflow VDN  Net Redir?

1 Kansas City  1111919131234015    919131234115          n
2 New York   1112 912121234015    912121234115          n
3 Montreal   1113 915141234015    915141234115          n
3 London     1114 90114411234015  90114411234115        n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
___ _____________   ____________________    ____________          n
```

The following Vector Directory Number example shows the VDN screen for VDN 51110, the VDN that is used in this BSR Application Plan example. In the example, the entry in the **BSR Application** field links this VDN to BSR Application Plan 10. Also note the **EAD-MIA** entry in the **BSR Available Agent Strategy** field. If vector 100 uses BSR commands, calls are not automatically delivered to the first resource found with an available agent. In each consider sequence, when the **`queue-to best`** or **`check best`** step executes, one of the following things happens:

If …	**Then…**
No skill has an available agent	The call is queued to the skill with the lowest adjusted EWT.
Only one skill has an available agent	The call is delivered to that skill.
Two or more skills have available agents	The call is delivered to the skill with the most expert agent, which is the agent with the lowest skill level.
Two or more skills have available agents with the same skill level	The call is delivered to the skills that has the most idle agent.

Also note that **Allow VDN Override?** is set to **n**. If a second VDN and vector are used to process this call, the, the EAD-MIA strategy and the application plan that is specified for VDN 51110 is still used.

BSR example of primary VDN

```
change vdn xxxxx                                              page 1 of 3
              VECTOR DIRECTORY NUMBER

                             Extension: 51110
                                  Name: Multi-site BSR
                         Vector Number: 100
                     Attendant Vectoring? n
                      Meet-me Conference? n
                      Allow VDN Override? n
                                   COR: 59
                                    TN: 1
                              Measured: none
       Acceptable Service Level (sec): 20
              Service Objective (sec):
        VDN of Origin Annc. Extension: 1001
                             1st Skill:
                             2nd Skill:
                             3rd Skill:
```

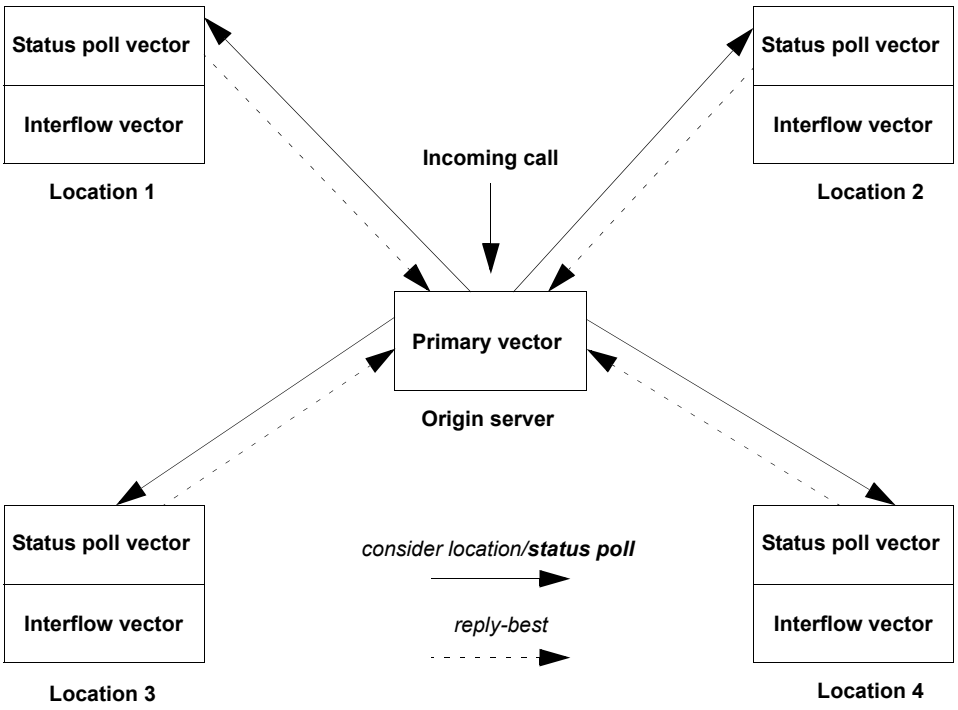
```
change vdn xxxxx                                              page 2 of 3
              VECTOR DIRECTORY NUMBER

                            Audix Name:
                  Messaging Server Name:
                     Return Destination:
                  VDN Timed ACW Interval:
                         BSR Application:15
          BSR Available Agent Strategy: UCD-MIA
              Observe on Agent Answer?:n
```

With four remote servers to be considered, the overall application is represented in the following figure. Application plan 10 on the origin server identifies the remote servers and provides the digit strings to dial into the VDNs for both the status poll vector and the interflow vector on each server.

Each **consider location** command in the primary vector places a status poll call to its specified location. The status poll vector at that location executes a series of **consider skill** commands and returns data on the best resource to the origin server through a **reply-best** command.

BSR example of multi-site application with four servers and limited tie trunk capacity



The following example shows the primary vector for this application. The first consider series in the primary vector tests two local skills. If either skill has an available agent, step 4 jumps to step 9 and the call is queued locally. No remote locations are polled. If no agents are available in either local skill, though, steps 5 to 8 test 4 remote locations. In general, you should not put other commands between **consider** steps. This use of the **goto** step is one of the few exceptions to that rule.

If the best remote location's adjusted EWT can reduce the call's current adjusted EWT, step 9 interflows the call to that location. In this vector, a local available agent is always favored over a remote available agent. Whichever location services a call, it will always be directed to the most idle, best skilled agent available.

Multi-site BSR example

```
1.wait time 0 secs hearing ringback
2.consider skill1 pri m adjust-by  0
3.consider skill 2 pri madjust-by 20
4.goto step 9 if expected-wait for best = 0
5.consider location 1adjust-by 30
6.consider location 2adjust-by 30
7.consider location 3adjust-by 50
8.consider location 4adjust-by 50
9.queue-to best
10.announcement 1001
11.wait time 60 secs hearing music
12.goto step 10 if unconditionally
```

In the primary vector, note that user adjustments are entered for local skill 2 as well as for all the remote locations. These indicate the administrator's preferences regarding both local and remote resources. For this example, let's say that neither local resource has an available agent and therefore an EWT greater than 0.

Status poll vector

Each receiving server in a multi-site application must have a status poll vector. To collect information from these locations, each **consider location** command in the primary vector places a status poll to the status poll vector for the appropriate server. The following example shows the status poll vector on the server at location 3.

BSR example of status poll vector at location 3

```
1.consider skill 2 pri m adjust-by 0
2.consider skill 11pri m adjust-by 20
3.consider skill21 pri m adjust-by 30
4.reply-best
```

This vector compares skills 2, 11, and 21, identifies the best one, and sends this information back to the origin server through the **reply-best** command. Notice that user adjustments are applied to skills 11 and 21 to adjust the skill's EWT. When EWT adjustments are applied at both the origin and remote servers, the two adjustments are added at the origin server. For more detail on user adjustments in multi-site applications, see User adjustments in multi-site BSR.

In this example, suppose that skill 11 has the best adjusted EWT at location 3. Its data, including a user adjustment of 20, is returned to the origin server by the **reply-best** command.

Finding the best resource

Once the remote servers have returned the best data for each location, the second consider series in the primary vector can be completed. In this example, let's suppose that no agents are available at any remote location.

The following table shows how user adjustments at the origin and remote servers yield the adjusted EWT for each location.

BSR best resource user adjustments

Location	Actual EWT of remote best (sec.)	User adjustment on origin server	User adjustment on remote server	Adjustment applied by origin server (sec.)	Adjusted EWT used in BSR calculations (sec.)
1	60	30	0	30	90
2	45	30	10	40	85
3	40	50	20	70	110
4	70	50	0	50	120

The second consider series identifies location 2 as the best remote location, with an adjusted EWT of 85, and the **`queue-to best`** step interflows this call to location 2.

Interflow vector

The interflow vector on a remote server in a multi-site application accepts the interflowed call from the origin server. It also executes the same consider series as the status poll vector to identify the current best resource, in case conditions have changed since the status poll.

The following example shows the interflow vector on a remote server.

BSR example of interflow vector at location 2

```
1.consider skill 2 pri m adjust-by 0
2.consider skill 11pri m adjust-by 20
3.consider skill 21pri m adjust-by 30
4.queue-to best
```

As happens today when a call is interflowed, it is removed from any queues at the origin server and any audible feedback at the origin server is terminated.

CAUTION:
BSR will not operate correctly unless the consider series in the status poll vector and the interflow vector use the same splits/skills with the same queue priorities.

Example of multi-site BSR with slow networks

Network response times are not an issue for most users. This example is intended for those users, if any, who experience such a problem. This example uses the same VDN, application plan, and four-server network that is described in the Example of multi-site BSR with limited trunking on page 333. The vector in that example minimized interflows by using a **goto** step that skips the remote consider series if a local resource has an available agent. This design is especially useful if network response times are slow. Calls are always queued once locally before remote locations are considered.

Furthermore, both status polls and interflows are conditional. The call can wait in the queue for a local resource while BSR looks for a better split/skill at remote locations.

This example also shows the function of the **check best** command and the **wait-improved** conditional.

The following example shows the primary vector for this application, vector 100. The first consider series in the primary vector tests two local splits and queues the call to the best one. If the EWT for the best split is 30 seconds or less, step 5 jumps to the loop in step 11 and the second consider series is not executed. If the EWT for the best split is over 30 seconds, though, steps 6 through 9 test 4 remote locations. If the best remote location can reduce the call's EWT by more than 30 seconds as compared to its EWT in the best local queue, step 10 interflows the call to that location.

CAUTION:
Be certain to queue calls at least once before using the wait-improved conditional in a vector step. If calls are not already queued when the step with the wait-improved conditional executes, The server reads the call's EWT as infinite. This could result in a vector that interflows all calls, even if that is not its intended function.

Multi-site BSR with EWT

```
1.wait time 0 secs hearing ringback
2.consider skill 1 pri m adjust-by  0
3.consider skill 2 pri m adjust-by 20
4.queue-to-best
5.goto step 11 if expected-wait for call <= 30
6.consider location 1adjust-by 30
7.consider location 2adjust-by 30
8.consider location 3adjust-by 50
9.consider location 4adjust-by 50
10.check best if wait-improved > 30
11.announcement 1001
12.wait time 60 secs hearing music
13.goto step 11 if unconditionally
```

A consider series can end with either a **queue-to best** or a **check best** step. The **check best** command lets you set conditions that must be met before a call is queued to the best resource. In this example, step 10 in the primary vector is **check best if wait-improved > 30**. In other words, step 10 interflows the call to the best location found by the consider series only if the EWT for that location is more than 30 seconds better than the call's EWT in the local queue.

You can use up to 3 consider series in one vector. It is possible to write more than 3 consider series in a vector, but there's no benefit in doing so. The server only allows you to queue a call simultaneously to 3 different local resources. Since each consider series ends by queuing a call (assuming no agent is available), using more than 3 series in a vector will not place the calls in additional local queues. If the call interflows to another communication server, it's removed from vector processing and any queues it was in on the origin server.

It is also possible to combine single-site and multi-site consider series, as this example shows. Note that user adjustments are entered for local skill 2 as well as for locations 3 and 4. These indicate the administrator's preferences regarding both local and remote resources. In this example, say that step 2 queues the call to skill 1, which has an EWT of 65 seconds, before the second consider series is executed.

Status poll vector

Each receiving server in a multi-site application must have a status poll vector. To collect information from these locations, each **consider location** command in the primary vector places a status poll to the status poll vector for the appropriate server. The following example shows the status poll vector on the server at location 3.

BSR example of status poll vector at location 3

```
1. consider skill 2 pri m adjust-by 0
2. consider skill 11pri m adjust-by 20
3. consider skill21 pri m adjust-by 30
4. reply-best
```

This vector compares skills 2, 11, and 21, identifies the best one, and sends this information back to the origin server through the **reply-best** command. Notice that user adjustments are applied to skills 11 and 21 to adjust the skill's EWT. When EWT adjustments are applied at both the origin and remote servers, the two adjustments are added at the origin server. For more details on user adjustments in multi-site applications, see User adjustments in multi-site BSR on page 332.

Suppose that skill 11 has the best adjusted EWT at location 3. Its data, including a user adjustment of 20, is returned to the origin server by the **reply-best** command.

Remember that the first consider series queued the call to local skill 1. Say that the second consider series identifies location 2 as the best remote resource. The **check** command in step 10 recalculates the call's current, unadjusted EWT in skill 1 and compares it to location 2's unadjusted EWT. If the call's actual (unadjusted) EWT can be improved by more than 30 seconds, the call is interflowed.

Note:
BSR uses adjusted EWT to determine which of the resources in a consider series is the best. Once the best resource is identified, subsequent **expected-wait** and **wait-improved** conditionals use the actual EWT values.

Interflow vector

When a call is interflowed to any of the remote locations, the interflow vector on that server accepts the interflowed call from the origin server. It also executes the same consider series as the status poll vector to identify the current best resource, in case conditions have changed since the status poll. The following example shows such an interflow vector.

BSR example of interflow vector at location 2

```
1. consider skill 2 pri m adjust-by 0
2. consider skill 11pri m adjust-by 20
3. consider skill21 pri m adjust-by 30
4. reply-best
```

CAUTION:
BSR will not operate correctly unless the consider series in the status poll vector and the interflow vector use the same splits/skills with the same queue priorities.

If the call is queued to a remote resource by step 10 in the primary vector, is the call removed from the local queue that it entered in step 4? When a call is interflowed, the call is removed from any queues at the origin server and any audible feedback at the origin server is terminated.

The second consider series can compare local and remote resources. If it does, and if step 10 queues the call to another local skill, will the call be removed from the local queue that it entered in step 4?

No. In general, the server can queue a call to as many as 3 local splits/skills simultaneously. BSR does not change this limit.

Example for handling excessive wait times

This short example shows a simple primary vector in a multi-site BSR application. If wait times are sometimes excessive because of high call volumes, step 4 of this vector directs calls to a **disconnect after announcement** step when wait time in the network exceeds 5 minutes. The following example shows a simple primary vector.

Multi-site BSR using disconnect for excessive wait times

```
1. wait 0
2. consider skill 1pri m adjust-by 0
3. consider location 2pri m adjust-by 30
4. goto step 6 if expected-wait for best ≤ 300
5. disconnect after announcement 3001
6. queue-to best
```

Announcement 3001 might say something like, *We're sorry. We are currently experiencing heavy call volume and cannot service your call at this time. Please try again later. We are normally least busy between 8 a.m. and 11 a.m. each morning.*

Planning and administering multi-site BSR

This section includes the following topics:

- About planning and administering multi-site BSR on page 342
- Selecting or administering application plans on page 343
- Administering the BSR Application Plan on page 343

About planning and administering multi-site BSR

This section presents information that is specific to BSR. Follow existing procedures to add or change other properties of VDNs and vectors not discussed in this section.

To create multi-site applications, follow the process below. List location numbers, Status Poll VDNs, and similar information so they will be available for planning and administration purposes. Define the purpose of the application

To define the purpose of the application:

1. Select the group of callers for which you want to create the application.
2. Define the goal of the application, for example, faster average speed of answer, better service by routing calls to the most qualified agents.

3. Decide which agent selection strategy (on VDNs) will best achieve your goal.
4. Decide whether you will implement BSR in a distributed system or a centralized system.
   - In a distributed system, all communication servers receive incoming calls and query other servers to interflow calls when appropriate.
   - In a centralized system, one server serves as a hub. All incoming calls arrive at this server and are routed from it to the other servers in the network.

   Since a distributed system is the more complicated of the two, the rest of this procedure is written in terms of implementing a distributed system. The same steps apply to implementing a centralized system, but only one server will have application plans and primary VDN/vector pairs.

Selecting or administering application plans

To select or administer a BSR application plan:

1. Select the VDNs on each server that serve the group of callers you have identified.

   On each server these are the Primary VDNs for your application. You may, of course, want or need to create new VDNs. In either case, record the extensions of each VDN that will point to a vector with a BSR application.
2. Select the locations that you want to include in each application plan. To uniquely identify each location, assign a number between 1 and 255 and a short name of 15 characters or less.
3. Record the node number of the server at each location.
4. Create Status Poll VDNs on each of the servers in the application plan.

   Record the full numbers you will need to route calls to these VDNs. These numbers will be entered on the Best Service Routing Application Plan screen when you create the plan.

   If you are creating new VDNs on the communication servers that will receive interflowed calls, record these numbers too. You will need them to complete the BSR Application Plan screen. Remember: you cannot use the same number for a Status Poll VDN and an Interflow VDN.

Administering the BSR Application Plan

This section includes the following topics:

- Defining the application plan on page 344
- Linking the application plan to a primary VDN and enter an agent selection strategy on page 345

Defining the application plan

To create an application plan on each communication server:

1. At the command line prompt, type **add best-service-routing *xxx*** and press Enter (where ***xxx*** is a number between 1 and 255 that you want to assign to this BSR application.)

   The system displays the Best Service Routing Application Plan screen. The number that you typed in the command appears in the **Application Number** field.

2. Assign a name to the plan.

   The best names are short and descriptive. This name cannot be longer than 15 characters.

3. Type in the information for the first remote location. Fill in the information for each field as shown below.

   Note:
   Each row on the screen contains all of the information the BSR application needs to identify and communicate with one of the resources in the plan.

Fields on application plan screen

Field	Type	Description
Num	Required	Type the number that you assigned to this location in 2.
Location Name	Optional	Type the name that you assigned to this location in 2.
Switch Node	Optional	This field is for user reference only. Leave it blank. If you are using the Universal Call ID feature, you may want to type each communication server node identity in this field. The server node identity is the number that is entered in the UCID Network Node ID field on page 4 of the Feature-Related System Parameters screen.
Status Poll VDN	Required	This is the complete digit string that your communication server will dial for the status poll call. The string can be up to 16 digits long.
Interflow VDN	Required	This is the complete digit string that your communication server dials to interflow a call to this location. The string can be up to 16 digits long.

4. Repeat 3 for each of the locations that you want to include in the application plan.
5. Press Enter to save your changes.

Note:
You must set up trunk groups to other sites. For information on setting up trunk groups, see Look-Ahead Interflow (LAI) on page 273 and Information Forwarding on page 189.

Linking the application plan to a primary VDN and enter an agent selection strategy

To link the application plan to a primary VDN and enter an agent selection strategy:

1. Go to the Vector Directory Number screen for the first VDN that you identified in 1.

   If this is a new application, create the VDN.

2. In the **Allow VDN Override?** field, type **y** or **n**. If the call is directed to another VDN during vector processing:
   - **y** allows the settings on the subsequent VDN, including its **BSR Available Agent Strategy**, to replace the settings on this VDN.
   - **n** allows the settings on this VDN, including its **BSR Available Agent Strategy**, to replace, or override, the settings on the subsequent VDN.
3. In the **BSR Application** field, type the application number you assigned to the plan.
4. In the **BSR Available Agent Strategy** field, type the identifier for the agent selection method you want this application to use:

If you enter…	The application will select the resource with…
1st-found	The lowest Expected Wait Time. If the application finds an available agent before it has compared all the locations in the plan, the application routes the call to that agent without contacting any other locations.
ucd-mia	The agent who has been idle the longest. The application compares all the locations in the plan.
ead-mia	The agent with the highest skill level, which is the lowest skill number, who has been idle the longest.
ucd-loa	The least-occupied agent.
ead-loa	The agent with the highest skill level, which is the lowest skill number, who is the least occupied.

5. Press **Enter** to save your changes.

Repeat steps 1 through 5 on each server that needs an application plan and a Primary VDN/ vector pair.

This process covers the administration that is needed for BSR vector commands to function. Now, of course, you need to write or modify the vectors that will control call processing.

Local treatment for remotely queued IP and ISDN calls

This section includes the following topics:

- About BSR local treatment for calls queued remotely on page 346
- Overview of local treatment operations on page 346
- Local treatment system requirements on page 348
- Local treatment administration on page 348
- Example vectors for the local treatment feature on page 349
- Special BSR local treatment considerations on page 353

About BSR local treatment for calls queued remotely

In a multi-site BSR configuration, a call that arrives at a local communication server can be rerouted to a remote server located in a different part of the world. To better meet the needs of such multi-site call centers, Avaya Communication Manager 2.0 or later includes a new **BSR Local Treatment for Calls Queued Remotely Over IP or ISDN Trunks** feature that allows you to provide local audio feedback for IP and ISDN calls while a call waits in queue on a remote server.

This feature provides the following potential benefits for call center operations:

- For multi-site BSR operations that include sites located in different countries, the new local treatment feature can result in significant bandwidth savings for IP calls.
- Audio quality concerns that occur when music is sent over wide area networks that use low bit-rate codecs are eliminated.
- Announcements and other treatments can be maintained and managed in a central location.

Overview of local treatment operations

This section describes local treatment feature operations that occur when BSR redirects IP or ISDN calls to a remote queue.

Important:
The local treatment operations described in this section assume that the required feature and vector administration steps are implemented on both the local *and* remote communication servers.

For information about feature administrations, see Local treatment administration on page 348.

For information about required vector design, see Example vectors for the local treatment feature on page 349.

The following steps describe the basic process for local treatment operations in a multi-site BSR environment:

1. A call arrives at the local communication server and is processed by a VDN that is enabled for BSR local treatment.
2. The local vector includes the **consider**, **queue-to best**, and **wait** hearing announcement steps that are required for BSR local treatment operations.
3. A skill on a remote server is identified as best location and the local server attempts an interflow to the remove server. Vector processing is temporarily suspended on the local server while the interflow attempt is in progress.
4. If the interflow attempt succeeds, the remote server returns an ISDN_PROGRESS message with progress indicator of in-band information (8) to indicate that the call is in queue and local treatment operations can proceed.

   The remote server must meet the following requirements for the appropriate ISDN_PROGRESS message to be sent back to the local server:

   - The remote server is administered for BSR local treatment.
   - The call is directed to a VDN that is also enabled for local treatment.
   - The vector associated with the VDN includes only those steps and commands that are required for successful local treatment operations.
5. The local server receives the ISDN_PROGRESS message with progress indicator of in-band information (8), vector processing resumes with an appropriate treatment step and the caller receives feedback provided by the local server while they wait in the remote queue.

Important:
To ensure that the local treatment feature operates as designed, use only the vector commands that are recommended for local treatment implementation. Although local treatment operations do not impose restrictions on the types of vector steps that are administered on the local server after call processing resumes, use of inappropriate vector steps can interfere with local treatment operations. For more information, see Example vectors for the local treatment feature on page 349.

6. When an ACD agent on the remote server accepts the call, an ISDN_ALERTING message is sent to the local server. Vector processing is discontinued on both the local and remote servers.

Local treatment system requirements

The BSR Local Treatment for Calls Queued Remotely Over IP or ISDN Trunks feature works on all platforms and operating systems that are supported by the Avaya communication server. You must meet the following licensing and system requirements to use the local treatment feature:

- The Avaya Communication Manager release must be 2.0 or later.
- The system license file must be configured to enable the following features:
  - Call Center Release 12.0 and later
  - LAI
  - BSR
  - BSR Local Treatment for IP and ISDN

Local treatment administration

The following tables show the administration screens used to administer the local treatment feature.

Important:
The **BSR Local Treatment?** field must be set to **y** on both the local and remote vdns. If the local vdn is set to **n** and the remote vdn is set to **y**, the remote communication server returns an ISDN_PROGRESS message with a progress indicator of in-band information. The local communication server considers this type of progress message to be invalid unless the local treatment flag is set and all interflow attempts result in dropped calls.

Local treatment administration - verify required features[1]

Administration command:	`display system-parameters customer options`	
Page name:	Call Center Optional Features	
Required field(s):	Call Center Release	**12**[2]
	Lookahead Interflow (LAI)?	y
	Vectoring (Best Service Routing)?	y
	BSR Local Treatment for IP & ISDN	y

1. Contact your Avaya account representative if this screen indicates that any of the required feature selections are not enabled.
2. Call center release 12 or later.

Local treatment administration - enable VDN

Administration command:	`change vdn xxx`	
Page name:	Vector Directory Number	
Required field(s):	BSR Local Treatment?	**y**[1]

1. The **BSR Local Treatment?** field must be set to **y** on both the local and remote vdns, or else call interflow attempts may result in dropped calls.

Example vectors for the local treatment feature

This section provides vector guidelines and examples that describe how to implement the local treatment feature. Vector administration typically requires polling vectors on both the local and remote communication server and an interflow vector on the remote server. The polling vector on the local server should also be administered to provide an appropriate local call treatment.

This section includes the following topics:

- Implementation guidelines for local treatment vectors on page 350
- Example vectors for the local communication server on page 350
- Example polling vector for the remote communication server on page 352
- Example interflow local treatment vector for the remote communication server on page 353

Implementation guidelines for local treatment vectors

This section describes the best practices for successful implementation of the local treatment feature.

Important:
Read these guidelines before you implement the local treatment feature.

Implementation of the local treatment feature requires use of specific vector steps to generate the correct ISDN messages between the local and remote communication servers. If the treatment, polling and interflow vectors that are administered to implement this feature include vector steps other than those recommended in this section, the feature may not work as intended and the associated bandwidth savings may not be realized.

For polling vectors: You must be careful to administer your local treatment polling vectors so that calls are not unintentionally dropped or phantom calls are generated. If the **queue-to best** step is followed by vector steps that include any commands other than **announcement**, **wait**, or **goto**, the trunk to the remote queue may be dropped. For example, the addition of **consider** steps after a **queue-to best** command can cause intermittent call behavior. The addition of a **queue-to** step after a **queue-to best** step may cause phantom calls to be queued to the remote server.

Tip:
You can also exploit this functionality to allow the local server to *take back* calls that remain in queue on a remote server after a specified time limit is exceeded. For more information, see Take back example on page 351.

Interflow local treatment vectors on the remote communication server: When the BSR Local Treatment feature is enabled, specific ISDN messages must be exchanged between the remote and local communication servers. If additional vector steps are included either before or after the **consider** steps (if used) and **queue-to best** in the interflow vector on the remote server, the following results occur:

- Either an ALERTING or PROGRESS message (with in-band information) is returned from the remote server to the local server.
- In response to the message, trunk bandwidth is immediately allocated and the call is removed from the local queue.
- Local treatment operations cease, trunk bearer resources are allocated for the call sooner than required and cost savings associated with the local treatment feature are not realized.

Example vectors for the local communication server

The following examples shows two different vector strategies that you can use to implement the local treatment feature on the local server. Vectors created for this purpose are the same as those used in all BSR polling operations, which include a **consider** series followed by a **queue-to best** step.

Important:
You must be careful to administer your local treatment polling vectors so that calls are not unintentionally dropped. For more information, see Implementation guidelines for local treatment vectors on page 350.

After the various skills and locations are polled and the call is placed in queue at the identified best location, the local server continues to maintain control of the call until it is answered by an agent. While the call is in queue, the local server continues to provide additional vector steps to implement the local call treatment.

At a minimum, the local treatment vector should include **announcement** and **wait-time** steps to provide appropriate feedback to the caller. However, the local treatment vector can be designed to use either a continuous loop or take back strategy. These alternate local call treatment strategies are described in the following sections.

Continuous loop example: the following example shows a vector that provides a sequence of call treatment steps on the local server that proceed in a continuous loop until an agent answers the call at the remote location.

In the following vector example, step 6 places the call in queue at the identified best location. Step 7 provides an appropriate announcement and step 8 provides 10 seconds of music. Step 9 uses an unconditional **goto** step to loop call processing back to step 6, where the treatment process continues.

```
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 7 if unconditionally
```

Take back example: The previous example set up the local treatment process as a continuous loop that repeats indefinitely while the call remains in queue at the identified best location. However, you can also design vectors that allow the local server to take back a call after it remains in queue for a specified amount of time.

In the following vector example, the **queue-to best** in step 6 is followed by a series of **announcement** and **wait-time** commands provided in steps 7 through 12. If the treatment steps complete and the call still remains in the remote queue, vector processing proceeds to step 13, which uses a **route-to** command that causes the call to the remote server to be dropped. The **route-to** step can be used to provide alternate services for the call.

Note:
When the call to the remote server is dropped, a type 305 vector event is logged.

```
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 announcement 3001
10 wait-time 10 secs hearing music
11 announcement 3001
12 wait-time 10 secs hearing music
13 route-to number 54010 if unconditionally
```

For another method to take back the call based on the amount of time the call has been in the system, see vdn type variable on page 127.

Example polling vector for the remote communication server

The following example shows a call vector that polls skills on the remote server. This vector does not differ from other typical BSR polling vectors.

```
change vector 31                                                    Page 1 of 3
                                    CALL VECTOR

Number: 31                    Name: Remote BSR poll 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 consider     skill    3   pri m adjust-by 0
02 consider     skill    4   pri m adjust-by 0
03 reply-best
```

Example interflow local treatment vector for the remote communication server

The following example shows a call vector that is used to interflow the call to the remote server while local treatment is provided for the call.

Important:
When the BSR Local Treatment feature is enabled, specific ISDN messages must be exchanged between the remote and local communication servers. If additional vector steps are included either before or after the **consider** steps (if used) and **queue-to best** in the interflow vector on the remote server, the following results occur:

- Either an ALERTING or PROGRESS message (with in-band information) is returned from the remote server to the local server.
- In response to the message, trunk bandwidth is immediately allocated and the call is removed from the local queue.
- Local treatment operations cease, trunk bearer resources are allocated for the call sooner than required and cost savings associated with the local treatment feature are not realized.

```
change vector 32                                                    Page 1 of 3
                                    CALL VECTOR

Number: 32                    Name: Remote BSR interflow 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 consider     skill    3   pri m adjust-by 0
02 consider     skill    4   pri m adjust-by 0
03 queue-to     best
```

Special BSR local treatment considerations

You should also understand the following items that pertain to the BSR local treatment feature:

Trunk group status: Calls that are queued remotely but are receiving local treatment are displayed as 'active' trunk members if the 'status trunk-group' command is performed on the interflowed trunk group. Even though the H.323 (IP) trunk member is 'active', no bandwidth is used because no voice packets are transmitted while local treatment is performed.

Path replacement : Path replacement is not supported for BSR Local Treatment calls. Both ends of the connection must be answered for path replacement to work. When BSR local treatment is enabled, the local VDN has answered, but the remote VDN where the call is queued has not answered. Therefore, path replacement can not occur when a call is queued remotely by local treatment VDNs. For more information about BSR path replacement, see BSR-initiated path replacement for calls in vector processing on page 355.

Troubleshooting for multi-site BSR

You should regularly execute a **`display events`** command for the appropriate vectors, especially if you have just implemented a new BSR application. Vector events will identify and indicate the source of common malfunctions and administration errors.

When tie-trunks or queue slots become exhausted, BSR cannot effectively balance calls across the network. If such problems are revealed frequently by vector events, review the design of the BSR application involved. If tie-trunks are frequently exhausted, the user adjustments on **`consider location`** steps may be set too low.

For a list of BSR vector events and definitions, see Tracking unexpected events on page 707.

Note:
Only the most recent events are displayed when a **`display events`** command is executed. For this reason, you should periodically display vector events to help quickly identify problems.

To verify that your BSR vectors are operating as intended, use a **`list trace vdn`** or **`list trace vec`** command to observe processing of an individual call. For more information, see Clearing events on page 726.

Note:
The **`list trace vdn`** and **`list trace vec`** commands are blocked if the Tenant Partitioning feature is enabled.

BSR status poll vectors must always end with a **`reply-best`** step. A **`busy`** or **`disconnect`** command should never be used.

Tips for writing BSR vectors

Before you write your first vector using BSR, you should study the sample vectors that are provided and familiarize yourself with the new commands and command elements. Sample vectors are provided in Single-site BSR on page 304 and Multi-site BSR on page 320. The new commands and command elements are explained in Call Vectoring commands on page 521.

As you write BSR vectors, it is strongly recommended that you follow the guidelines below.

- Arrange your **consider** steps in order of preference.

  The **consider** step that tests the main, or preferred, resource should be the first in the series. The second **consider** step should test the resource that is your second preference for handling the given call type, and so on. To avoid unnecessary interflows, put **consider** steps for local resources before steps that consider remote resources. This arrangement also provides a local best as a backup in case the interflow fails.

  Arranging consider steps in order of preference is recommended for all BSR vectors. It is especially important when the active VDN for the call is using the 1st-found agent strategy since the server delivers the call to the first available agent found, arranging **consider** steps in order of preference ensures that calls are delivered to the best of the available resources and that unnecessary interflows are avoided.

- Do not put any commands between the steps of a consider series that would cause a delay. **Goto** commands are OK.
- Do not put a consider series in vector loops.
- Confirm that calls queue successfully.

  This check is recommended for all vectors. Since EWT is infinite for a call that has not queued, a step that checks EWT after a queue attempt is a good confirmation method. After a **queue-to best** step, for example, a command such as **goto step X if expected-wait for call < 9999** should be included.

- Do not use the wait-improved conditional in a vector before you have queued the call at least once.

  The **wait-improved** conditional compares the call's EWT in its current queue to the best resource that is found by a consider series. If a call has not been queued and a vector step such as **check best if wait-improved > 30** is executed, the server interprets the call's current EWT as infinite and the **check best** step always routes the call to the best resource. In other words, in this situation the **check best** step functions like an unconditional **goto** or **route-to** command.

BSR-initiated path replacement for calls in vector processing

Path replacement for calls in queue and vector processing can be accomplished using QSIG or DCS with Reroute using ISDN SSE. For calls that are waiting in queue or in vector processing, even if the call is not connected to an answering user, path replacement can be attempted to find a more optimal path for this call. This results in more efficient use of the trunk facilities.

The **queue-to best** command is used in BSR to initiate a QSIG path replacement for a call. The following scenario can take place:

At the terminating communication server, if a Path Replacement Propose operation is received for a call that is in queue or vector processing, the server can immediately initiate path replacement using the Path Replacement Extension if the **Path Replace While in Queue/Vectoring** field is set to **y** and the **Path Replacement Extension** field has a valid entry. These fields are located on the ISDN parameters page of the Feature-Related System Parameters screen.

The ability to track a measured ACD call after a path replacement has taken place is available for CMS versions r3v9ai.o or later. Starting with the r3v12ba.x release, CMS reports a path replacement as a *rename* operation rather than a path replacement. The *rename* operation properly reports scenarios where a path replacement takes place from a measured to an unmeasured trunk facility. Avaya recommends that you upgrade CMS to r3v12a.x or later and administer all trunks associated with path replacement as *measured* by CMS to ensure better CMS tracking of path-replaced calls.

Example vector

The following example shows how a BSR vector can be written to trigger path-replacement at the terminating communication server.

Note:
In order for a path-replacement to be attempted, the incoming and outgoing trunks that are used for the call must be administered with the **Supplementary Service Protocol** field set to **b**.

BSR-initiated path-replacement vector

```
1.wait 0
2.consider skill 1
3.consider skill 5
4.consider location 10 adjust-by 10
5.consider location 24 adjust-by 20
6.queue-to best
```

At the terminating (receiving) server, the vector that is executed by the incoming call must be programmed with an **announcement**, or **wait hearing music** vector command. The use of one of these commands is what makes it possible for path-replacement to take place while the call is in vector processing.

Holiday Vectoring

Holiday Vectoring enables a set of commands that can be used to write call vectors for calls to be routed on holidays or any days when special processing is required.

This section gives you the information you need to use this vectoring option and includes the following major topics:

- Command set on page 357
- Holiday Vectoring overview on page 358
- Administering Holiday Vectoring on page 359
- Holiday Vectoring considerations on page 365

Command set

The following table shows the commands that are available for use in Holiday Vectoring.

Holiday Vectoring command set

Command category	Action taken	Command
Branching/programming		
	Go to a vector step Go to a vector	`goto step` `goto vector`

Branching/programming commands

Holiday Vectoring allows use of two branching/programming commands, including:

- goto step command on page 358
- goto vector command on page 358

The following sections detail the syntax that can be used for these commands and any information that is specific to their use in Holiday Vectoring.

goto step command

Syntax 1

```
goto step <step #> if holiday in table <table #>
```

This command directs the call to a specific vector step if the conditions of the call match a holiday that is in the specified Holiday Table.

Syntax 2

```
goto step <step #> if holiday not-in table <table #>
```

This command directs the call to a specific vector step if the conditions of the call do not match any of the holidays that are in the specified Holiday Table.

goto vector command

Syntax 1

```
goto vector <vector #> if holiday in table <table #>
```

This command directs the call to a specific vector if the conditions of the call match a holiday that is in the specified Holiday Table.

Syntax 2

```
goto vector <vector #> if holiday not-in table <table #>
```

This command directs the call to a specific vector if the conditions of the call do not match any of the holidays that are in the specified Holiday Table.

Holiday Vectoring overview

Holiday Vectoring is an enhancement that simplifies vector writing for holidays. It is designed for customers who need to reroute or provide special handling for date-related calls on a regular basis.

This feature provides the user with the capability to administer as many as 99 different Holiday Tables, then use those tables to make vectoring decisions. Each table can contain up to 15 dates or date ranges. All of this can be done in advance to ensure seamless call routing over holidays when staffing is reduced or call centers are closed.

When vector processing encounters a **goto xxx if holiday in table #** step, it determines if the current date and time qualifies as a holiday according to the given table. That information is then used to decide whether the goto condition is true or false, and therefore, whether to **goto** the given step or vector or not. The date and time match is done at the time that the call is in vector processing. It is done just like time-of-day routing. This means that it is checking the system date and time on the Processor Port Network **(**PPN), rather than the local port network time on the Expansion Port Network (EPN).

The Holiday Vectoring feature is not limited to holiday use, but can also be applied to any date-related special processing. For example, vectors can be modified or created to perform special processing during a two-week television promotion or a semiannual sale.

This feature was developed in response to customer needs, especially for some customers who may have as many as 30 bank holidays to administer throughout the year. Holiday Vectoring streamlines vectoring tasks and ensures seamless operation over holiday (or special-event) periods.

Without this feature, call center administrators had to write special vectors for each holiday or other special date-related circumstances, and make sure that these vectors were administered at the appropriate times. In some cases, administrators were required to go to work on holidays just to administer vectors. This feature was developed in response to customer needs, especially for some customers who may have as many as 30 bank holidays to administer throughout the year.

Administering Holiday Vectoring

This section gives you step-by-step instructions on setting up Holiday Tables and writing vectors to include Holiday Vectoring. This section includes the following topics:

- Enabling Holiday Vectoring on page 359
- Setting up a Holiday Table on page 360
- Changing vector processing for holidays on page 362

Enabling Holiday Vectoring

The Holiday Vectoring customer option can be enabled if either Vectoring (Basic) or Attendant Vectoring is enabled.

You can have up to 99 different Holiday Tables if you have the Communication Manager 3.0 Enhanced Vectoring option enabled. Otherwise, you can have up to 10 Holiday Tables.

On the Customer Options Screen, the Vectoring (Holidays) field should be set to **y**. If the feature is not enabled, contact your Avaya customer support or authorized representative to have the feature enabled.

Setting up a Holiday Table

This section describes how to set up a Holiday Table before adding to a vector and includes the following topics:

- Holiday Table command syntax on page 360
- Using the Holiday Table commands on page 360

Holiday Table command syntax

This section describes the syntax of each Holiday Vectoring command.

Syntax 1

```
change holiday-table x
```

This command allows you to change the entries in a Holiday Table.

To create a new Holiday Table, you must use the change command and give the number of a blank table. For example, change holiday-table 9, where table 9 has not been used to define holidays.

Syntax 2

```
display holiday-table x
```

This command allows you to display the entries in a Holiday Table.

Syntax 3

```
list holiday-table
```

This command lists all of the Holiday Tables.

Syntax 4

```
list usage holiday-table x
```

This command lists all vector steps that refer to the selected Holiday Table.

Using the Holiday Table commands

After ensuring that Holiday Vectoring is enabled on the Customer Options screen, enter:

```
change holiday-table 1
```

On the Holiday Table screen, which is shown in the following example, enter the holiday information.

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              12    31               Christmas
    01    01  00   00     01    01  10   00      New Year's Day
```

Note:

When using a range of dates, the end date must be greater than the start date. Ranges must be within one calendar year. In the example above, two entries were made, one for each calendar year.

The Holiday Table screen can be used for entering individual holidays or holiday ranges. The following rules apply to entering dates on this screen:

- If a day is entered, the corresponding month must be entered.
- If a month is entered, the corresponding day must be entered.
- If an hour is entered, the corresponding minute must be entered.
- If a minute is entered, the corresponding hour must be entered.
- If an hour and minute is entered, the corresponding month and day must be entered.
- If a month and day is entered, the corresponding hour and minute is not required.
- If an end month and day is entered, the corresponding start month and day must be entered.
- If a start month and day is entered, the corresponding end month and day is not required.
- To enter an individual holiday, enter a start month and day, but do not enter an end month and day.
- To enter a holiday range, enter both a start month and day and an end month and day.
- The start month, day, hour, and minute must be less than or equal to the end month, day, hour, minute.
- The description field is an alpha-numeric field that is used for identification.

After creating a holiday table, use the **display holiday-table** command to view the entries. To list all of the holiday tables, use the **list holiday-table** command, as shown in the following example.

Listing the Holiday Tables

```
list holiday-table
                                HOLIDAY TABLES

                            Table Number      Name
                                 01           Business Holidays
                                 02           Annual Promotion Dates
                                 03           Summer Special
                                 04
                                 05
                                 06
                                 07
                                 08
                                 09
                                 10
```

Changing vector processing for holidays

After administering the holiday tables, add or change vector processing for those holidays.

On the command line, enter **change vector *x*** (where ***x*** is the vector number). The Call Vector screen contains a display-only field that indicates that Holiday Vectoring is enabled. On the Call Vector screen, customers can enter a new goto conditional for the holidays.

When Holiday Vectoring is optioned, a field on the Vector screen identifies if the vector on which you are currently working is a Holiday Vectoring vector, as shown in the following example.

Call Vector screen

```
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? n   ANI/II-Digits? n   ASAI Routing? n
 Prompting? y   LAI? n  G3V4 Adv Route? n   CINFO? n   BSR? n   Holidays? y

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

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

The following examples use **goto** commands to route calls for holidays.

Holiday Vectoring example 1

```
 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
```

Holiday Vectoring example 2

```
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
```

After you have assigned Holiday Tables to several vectors, you can use the **list usage holiday-table** command, as shown in the following example, to display which vectors and vector steps are using the selected Holiday Table.

List of Holiday Table use in vectors

```
list usage holiday-table
                              LIST USAGE REPORT

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

Holiday Vectoring considerations

Consider the following when administering Holiday Vectoring:

- Administration of Holiday Tables is supported only on the communication server and cannot be changed using adjunct vectoring tools.
- Holiday Vectoring is only available when Vectoring (Basic) or Attendant Vectoring is enabled.
- There is no validation that verifies the consistency among the 15 holidays in any table. If the same holiday is entered twice, the system stops checking with the first entry that is found.
- With holidays that are ranges of dates, the ranges could overlap. When a call is in vector processing, the holidays are checked from top to bottom on the table and the check stops if a match is found. Even though there might be multiple entries that would match, the check stops at the first match.
- There is a validation that the day of the month that is entered is valid with the given month. Specifically, if the month is April, June, September, or November, then the date must be a number between 1 and 30. If the month is January, March, May, July, August, October, or December, then the date can be a number between 1 and 31. If the month is February, then a the date can be a number between 1 and 29.

  Note:
  The year is not checked in holiday vector processing. This allows the same holidays to be used year-to-year when the holiday is on a fixed date. For holidays where the date changes from year-to-year, the holiday tables must be readministered.

- When disabling the Holiday Vectoring feature (changing the value of the **Vectoring (Holidays)** field from **y** to **n** on the Customer Options screen), the vectors are checked for any **`goto...if holiday`** steps. If any of these steps are found, an error message is displayed, and the change is not allowed. The customer must remove those vector steps first before the feature can be disabled.

Service Hours Table Routing

Service Hours Table Routing specifies office working hours using up to 99 different tables.

This section includes the following topics:

- Service Hours Table Routing overview on page 367
- Administering Service Hours Table Routing on page 368
- Service Hours Table Routing considerations on page 369
- Service Hours Table Routing scenario on page 370

Service Hours Table Routing overview

Use Service Hours Table Routing to simplify the vectors you use for handling calls based on office hours. Vectors use the Service Hours Routing tables to determine how to handle calls that are received during working hours versus calls that are received out of hours. Customers can use this feature as an alternative to tod (time of day) routing and can specify working hours on a daily or hourly basis. This feature allows you to administer as many as 99 different tables, then use those tables to make vectoring decisions.

Reason to use: Before this feature, customers added multiple time-of-day steps to their vectors in order to define the hours of operation for a specific business application (VDN or vector). This feature allows customers to define service hours clearly, simply, and in one place. One simple vector command can check to see if the call meets the administered service hours.

goto processing

When vector processing encounters a **goto...if service-hours** step, it determines if the current day of week and time is within the service hours listed in the corresponding table. This information is used to decide if the **goto** condition is true or false, and therefore, whether or not to "go to" the given step or vector. The day of week and time match is based on the system time on the communication server that receives the call. The time used in the calculations is the time the call reaches the **goto** step.

Time adjustments on the Service Hours Table screen

The time used in the calculations can be adjusted using the **Use time adjustments from location** field on the Service Hours Table screen. This field indicates the location number on the Locations screen that specifies how the adjustments are performed.

You can make the following time adjustments using the **Use time adjustments from location** field:

- Adjust the daylight savings time from the system time
- Apply the time zone for a specific location
- Apply the daylight savings time for a specific location

If this field is blank, no adjustments are made.

Administering Service Hours Table Routing

This section gives you step-by-step instructions on setting up Service Hours tables and using vectors to control the flow of the call based on the data specified by the corresponding table.This section includes the following topics:

- Administering the Service Hours Table screen on page 368
- Administering the goto conditional on page 369

Administering the Service Hours Table screen

To administer a Service Hours Table:

1. Enter **`add service-hours-table x`**

   x = 1-99

2. Enter values in the following fields:
   - **Description**
   - **Use time adjustments from location**
   - **Start**
   - **End**

   For a description of these fields, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Administering the goto conditional

Service Hours Table Routing allows the use of two branching and programming commands,:

- goto step command on page 369
- goto vector command on page 369

goto step command

Syntax 1

goto step *x* if service-hours in table *y*

This command directs the call to a specific vector step if the conditions of the call match the service hours specified in the Service Hours Table.

Syntax 2

goto step *x* if service-hours not-in table *y*

This command directs the call to a specific vector step if the conditions of the call do not match any of the service hours that are in the specified Service Hours Table.

goto vector command

Syntax 1

goto vector *x* @step *z* if service-hours in table y

This command directs the call to a specific vector if the conditions of the call match service hours that are in the specified Service Hours Table.

Syntax 2

goto vector *x* @step *z* if service-hours not-in table y

This command directs the call to a specific vector if the conditions of the call do not match any of the service hours that are in the specified Service Hours Table.

Service Hours Table Routing considerations

Consider the following when administering Service Hours Table Routing:

- Service Hours Table Routing is not available when upgrading from a previous release.

- Vectoring (Basic) must be enabled.
- The Call Center Release field must be set to 4.0 or later.

Service Hours Table Routing scenario

The following is a very basic scenario (not considering time adjustments):

Basic Service Hours Table scenario

```
                              SERVICE HOURS TABLE

                                  Number: 99
                          Description: Call-ahead Reservations_____
                 Use time adjustments from location: 2__

     MON           TUE           WED           THU           FRI
 Start  End    Start  End    Start  End    Start  End    Start  End
 08:00 12:30   10:30 02:00   __:__ __:__   __:__ __:__   __:__ __:__
 13:00 16:30   15:00 20:30   __:__ __:__   __:__ __:__   __:__ __:__
 __:__ __:__   __:__ __:__   __:__ __:__   __:__ __:__   __:__ __:__
 __:__ __:__   __:__ __:__   __:__ __:__   __:__ __:__   __:__ __:__
 __:__ __:__   __:__ __:__   __:__ __:__   __:__ __:__   __:__ __:__

                          SAT           SUN
                      Start  End    Start  End
                      __:__ __:__   __:__ __:__
                      __:__ __:__   __:__ __:__
                      __:__ __:__   __:__ __:__
                      __:__ __:__   __:__ __:__
                      __:__ __:__   __:__ __:__
```

VECTOR 1:

goto vector 2 @step 1 if service-hours not-in table 99

<Service hours – Call-ahead Reservation processing>

VECTOR 2:

<After hours processing>

The following table shows how calls at different times will be processed:

Day of week / Time of call	Which processing?
Sunday / any time	After hours
Tuesday / 14:59	After hours

Day of week / Time of call	Which processing?
Tuesday / 15:00	Service-hours
Monday / 12:30	Service-hours
Monday / 12:31	After hours

The same scenario (considering time adjustments):

```
                                DAYLIGHT SAVINGS RULES
Rule          Change Day                  Month   Date    Time    Increment
 0: No Daylight Savings
 1: Start: first Sunday       on or after April     1    at 02:00   01:00
    Stop: first Sunday        on or after October   25   at 02:00
 2: Start: first Sunday       on or after April     1    at 02:00   02:00
    Stop: first Sunday        on or after October   25   at 02:00
```

```
LOCATIONS
ARS Prefix 1 Required For 10-Digit NANP Calls? y
 Loc.    Name          Timezone  Rule    NPA                              Proxy Sel.
 No.                   Offset                                             Rte. Pat.
 1:     Main          + 00:00     1
 2:     Branch        + 02:00     2
 3: :
 4: :
 5: :
 6: :
 7: :
 8: :
 9: :
10: :
11: :
12: :
13: :
14: :
```

The following table shows how calls at different times will be processed:

System time	Daylight Savings?	Adjustments	Adjusted time	Which processing?
Tuesday / 07:30	yes	- subtract 1 hour (put system into standard time) - add 2 hours (location time zone) - add 2 hours (location DST)	Tuesday / 10:30	Service-hours
Tuesday / 11:01	yes	- subtract 1 hour (put system into standard time) - add 2 hours (location time zone) - add 2 hours (location DST)	Tuesday / 2:01	After hours
Monday / 06:00	no	- add 2 hours (location time zone)	Monday / 08:00	Service-hours
Monday / 10:31	no	- add 2 hours (location	Monday / 12:31	After hours

Percentage allocation routing

This section includes the following topics:

- About percentage allocation routing on page 373
- Administering percentage allocation routing on page 374
- Percentage allocation routing example on page 374
- Considerations for implementing percentage allocation routing on page 375
- Rules for percentage allocation routing on page 377

About percentage allocation routing

This feature allows you to distribute calls among a set of call centers or VDNs based on specified percent allocation. Various types of incoming calls that arrive at a particular VDN can be be directed to a Policy Routing Table (PRT) instead of to a vector. The PRT then distributes the calls to the administered Route-to VDNs based on the specified percent allocation targets.

This feature is useful for segmented call-handling, outsourcing, and optimizing call handling in a multiple-location enterprise. Using percentage allocation routing, you can allocate target percentages and, for example, do the following:

- Allocate certain call types among multiple answering groups with similar skills
- Allocate maximum calls to a more economical calling group
- Ensure that the organization meets the terms of a service level agreement

Administering percentage allocation routing

The following screens and fields are required to administer the percentage allocation routing options:

Screen	Field
Vector Directory Number (VDN)	Destination Number (select Policy Routing Table and enter the PRT number)
Policy Routing Table (PRT)	Number Name Type Period Route-to VDN Target %

For a detailed description of the VDN and PRT screens, see the ACD Call Center Screens section in the *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Percentage allocation routing example

The Destination and Number fields in the Vector Directory Number screen allow you to specify the routing destination as either a vector or a PRT table. To implement percentage allocation routing, you need to specify routing destination as a PRT table (Destination: Policy Routing Table and Number: PRT table number).

Using the Policy Routing Table screen, you can specify the various routing destinations and the percentages allocated for each of the destinations. The following example Policy Routing Table screen includes the Target and Actual traffic routed to each of the VDNs.

```
display policy-routing-table 1957
                              POLICY ROUTING TABLE
Number: 1957  Name: % distribution      Type: percentage   Period: max count

                                                    Target Actual Call
Index Route-to VDN  VDN NAME                         %      %      Counts
  1   2220071       Gizmo support                    25     27.2      3
  2   2221501       Ultra support                     5      9.0      1
  3   2220601       Customer Service South           35     27.2*     3
  4   2220511       Outsourcer Charlie               10      9.0      1
  5   2220501       Survey after service             10      9.0      1
  6   2220072       Outsourcer International         15     18.1      2
  7
  8
  9
 10
 11
 12
                                          Totals 100              11

Command:
```

Considerations for implementing percentage allocation routing

Consider the following when implementing percentage allocation routing:

- Modification of the table while calls are being routed to the VDN(s) the PRT is assigned to can result in calls being miss-routed, miss-appropriated or other indeterminate actions. To avoid this, first create a new PRT table with the required changes and then replace the old version.
- The following are the various valid entries in the period field:
  - 100_count (default): Call counts (and displayed %) are reset after total calls for the PRT reach 100 which is when the total calls match the target routing pattern percentages. This ensures that the routing points have equal distribution of calls all the time.
  - max_count: Call counts are maintained until calls delivered to at least one of the VDNs exceed 65,400. At that point calls are continued to be distributed over the VDNs but the call counts are reset when the actual percentages equal the targets for all of the VDNs at the same time.
  - Half-hour: Resets the call counts at the top of the hour and at the 30 minute point.

- hour: Resets the call counts at the top of the hour.
- daily: Resets the call counts at midnight, every night.
- weekly: Resets the call counts at midnight on Saturday.

Note:
For a detailed description of Policy Routing Table screen and its fields, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

- Actual % and Call Counts are reset whenever the PRT form is changed.
- Calls are routed to the VDN destination that is farthest from meeting its target allocation.
- Calls are routed based on actual call counts, not actual %. When there are no actual call counts (at startup or after reset), the route point with the highest target is selected.
- For VDN domain monitoring with CTI/ASAI, a call to a VDN that is routed via a PRT appears as if the call was routed to the destination VDN by a route-to number command in a vector assigned to the original VDN
- Calls routed through a PRT are reported to BCMS, CMS and IQ as though they were routed to a VDN assigned to Vector 0 (vector number set to 0). VDN reports can be created using existing CMS custom reporting capabilities to show the percentages and the number of calls that have distributed among the destination VDNs.
- A Policy Routing Table can be assigned to multiple VDNs. There are no restrictions as to the VDNs that can be entered as route points.
- There are limits for the number of PRT tables that can defined and there is a system routing point (PRTs x VDN destinations) limit. Both of these limits can be found in the Communication Manager System Capacity tables and displayed on the display capacity form.
- In addition to add, change, display, list, and remove (explained in the *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference*), following also are commands that support PRT:
  - list vdn: Displays destination type V(ector) or P(RT)
  - list usage policy-routing-table: Lists the VDNs that use the specified PRT
  - list history: displays history of add and change policy-routing-table commands
  - list trace vdn: displays calls to a PRT
- When Destination is a PRT, the Attendant Vectoring and Meet-Me Conferencing fields do not appear.
- For VDN domain monitoring with CTI/ASAI, a call to a VDN that is routed via a PRT appears as if the call was routed to the destination VDN by a route-to number command in a vector assigned to the original VDN
- To evenly distribute calls across 3 routing points, administer the 3 routing points with a 33% target, then add a 4th routing point that routes back to the PRT with a target of 1%. Set the Period to "100_count" and calls will evenly distribute.

Rules for percentage allocation routing

Consider the following rules when implementing percentage allocation routing:

- The target percentages must be whole numbers (integers). The form will not let you use decimals or fractions.
- The target percentage of all the VDNs must add up to 100% before form submittal.

Network Call Redirection

This section includes the following topics:

- About NCR on page 379
- NCR options supported by PSTNs on page 380
- NCR considerations on page 386
- NCR and Information Forwarding on page 387
- NCR feature interactions on page 389
- NCR implementation methods on page 390
- NCR administration on page 397
- NCR troubleshooting on page 408

About NCR

Network Call Redirection (NCR) provides an Avaya communication server call routing method between sites on a public network or a Virtual Private Network (VPN) that can reduce trunking costs. These cost reductions are particularly valuable in enterprises or multi-site call center environments where trunk costs are high.

When an incoming call arrives at an Avaya communication server that has the NCR feature enabled, call redirection is managed by the Public Switched Telephone Network (PSTN) or VPN switch instead of the local Avaya server. As a result, trunks that the server would otherwise retain to accomplish a trunk-to-trunk transfer are released after the call redirection takes place.

The cost reductions associated with reduced trunk use can be significant particularly when Avaya virtual routing features, such as Best Service Routing (BSR) with Expected Wait Time (EWT), are implemented. The cost-savings are achieved by the Avaya customer requiring fewer trunks to handle the same number of incoming/outgoing calls after the NCR feature is implemented within the local communication server.

NCR options supported by PSTNs

This section describes the various NCR redirection options that are supported by Public Switched Telephone Networks (PSTNs). All of the NCR protocols described in this section support Information Forwarding using UUI transport to the redirected-to location over the PSTN or VPN network.

This section includes the following topics:

- Protocols not supported by NCR on page 380
- Network Call Transfer-type options on page 380
- Network Call Deflection (NCD) on page 382
- AT&T In-Band Transfer and Connect on page 383

Protocols not supported by NCR

The PSTN call-redirection protocols that are currently available but are *not* supported by the NCR feature include the following:

- AT&T 4ESS Out-of-Band Transfer and Connect
- Nortel RLT

Network Call Transfer-type options

This section describes the features and operations that are common to various Network Call Transfer-type (NCT-type) redirection protocols over ISDN trunks.

This section includes the following topics:

- About NCT-type feature operations on page 381
- Specific NCT-type protocols on page 381
- Selection of outbound call leg for NCT-type NCR protocols on page 382

About NCT-type feature operations

A key advantage of NCT-type protocols invoked by call vectoring or by manual station call-transfer or call-conference/release operations is that the redirecting server retains control over the call and can continue to use a trunk-to-trunk connection if the PSTN switch does not accept the request to merge B-channels for both legs of the call. If the PSTN switch returns a *failure* message to the PSTN, the originating server maintains a trunk-to-trunk connection for the call. For vector call processing, NCR call processing still considers the NCR attempt to be successful, but the following outcomes occur:

- A vector event is logged to indicate that the NCT operation as attempted with the PSTN failed.
- If Avaya CMS is used to track incoming calls to an externally measured VDN, the call is not counted as deflected.

For NCR invocation by call vectoring, the local Avaya communication server sets up the second leg of the call, waits for the second site to be connected, and then requests the PSTN or VPN switch to merge the first leg of the call with the second leg. If this request is accepted, the PSTN or VPN switch joins the original caller to the redirected-to endpoint, sends a *success* message back to the redirecting server and then drops both legs of the call at the redirecting server.

For NCR NCT or TBCT invocation by a station, ACD agent, VRU, or CTI-controlled call doing a call-transfer or call-conference/release operation, if the second leg of the call is set up over an outgoing trunk in the same signaling group as the incoming call, then call-redirection takes place when the call-transfer or call-conference/release occurs. For the NCR ISDN ETSI ECT protocol, the call redirection will take place when the outgoing trunk B-channel either has the same or a different D-channel than the incoming call.

Specific NCT-type protocols

Specific NCT-type protocols include the following protocols:

MCI Network Call Transfer over ISDN PRI: Network Call Redirection and PSTN switch operations associated with the MCI NCT protocol are consistent with those described in About NCT-type feature operations on page 381.

MCI Network Call Redirection/Network Call Transfer is compliant with ANSI Explicit Network Call Transfer (ENCT) T1.643 (1995), the MCI Nortel variant of ANSI ECT (1995).

Note:
MCI NCT is offered in the United States by MCI for their Nortel DMS-250 and Alcatel DEX-600 PSTN switches.

Two B-Channel Transfer (TBCT) over ISDN PRI: Network Call Redirection and PSTN switch operations associated with the TBCT protocol are consistent with those described in About NCT-type feature operations on page 381.

The Network Call Redirection/Telcordia Two B-Channel Transfer (TBCT) protocol is compliant with the Telcordia Two B-Channel Transfer and ANSI Explicit Call Transfer (1998) standards. For more information, see any of the following:

- Telcordia GR-2865-CORE
- ANSI T1.643 (1998)
- Lucent 99-5E-7268

Note:
TBCT is offered in the U.S. by AT&T for their DMS-100 PSTN switches configured with the NI2 network protocol. TBCT is offered in Canada by Bell/ Canada for their DMS-100 PSTN switches; and by AT&T/Canada, for their DMS-500 PSTN switches.

ETSI Explicit Call Transfer: Network Call Redirection and PSTN switch operations associated with the European Telecommunications Standard Institute (ETSI) Explicit Call Transfer (ECT) protocol are consistent with those described in About NCT-type feature operations on page 381.

The Network Call Redirection/ETSI Explicit Call Transfer protocol is compliant with ETSI standard EN 300 369-1.

Note:
ETSI ECT is offered in Europe by France Telecom and other in-country PSTN service providers for their Ericsson AXE-10 PSTN switches. ETSI ECT is offered in the United Kingdom by MCI for their DMS-100 PSTN switches.

Selection of outbound call leg for NCT-type NCR protocols

For the MCI NCT and Telcordia TBCT NCR protocols, the PSTN switch requires that the outbound call leg of a redirected PRI call is in the same trunk group and has the same Direct Access Line (DAL) D-channel as the inbound call. For vector-initiated invocation of the NCR feature by either a BSR **`queue-to best`** or **`route-to number`** vector step, the Avaya communication server enforces this requirement by automatically selecting an outbound B-channel that has the same signaling group as the incoming call's D-channel. This results in sending the NCR invocation request on the same D-channel used for the first call leg's associated signaling or for the same associated D-channel when the Non-Facility Associated Signaling (NFAS) D-channel backup configuration is used.

For the ETSI ECT NCR protocol, there is no restriction that the outbound PRI call leg must have the same Direct Access Line (DAL) D-channel used for the first call leg's associated signaling. If the PRI trunk group has more than one associated D-channel, NCR processing sets up the second call leg for call redirection using any B-channel in the trunk group independent of its associated D-channel.

Network Call Deflection (NCD)

The Network Call Deflection (NCD) operation by a PSTN switch can occur only if the incoming call to the Avaya communication server is not answered (that is, an ISDN CONNECT message is not sent to the PSTN switch from the incoming server).

The NCR NCD feature is compliant with ETSI Supplementary Services Network Call Deflection ETS 300 207-1 (partial call rerouting in the public network).

Important:
Some call vectoring commands cause CONNECT messages to be sent to the PSTN switch. If call vectoring methods are used to implement NCR and the PSTN switch supports the NCD protocol, call vectors used to invoke NCR must not include any of the following vector commands:

```
announcement

collect x digits

converse-on split/skill

wait hearing music

wait hearing (announcement extension) then (continue, music,
  ringback or silence)
```

When the Avaya server invokes the NCD feature, the PSTN switch sets up the second leg of the call instead of the redirecting Avaya communication server. There are two PSTN options for NCD specified by the ETSI standards: *retain call until alerting/connect* and *clear call upon invocation*. The *clear call upon invocation* option is commonly referred to as a *partial call reroute*.

When the *clear call on invocation* option is used, a successful NCR/NCD attempt is indicated when the PSTN or VPN switch has validated the NCR request and sends a *call reroute return* DISCONNECT message to the originating server. In this case, the server loses control of the call after it is transferred to the PSTN or VPN redirection endpoint, and no alternate transfer method is possible if the PSTN or VPN switch fails to transfer the call to the second location.

The *retain call until alerting/connect* option is not available because there are presently no known PTSN or VPN offers with this protocol. With this option, the PSTN or VPN switch sets up the second leg of the call, waits until an ALERTING message is received, and then sends a *call reroute return* FACILITY message followed by a DISCONNECT message to the originating server. In this case, if the second leg of the call fails, the server can redirect the call with a trunk-to-trunk connection so that the call is not lost.

NCD is offered in Europe by British Telecom for their Marconi/Plessey System X and Ericsson AXE10 PSTN switches; and by Deutsche Telecom for their Siemens EWSD and Alcatel S12 PSTN switches. NCD is offered in Australia by Telstra for their Alcatel S12 PSTN switches.

AT&T In-Band Transfer and Connect

This section describes PSTN redirection operations associated with the AT&T In-Band Transfer and Connect service. Details of the service are described in AT&T Technical Reference 50075.

NCR provides Information Forwarding support for the AT&T In-Band Transfer and Connect network service ISDN D-channel data-forwarding capability. The Information Forwarding feature forwards the UUI that is associated with the call to the redirected-to location. When call vectoring and AT&T In-Band Transfer and Connect are used to transfer a call, and NCR is enabled for the system, the **disconnect** vector step causes UUI IE information to be inserted into the ISDN DISCONNECT message generated by a successful AT&T In-Band Transfer and Connect operation.

Note:
For information about NCR administration and other administration measures that are required when the AT&T In-Band Transfer and Connect service is used, see Administering NCR with AT&T In-Band Transfer and Connect on page 404.

AT&T In-Band Transfer and Connect operations can be initiated by call vectoring after first doing the following switch administration:

1. Administering a **route-to number** vector step with an announcement extension, where the associated announcement is recorded with Dial Tone Multi-Frequency (DTMF) tones that include a ***T** followed by a PSTN endpoint number.
2. Administering a BSR location **VDN Interflow** field on the Best Service Routing Application Plan screen as an announcement extension, where the associated announcement is recorded with DTMF tones that include a ***T** followed by a PSTN endpoint number.
3. Administering a BSR location **VDN Interflow** field on the Best Service Routing Application Plan screen as a local switch VDN number associated with a vector that contains an **announcement** step, where the associated announcement is recorded with DTMF tones that include a ***T** followed by a PSTN endpoint number.

When the **route-to number** vector in action 1 is executed, or when a **queue-to best** vector step is executed and the BSR location described in action 2 or action 3 above is selected as the BSR *best* location for call interflow, the AT&T In-Band Transfer and Connect operation succeeds, but no UUI IE information is sent to the redirected-to PSTN endpoint by the Avaya communication server. However, for Step 3 above, NCR can be administered for use with the AT&T In-Band Transfer and Connect feature and a **disconnect hearing announcement none** vector added after the announcement step such that UUI information associated with the call is passed to the routed-to endpoint when the call redirection is completed. This UUI information can be used to do agent screen pop-ups at the redirected-to PSTN endpoint where the call is interflowed.

BSR call-flow resulting in AT&T In-Band Transfer and Connect UUI IE

A typical BSR call-flow that results in UUI IE information being inserted in the ISDN DISCONNECT message during a successful AT&T In-Band Transfer and Connect operation is as follows:

1. A PRI call from the PSTN switch arrives at the local Avaya communication server and is routed to a VDN that uses a vector to do subsequent BSR vector processing.

2. The BSR polling vector steps on the local server receive status information from various local skills and remote BSR locations, and identifies a remote call center site as the BSR *best* location.
3. Call control passes to the interflow VDN selected as the BSR *best* location specified on the Best Service Routing Application Plan screen.

   For information specific administering a BSR application plan, see Call vectoring methods used with AT&T In-Band Transfer and Connect service on page 407, or for general information about BSR application plans, see Selecting or administering application plans on page 343.

   Important:
   The **Net Redir?** field in the BSR application plan for the remote location must be set to **n**.
4. The vector associated with the interflow VDN for the BSR *best* location includes the following:
   - An **announcement** vector step that specifies an extension for which a special sequence of DTMF digits has been recorded. The recorded DTMF digits return in-band information about the redirected-to endpoint back to the PSTN. The DTMF digits provided in the announcement are entered from a Touch-tone keypad, and use the format:

     `*T + PSTN number`

     T corresponds to the number 8 button on a DTMF keypad, and *PSTN number* represents the PSTN endpoint number where the call is redirected.

   Note:
   The phone equipment required to create the announcement is described in Setting up DTMF announcements for AT&T In-Band Transfer and Connect on page 406.

   - A **`wait-hearing silence`** step provides a brief interval to allow sufficient time for the PSTN switch to process the DTMF digits.
   - A **`disconnect after announcement none`** vector step. This vector step sends an ISDN DISCONNECT message that includes a UUI Information Element. The UUI IE contains Avaya Information Forwarding for the call that is sent to the PSTN switch.
5. The PSTN switch makes the connection to the specified redirected-to endpoint and releases the B-channel connection to the Avaya communication server.

NCR considerations

The following sections describes things that you should understand when you implement NCR:

- Limitations on call redirection on page 386
- NCR operational considerations on page 387

Limitations on call redirection

You should understand the following items that pertain to limitations on the NCR feature:

NCR feature support: PSTN support for NCR varies with geographical location and may be limited or absent in some areas. Consult your Avaya account team to determine PSTN Service Provider availability of one of the NCR protocols in your area.

NCD redirection protocol support: At this time, no PSTNs offer the Network Call Deflection *retain call until alerting/connect* operation. Therefore, only the Network Call Deflection *clear call upon invocation* offer is available from PSTNs. Both methods are described in this document. It is advised that you negotiate with your PSTN as the NCR feature will work on either platform. NCR is limited by which PSTN platform is available to you.

Allowable number of redirection per call: There may be limits placed on the number of times a call may be redirected over the public network. These limits are imposed by the public network service provider. For example, in the United States, MCI currently allows only one redirection per call. In the United Kingdom, there is a limit of 20 call deflections per call. In addition, there may be additional charges associated with redirected calls.

User-to-User information forwarding support: Some public network service providers do not support forwarding of User-to-User Information (UUI), including Adjunct Switch Application Interface (ASAI) user data, collected digits, VDN name, the VDN in-time (as reflected by the NETINTIME database items), and the UCID. In such situations, Information Forwarding will be lost and the second leg of the redirected call will look like an entirely new call to the redirected-to server at the second location.

One of the data items lost is the VDN name, which is rerouted to the originally called service (DNIS) information. The indication that the call has been forwarded can be achieved by using dedicated VDNs for call forwarding, but this strategy loses the benefits of Information Forwarding inherent with NCR and limits use of CTI applications.

PSTN service providers typically charge by call or by a monthly rate for the redirect and UUI transport services. For more information about such charges, contact your Avaya account team.

NCR operational considerations

Reserving outbound trunk B-channels to ensure NCR operations succeed: When the trunk group service type is set to call-by-call, the trunk group Usage Allocation capability can be used to reserve a minimum number of trunk channels for outgoing PRI B-channel calls within the same trunk group and same D-channel.

For more information, see Reserving trunk group B-channels for NCT-type redirection operations on page 401.

Call vectoring configuration required for successful MCI NCT operations: When NCR is used with the MCI NCT protocol, the VDN call vector the call is redirected to by a successful MCI NCT operation must immediately return an ISDN CONNECT message to the PSTN switch. To meet this requirement, either a **`wait 0 secs hearing music`** or an **`announcement`** vector step should be the first step executed in the redirected-to call vector.

Ericsson AXE-10 configuration required for successful ETSI ECT operations: Following is Ericsson AXE-10 release and configuration information required for successful NCR ETSI ECT operations:

- Verify that AXE-10 has *VN7 Translocal 4.2* or later software. This is also called *GOAS 2.1* by Ericsson.
- Configure the AXE-10 for the *pure ETSI* level.
- Configure all PRI trunks used with the Avaya Communication Manager 2.0 NCR/ETSI ECT feature to subscribe to the AXE-10 *ETSI ECT* mode. On the AXE-10 trunk configuration screen, configure the ECT category to *ON*.
- Do *not* configure the AXE-10 PRI trunk to expect a *HOLD* ISDN message to be sent by the NCR ETSI ECT feature as part of the ETSI ECT invocation sequence.

NCR and Information Forwarding

The Avaya Information Forwarding feature is supported with NCR when the PSTN supports ISDN UUI IE transport in conjunction with the specific network redirection protocol used by the switch.

This section includes the following topics:

- UUI data included in Information Forwarding on page 388
- UUI data forwarding on page 388
- PSTN terms used for UUI transport service on page 388

UUI data included in Information Forwarding

Information Forwarding forwards the following call center-related data (as User-to-User Information) with an ISDN call:

- Adjunct Switch Application Interface (ASAI) user data
- Universal Call ID (UCID)
- Collected digits
- In-VDN time
- VDN name.

UUI data forwarding

When an NCT-type option is used for NCR, the UUI is forwarded by the Avaya communication server in the ISDN SETUP message sent with the call to the second site.

When the NCD option is used for NCR, the UUI is included in the ISDN FACILITY invoke message sent from the Avaya communication server to the PSTN. The PSTN then forwards the UUI to the second site.

When the AT&T In-Band Transfer and Connect service is used for NCR, the UUI is returned in an ISDN DISCONNECT message that includes the data in a codeset 0 or 7 UUI IE element.

PSTN terms used for UUI transport service

For NCT-type options and the NCD option, the PSTN service provider must configure the PRI trunks used with the Avaya NCR feature to transport the UUI data associated with the Avaya Information Forwarding feature. The various PSTN terms used in different countries for UUI IE transport are listed in the following table.

Country	UUI Transport Term	Providers
Australia	UUS Service 1	Telstra
Canada	UUS Service 1	AT&T/Canada Bell Canada
France	Mini-Message	France Telecom
Germany	(included in basic ISDN package)	Deutsche Telecom
Singapore	Not supported	

Country	UUI Transport Term	Providers
UK	Not supported	
USA	N-Quest Type 1 Service MA UUI Type 1 Service	MCI AT&T

NCR feature interactions

NCR interacts with the following call center features:

- Attendant Vectoring — Attendant Vectoring can use the **`route-to number`** vector step to route calls to attendants located at another communication server node. The operation of the NCR feature using the NCT-type or NCD networks features to accomplish the call redirection is exactly the same as for redirecting ACD calls.

  For more information, see Using the route-to command for NCR on page 623.

- Advice of Charge — No new capabilities are added for the NCR feature for the Advice of Charge PSTN feature. The Advice of Charge feature should be used with the same trunk facilities used for the NCR feature.

- BCMS — No change is made to BCMS for support of NCR. Redirected calls are tracked as completed calls since the PSTN disconnects the incoming facility of the original call when the call is redirected to another site.

- Enhanced Information Forwarding — For the NCR feature, Enhanced Information Forwarding transports User-to-User information (UUI) for the incoming ISDN call to the PSTN endpoint that receives the redirected call. The use of the Enhanced Information Forwarding capability with NCR (the recommended configuration) requires that the incoming call trunk group be assigned as shared (i.e., the **UUI IE treatment** field is set to shared). However, if the trunk group is set up as service provider, only the ASAI user information (or user information provided by the incoming ISDN call) will be included in the UUI IE sent on a non-shared basis to the redirected-to PSTN endpoint. NCR supports Information Forwarding for AT&T In-Band Transfer and Connect service.

- Look-Ahead Interflow — NCR activation using the **`route-to number`** vector step does not require Look-Ahead Interflow to be active to provide multi-site capabilities, which are required for considering remote locations and access to the BSR Application Plan screen.

- Service Observing by VDN — If the Service Observing by VDN feature is used to service observe a VDN, where the NCR feature is used to redirect incoming ISDN calls, the service-observer will hear the same tones, music, and/or announcements heard by the incoming caller before the NCR feature reroutes the call to another PSTN endpoint. When the NCR operation is completed, the service-observer will be dropped as an observer of the incoming call and placed in the service-observing queue associated with the VDN.

- Trunk-to-Trunk Transfer — If the NCR feature is optioned and the ASAI Third-Party make Call/transfer operation is used to redirect an incoming ISDN to a PSTN endpoint, the **Trunk-to-Trunk Transfer** field on the System-Related Customer Options for must be set to y for the call redirection to succeed. If the **route-to number** or BSR **queue-to best** vector step uses the NCR feature to redirect an incoming ISDN call to a PSTN endpoint, the Trunk-to-Trunk Transfer customer option does not need to be set to y.

  For more information, see Using the route-to command for NCR on page 623.
- VDN Return Destination — If the VDN Return Destination feature is administered for the VDN that is associated with a vector that causes the NCR feature to be invoked, the VDN Return Destination feature will be canceled when the call is redirected by NCR.
- CMS database items — The following Avaya CMS database items are affected by NCR:
  - DEFLECTCALLS: In the VDN CMS database tables, the DEFLECTCALLS item includes the number of calls that are redirected using NCR through the BSR feature by using the **route-to number** or **queue-to best** commands. Successful NCR attempts are pegged as DEFLECTCALLS.
  - INTERFLOWCALLS: In the VDN CMS database tables, the INTERFLOWCALLS item includes successful BSR interflows using NCR redirections.
  - LOOKATTEMPTS: In the VDN CMS database tables, the LOOKATTEMPTS item includes the number of times the Look-Ahead Interflow or BSR interflow was attempted for calls in the vector. Successful Look-Ahead Interflow or BSR attempts ar also counted. NCR invoke attempts (NCD or NCT) are also reflected in LOOKFLOWCALLS.
  - LOOKFLOWCALLS: In the VDN CMS database tables, the LOOKFLOWCALLS item includes the number of INTERFLOWCALLS that were redirected by the Look-Ahead Interflow or BSR features. LOOKFLOWCALLS is a subset of INTERFLOWCALLS and includes LOOKATTEMPTS for the Look-Ahead Interflow or BSR interflows. With BSR interflow using trunk-to-trunk transfer or NCR, every LOOKATTEMPT will also be counted as a LOOKFLOWCALLS unless a failure occurs.

NCR implementation methods

This section describes the different methods that you can use to activate the NCR feature, which include the following:

- NCR activation using call vectoring methods on page 391
- NCR activation using ASAI Call Transfer and third-party Merge/Release operations on page 395
- NCR activation using station call transfer or call conference/release operations on page 396
- NCR activation using ASAI adjunct route operations on page 397

NCR activation using call vectoring methods

This section describes the call vectoring methods that can be used to implement NCR and provides some basic example vectors. This section includes the following topics:

- Summary of call vectoring-activated NCR operations on page 391
- Using BSR queue-to best vector step to activate NCR on page 392
- Using route-to number ~r vector step to activate NCR on page 393
- Other things to know about using NCR with ASAI on page 395

Summary of call vectoring-activated NCR operations

The processes by which NCR is implemented by a call vectoring method is summarized in the following steps:

Note:
The following description does not apply when the AT&T In-Band Transfer and Connect service is used with NCR. For a description of NCR operations associated with that service, see AT&T In-Band Transfer and Connect on page 383.

1. The PSTN switch sends an incoming ISDN call to the Avaya communication server, where the call enters vector processing.
2. One of the following occurs:
   - If the Avaya communication server trunk group and PSTN or VPN switch are configured to use an NCT-type redirection protocol, the redirecting communication server must return an ISDN CONNECT message to the PSTN switch. Any of the following vector commands can be used to return the message:
     - `announcement`
     - `collect x digits`
     - `converse-on split`
     - `wait hearing music`
     - `wait hearing (announcement extension) then ("continue", "music", "ringback" or "silence")`

   Note:
   If the redirecting communication server does not execute one of the vector steps listed above, a CONNECT message is automatically returned to the PSTN switch.

   - If the server trunk group and PSTN or VPN switch are configured to use the NCD redirection protocol, a CONNECT message must *not* be sent to the PSTN switch. Therefore, when the NCD protocol is applied, none of the vector commands listed above should be included in call vectors that implement NCR.

3. Call processing proceeds to either a **route-to number ~r** or BSR **queue-to best** vector step. Depending on which type of redirection is administered for the incoming trunk group, either NCT-type or NCD processes are initiated. In either case, a FACILITY message is sent to the public network over the D-channel associated with the incoming trunk to invoke redirection of the call.

   Note:
   You should understand the following items that pertain to the PSTN or VPN endpoint number and receiving vector for the interflow location.

4. The PSTN or VPN switch indicates redirection success or failure consistent with the protocol-specific operations described in NCR options supported by PSTNs on page 380. An unsuccessful NCR attempt results in one of the following outcomes:

   - If an NCT-type protocol is used, the redirecting communication server establishes a trunk-to-trunk connection.
   - If the NCD protocol is used and the Avaya DEFINITY version is earlier than load 37 of Release 10, vector processing continues to the next vector step that follows the **queue-to best** vector step without any best local BSR call treatment.
   - If the NCD protocol is used, the call may be redirected to the best location by means of a trunk-to-trunk connection. However, the ability of the originating server to establish such a trunk-to-trunk connection depends on the specific features of the NCD protocol in use. For more information, see Network Call Deflection (NCD) on page 382.

Using BSR queue-to best vector step to activate NCR

NCR is especially useful for multi-site call center operations in which the Best Service Routing feature is enabled, since the number of PRI B-channels needed for call interflows is reduced. The **queue-to best** vector step can be used to interflow ISDN calls between communication servers over the PSTN. This method provides the best approach for balancing loads across a multi-site environment and is more cost effective and accurate than pre-delivery routers. For more information about BSR, see Best Service Routing (BSR) on page 297.

NCR is activated by the **queue-to best** vector step when the BSR feature determines a BSR location is the best BSR location and that location is administered with the **Net Redir?** option set to **y** on the BSR Application Table screen. Note that the administered **Interflow VDN** field on the Best Service Routing Application screen must be a PSTN or VPN endpoint number without a trunk/ARS/AAR access codes included. For some PSTN switch dialing plans, the long-distance access code (for example, a "1" in the United States) must be prefixed to the PSTN number for the call to be succesfully routed by he PSTN switch.

As shown in the following example, the Best Service Routing Application Plan screen must include locations that have the **Net Redir?** field set to **y**.

```
                  BEST SERVICE ROUTING APPLICATION

Number: 1    Name:        Maximum Suppression Time: 60      Lock? y

Num  Location Name   Switch Node    Status Poll VDN  Interflow VDN    Net Redir?
1    Omaha                          95552011       3035551211           y
2    Paris                          95552022       18005551234          y
3    Sydney                         95552033       18665553456          y
```

An appropriate vector is then used to identify a BSR best location and NCR is activated by the **queue-to-best** vector step.

```
wait 2 seconds hearing ringback
consider skill 1 pri l adjust-by 0
consider location 1 adjust-by 20
consider location 2 adjust-by 40
consider location 3 adjust-by 20
queue-to best
```

Using route-to number ~r vector step to activate NCR

This method can be used to invoke NCR when a **route-to number** vector step that specifies a number that begins with the ~r character. This method can be used to invoke NCR with or without the **LAI** option set to **y** or with Attendant Call Vectoring active.

Note that the administered **route-to number** vector step **number** field must be a PSTN or VPN endpoint number without a trunk/ARS/AAR access codes included. For some PSTN or VPN switch dialing plans, the long-distance access code (for example, a "1" in the United States) must be prefixed to the number for the call to be succesfully routed by the PSTN or VPN switch.

Example route-to number ~r vectors: The following examples show vectors that include **route-to number** commands to activate NCR, either with or without use of the Attendant vectoring feature.

```
wait 0 seconds hearing ringback
goto step 4 if skill oldest-call < 30 secs
route-to number ~r13035403001
queue-to skill 35 priority m
...
```

```
goto step 6 if time-of-day is all 17:00 to 09:00
wait 0 seconds hearing ringback
queue-to attd-group
wait 999 secs hearing music
stop
route-to number ~r13035551002
```

Using vector/VDN variables with route-to number ~r to activate NCR

The **number** field of the **route-to number ~r** vector command can be administered with a global vector variable A-Z or AA-ZZ instead of a PSTN endpoint number after the leading *~r* characters. The **number** field of the **route-to number ~r** vector command can also be administered with a VDN variable V1-V9 instead of a PSTN endpoint number after the leading *~r* characters.

An example of using the **route-to number ~r** vector command with a vector variable in the **number** field is shown in the following example. For this example, it is required in the Variables for Vectors screen that the following administration is done:

- Vector variable A is defined as of type collect for digit-buffer and L for local
- Vector variable T as of type tod to contain the current system clock time-of-day value

```
1.  goto step 5 if T < 0700      [if time-of-day is less than 7:00 a.m., set up NCR
                                  call-redirection to out of hours PSTN endpoint]
2.  goto step 5 if T > 1800      [if time-of-day is after 6:00 p.m., set up NCR
                                 call-redirection to out of hours PSTN endpoint]
3. set A = none CATR 18005555555   [set digit-buffer to in-office
                                          hours PSTN endpoint number]
4. goto step 6 if unconditionally [jump to step 6 to do NCR call-redirection ]
5. set A = none CATR 18661111111 [set digit buffer to out-of-office hours
                                                PSTN endpoint number]
6. route-to number ~rA [initiate NCR call-redirection operation]
```

For information about using variables with the ~r vector step, see route-to command with vector variables on page 117 or route-to command on page 619.

NCR activation using ASAI Call Transfer and third-party Merge/ Release operations

NCR NCT-type operations are activated by ASAI call processing when the Call Transfer or Third-Party Merge/Release operation is performed by a CTI application. This occurs in the following manner:

1. This is typically initiated by the CTI application user selecting an icon, menu item, or button to transfer an answered incoming ISDN call to another party over the PSTN.

   Since the incoming ISDN call must be connected to a station user before the Call Transfer or Third-Party Merge/Release operation is requested, NCR can only initiate the call redirection if an NCT-type protocol is optioned on the trunk.

2. If a call arrives at an ASAI-monitored VDN, ASAI will send appropriate information in the ASAI disconnect event to notify the CTI application that the call has been redirected by NCR.

For the ASAI operations listed above to succeed, the following conditions must be in effect:

- The **Network Call Redirection** field is set on the System Parameters Customer Options screen.
- An NCR NCT-type protocol is administered for both the incoming and outgoing call ISDN trunk group.
- The PSTN number that the CTI application uses to redirect an incoming ISDN call to another PSTN endpoint must be added to the ARS digit analysis screen in such a way that for the NCR MCI NCT and TBCT protocols, the second leg of the call transfer uses the same trunk group with a trunk that has the same D-channel as the incoming call. For the NCR ETSI ECT protocol, the CTI-initiated second call leg can be over a different trunk group with a different signaling group than the incoming call leg.

Note:
NCR-related AAR/ARS routing table administration is required for station transfer or conferencing with MCI trunks. For more information, see Station or ASAI transfer or conference/release administration on page 400.

Other things to know about using NCR with ASAI

Using ASAI data for call tracking : ASAI event reporting allows tracking of ISDN ACD calls that were redirected by NCR in a multi-server call center environment. These calls can be tracked by the UCID assigned to each call, or by the UUI information inserted by the application through either the Third Party Make Call or Adjunct Routing features.

ASAI drop event: Successful NCR call redirection causes an ASAI drop event to be sent to the CTI application with a CV_REDIR cause value of decimal (30) after the redirection is completed. Only one NCR drop event is received for a successful NCR operation when the NCT PSTN feature is used, even though two trunks are dropped by the PSTN.

ASAI third-party merge/call transfer: The CTI application requests a third-party merge/call transfer ASAI operation to transfer the call to the second communication server. This is only used if Network Call Transfer is not available. Once the two calls merge, then ASAI sends a third-party acknowledgement, and when the call is completed, ASAI sends a drop event report, and the third-party call ends.

NCR activation using station call transfer or call conference/ release operations

When an incoming ISDN call over a trunk with NCT-type PSTN service is answered at a station or voice response unit (VRU), the station user or VRU places the call on hold, and dials the number for a PSTN of VPN endpoint where the outgoing trunk B-channel is determined by AAR or AAS routing. The station user initiates a call transfer using the Transfer feature button or a switch hook flash, or the VRU initiates a call transfer by using an analog or line-side E1/T1 switch-hook flash.

The switch automatically sends an *invoke NCT* FACILITY message when the transfer is completed if the following conditions are met:

- The **Network Call Redirection** field is set to **y** on page 3 of the System Parameters Customer Options screen.
- An NCT-type protocol is administered for both the incoming and outgoing call ISDN trunk group.
- The second leg call is eligible for redirection by means of an NCT-type protocol, which requires for the MCI NCT and TBCT protocols the second leg of the call is in the same trunk group and has the same signaling group as the incoming call. For the NCR ETSI ECT protocol, the second leg of the call can be over a different trunk group with a different signalling group than the incoming call leg.

If the station user or IVR initiates and completes a three-way conference instead of doing a call transfer operation as above, and releases or hangs up from the conference with the following condition also being met, the switch automatically sends an *invoke NCT* ISDN message to the PSTN or VPN switch if also the following condition is met:

- The number of parties in the conference including the conference originator must be no greater than three parties.

Note:

NCR-related AAR/ARS digit-analysis and routing table administration is required for correctly setting up the second call leg over NCT-type trunks associated with the station or IVR call transfer and call conference/release operations. For more information, see Station or ASAI transfer or conference/release administration on page 400.

NCR activation using ASAI adjunct route operations

NCR can be invoked by specifying the activate NCR option for the ASAI route message in a **route select** ASAI message sent by a CTI application to the Avaya communication server after an **adjunct routing** vector step is executed during call vector processing. This Communication Manager 2.0 capability provides greater flexibility for CTI applications to directly route calls to PSTN or VPN endpoints without the need to specify a VDN extension in the **route select** ASAI message to route the call instead to a VDN and vector step that activates NCR via a **route-to number ~r** or **queue-to-best** vector step. The invocation of NCR by the **adjunct routing** vector step **route select** ASAI message for various NCT-type protocols follows the same rules as used for the **route-to number ~r** or **queue-to-best** vector step operations.

For more information, see the following ASAI documents:

- For information about the Call Options codepoint for NCR Routing or the ASAI Call Route Selection message, see *ASAI Protocol Reference*.
- For information about possible feature interactions, see *ASAI Technical Reference*.

NCR administration

The following sections list NCR administration requirements. Some of the administration requirements will vary according to the specific method used to implement NCR.

This section describes the following NCR administration requirements:

- Basic administration on page 398
- Station or ASAI transfer or conference/release administration on page 400
- Reserving trunk group B-channels for NCT-type redirection operations on page 401
- Administering NCR with AT&T In-Band Transfer and Connect on page 404

Basic administration

Important:
The basic administration requirements described in this section do *not* apply if NCR is being used with the AT&T In-Band Transfer and Connect service to invoke NCR. To see administration requirements specific to the AT&T service, see Administering NCR with AT&T In-Band Transfer and Connect on page 404.

NCR administration - verify customer options for NCR

Administration command:	`display system-parameters customer options`	
Page name:	Optional Features	
Required field(s):	G3 Version	V8 or later[1]
	ISDN/SIP Network Call Redirection	y

1. NCR NCT and NCD protocols requires V8 or later, NCR TBCT protocol requires V11 or later, and NCR ETSI ECT protocol requires V12 or later.

NCR administration - BSR screen[1]

Administration command:	`change best-service-routing x`	
Page name:	Best Service Routing Application	
Required field(s):	Net Redir?	y

1. Required only if the BSR feature is enabled.

NCR administration - trunk group screen

Administration command:	**`change trunk-group x`**				
Page name:	Trunk Group				
Required field(s):	Direction:	two-way			
	Service Type:	cbc			
	Usage Alloc:	y			
	Disconnect Supervision In?	y			
	Disconnect Supervision Out?	y			
		(Settings specific to PSTN redirection options)			
		MCI NCT	TBCT	ETSI ECT	NCD
Required field(s):	Group Type:	ISDN	ISDN	ISDN	ISDN
	Supplementary Services Protocol:	g	a	c	c
Page name:	Trunk Features				
	Network Call Redirection:	y			

NCR administration - signaling screen

Administration command:	**`change signaling group x`**				
Page name:	Signaling Group				
		Supported PSTN Redirection option			
		MCI NCT	TBCT	ETSI ECT	NCD
Required field(s):	Group Type:	ISDN			
	Network Call Transfer:	y	y	y	n

NCR administration - DS1 screen

Administration command:	`change ds1 [board location]`[1]				
Page name:	DS1 Circuit Pack				
		Supported PSTN Redirection option			
		MCI NCT[2]	TBCT[3]	ETSI ECT[4]	NCD[5]
Required field(s):	Country Protocol:	Any, but typically 1a	1b or 1d	etsi	etsi

1. Board location parameter values are: [cabinet(1-1)];carrier(A-E);slot(0-20) OR [gateway(1-10)];module(V1-V9).
2. MCI NCT: Verizon® Network Call Transfer
3. TBCT: Telecordia-Two B-Channel Transfer
4. ETSI ECT: ETSI Explicit Call Transfer
5. NCD: Network Call Transfer

Station or ASAI transfer or conference/release administration

Use this section when using station or ASAI call transfer/conference with NCT-type NCR trunks.

The NCR feature is automatically activated for a station or ASAI call transfer/conference when:

- The **Network Call Redirection** field is set to **y** on page 3 of the System Parameters Customer Options screen.
- For the MCI NCT or TBCT NCR protocols, the second call leg of the call transfer for an incoming ISDN call is made using the same trunk group with a B-channel that has the same D-channel as the incoming call.
- For the NCR ETSI ECT protocol, the second leg of the call can be over a different trunk group with a different signaling group than the incoming call leg.

In addition to this, you must do the following tasks to administer station or ASAI transfer:

1. In order for a NCT NCR-type operation to successfully completed, add the PSTN number that a station or ASAI user dials to transfer an incoming call to another PSTN endpoint to the ARS digit analysis screen. This entry must then be administered with an ARS routing pattern that routes the second call-leg to the same trunk group that is being used for the incoming call.
2. When using the NCR MCI NCT feature, for the Route-Pattern screen associated with the ARS digit analysis form entry, administer the Service/Feature and Number Format fields to be consistent with the service-type and dialing-plan configuration of the PSTN trunk by administering the following settings in an entry line on the lower-right part of the route pattern screen:

- Service/Feature field = **sdn**
- Number Format = **lev0-pvt**

For the other NCT-type NCR protocols, no administration is required for the Route-Pattern screen associated with the ARS digit analysis screen entry. NCR call-processing will automatically cause the Service/Feature and Number Format for the NCR second-leg call to be **unknown/unknown**.

3. Contact the PSTN service provider to verify that the configuration of the PSTN switch used for the Network Call Transfer operation has been properly configured. The PSTN switch should be configured to accept the outgoing digits used by the station or ASAI application to set up the second leg of the call transfer/conference.

Reserving trunk group B-channels for NCT-type redirection operations

Trunk groups that are used for NCR NCT-type operations must be administered as Call-By-Call (CBC) trunk types. Since CBC trunk groups can carry both incoming and outgoing call traffic, situations may occur in which transient levels of incoming traffic occupy all available B-channels. When no B-channels are available for outgoing calls, attempts to set up the outgoing leg for a redirected call will fail and the call redirection will fail.

Important:
When the NCR feature is used with high volumes of incoming calls, Avaya recommends reservation of a minimum number of trunk members for the outgoing leg of redirected calls by using CBC Trunk Group Allocation administration. However, the optimum number of trunk members to reserve depends on traffic patterns that are specific to each call center. A call traffic analysis should be performed to determine if reservation of B-channels is necessary.

If a trunk group has multiple D-channel signaling groups, the CBC Trunk Group Allocation operation does not guarantee the reservation of outgoing trunks associated with a particular D-channel. Instead, it reserves outgoing trunks considering the entire trunk group. Therefore, when NCR invocation is attempted for a trunk group having multiple D-channels, the CBC Trunk Group Allocation operation may not always prevent the blockage of the NCR second call leg setup due to no available outgoing trunk B-channels.

To reduce the blockage of NCR NCT-type operations due to no available outbound trunk B-channels:

- The Network Facilities screen must include one or more ISDN services or features that can be associated with trunk groups that are used for NCR calls.

Note:
When you administer an ISDN service or feature, you must also administer the **Incoming Call Handling Treatment** page on the Trunk Group screen. For more information, see *Administrator Guide for Avaya Communication Manager*.

- On the **CBC Trunk Group Allocation** page of the Trunk Group screen, minimum and maximum values must be specified for trunk members allocated to the designated service or feature.

About network facility types : Before you can specify a minimum number of trunk group members to be allocated for the outgoing legs of NCR calls, you must administer one or more ISDN services or features for this purpose. The Network Facilities screen includes two pre-defined features and ten predefined services. These predefined entries are associated with either Network Specific Facilities (NSF) Type 0 or Type 1. You can administer additional user-defined services or features on the Network Facilities screen. User-defined facilities can be Type 0, 1, 2, or 3. You must obtain support agreements with your PSTN service provider for Type 0 or Type 1 facilities.

Type 2 (incoming) and Type 3 (outgoing) facilities do not use NSF codings or require special support by the PSTN. These network facility types are offered because NSF information is not included with ISDN calls in some regions of the world.

Important:
If your PSTN does not support NSF, you must specify a Type 3 facility when you reserve trunk members for NCR operations, and the Usage Allocation Enhancements Optional Feature must be enabled before you can administer a Type 3 facility.

Example trunk allocation for PSTNs that supports NSF codings: The following example Network Facilities screens includes the basic default pre-defined services and features.

```
change isdn network-facilities                              Page   1 of   2
                          NETWORK-FACILITIES
                          Facility                          Facility
        Name             Type  Coding          Name         Type  Coding
    sub-operator          0     00110       mega800          1     00010
    operator              0     00101       megacom          1     00011
    outwats-bnd           1     00001       inwats           1     00100
    sdn                   1     00001       wats-max-bnd     1     00101
    accunet               1     00110       lds              1     00111
    i800                  1     01000       multiquest       1     10000
    ____________          _     _____
    ____________          _     _____
    ____________          _     _____
    ____________          _     _____
    ____________          _     _____
    ____________          _     _____
```

Once network facilities are specified, trunk members can be allocated on the basis of specific facilities or features. The following example shows a CBC Trunk Group Allocation screen for a CBC trunk group for which at least one B-channel is always available for the outgoing legs of redirected calls when the mega800 service is used. The specific feature or service that you specify in this screen depends on the support provided by your PSTN.

```
change trunk-group 29
                                  CBC TRUNK GROUP ALLOCATION

Usage Allocation Plan 1        Usage Allocation Plan 2        Usage Allocation Plan 3

                Min# Max#                      Min# Max#                      Min# Max#
Service/Feature Chan Chan       Service/Feature Chan Chan       Service/Feature Chan Chan
mega800           1   99
```

Example trunk allocation for PSTNs that do not supports NSF codings: The following example Network Facilities screens includes the basic default predefined services and features and an additional user-defined, Type 3 (outgoing) feature (bsr-redirect).

```
change isdn network-facilities                                  Page   1 of   2
                              NETWORK-FACILITIES
                       Facility                              Facility
        Name           Type  Coding          Name           Type  Coding
    sub-operator        0    00110       mega800             1    00010
    operator            0    00101       megacom             1    00011
    outwats-bnd         1    00001       inwats              1    00100
    sdn                 1    00001       wats-max-bnd        1    00101
    accunet             1    00110       lds                 1    00111
    i800                1    01000       multiquest          1    10000
    bsr-redirect        3    _____
    ____________        _    _____
    ____________        _    _____
    ____________        _    _____
    ____________        _    _____
    ____________        _    _____
```

After the user-defined feature is administered, you can specify a minimum number of reserved trunk channels to remain available for the outgoing legs of redirected calls when the feature is used.

```
change trunk-group 42
                                  CBC TRUNK GROUP ALLOCATION

Usage Allocation Plan 1        Usage Allocation Plan 2        Usage Allocation Plan 3

                Min# Max#                      Min# Max#                      Min# Max#
Service/Feature Chan Chan       Service/Feature Chan Chan       Service/Feature Chan Chan
bsr-redirect      5   25
```

Administering NCR with AT&T In-Band Transfer and Connect

The following sections describe general administration requirements, administration of DTMF announcement and BSR vectoring methods that are associated with use of the AT&T In-Band Transfer and Connect service.

Note:
For a description of NCR administration requirements when this AT&T service is *not* used to invoke NCR, see NCR administration on page 397.

This section includes the following topics:

- General administration for AT&T In-Band Transfer and Connect on page 404
- Setting up DTMF announcements for AT&T In-Band Transfer and Connect on page 406
- Call vectoring methods used with AT&T In-Band Transfer and Connect service on page 407

General administration for AT&T In-Band Transfer and Connect

The following tables show basic administration requirements associated with the AT&T In-Band Transfer and Connect service.

NCR administration - verify NCR customer option

Administration command:	`display system-parameters customer options`	
Page name:	Call Center Optional Features	
Required field(s):	Network Call Redirection	y

NCR administration - BSR Application Plan entries for polling and interflow locations[1]

Administration command:	`change best-service-routing x`	
Page name:	Best Service Routing Application	
Required field(s):	Net Redir?	n

1. Required only if the BSR feature is enabled. For more information, see Call vectoring methods used with AT&T In-Band Transfer and Connect service on page 407.

NCR administration - trunk group screen

Administration command:	**change trunk-group x**	
Page name:	Trunk Group	
Required field(s):	Group Type:	ISDN
	Supplementary Services Protocol:	a
Page name:	Trunk Features	
Required field(s):	UU IE Treatment:	shared
	Network Call Redirection	none

NCR administration - signaling screen

Administration command:	**change signaling group *x***	
Page name:	Signaling Group	
Required field(s):	Network Call Transfer:	y

NCR administration - DS1 screen

Administration command:	**change ds1 [*board location*]**[1]	
Page name:	DS1 Circuit Pack	
Required field(s):	Country Protocol:	1b or 1d

1. Board location parameter values are: [cabinet(1-1)];carrier(A-E);slot(0-20) OR [gateway(1-10)];module(V1-V9).

Setting up DTMF announcements for AT&T In-Band Transfer and Connect

You can create the announcement that provides the DTMF digits required for AT&T In-Band Transfer and Connect operations by either of the following methods:

Note:
For information about how DTMF announcements are used in vectors to implement NCR, see AT&T In-Band Transfer and Connect on page 383.

Also see Announcement recording tips for high traffic volume applications on page 541.

- Use an Avaya Communication Manager analog DTMF station to activate the record session for a specific announcement. When the record session starts, use the keypad to enter the Touch-Tone digits that correspond to the *T + PSTN endpoint number that is used to invoke AT&T In-Band Transfer and Connect operations. For example, if feature invocation is intended to redirect an incoming ISDN call to specified endpoint number 3035552104, then enter: *83035552104 when the announcement recording session begins.

Note:
You cannot use a digital phone (such as Callmaster, BRI, ISDN or IP) to record the announcement, since the station keypads do not generate audible DTMF tones during an announcement record session. If you record DTMF with these phones, use local arrangements to electronically connect an external keypad.

- Use a PC with VAL boards with an internal or external keypad or a commercially-available PC software tool.

Note:
To achieve the best recording quality, use local arrangements to electronically connect the external keypad. Do not acoustically couple the external keypad.

Call vectoring methods used with AT&T In-Band Transfer and Connect service

The AT&T in-band AT&T In-Band Transfer and Connect feature can be invoked by the Best Service Routing (BSR) feature. To do this, administer an *Interflow VDN* in the Best Service Routing Application table. This routes to a near-end switch VDN that causes a **T announcement* vector step to be executed rather than setting this same *Interflow VDN* number to route directly to a VDN on a far-end Avaya switch.

When a BSR polling vector identifies a BSR best location to which to route an incoming call, the BSR location must be administered on the BSR Application Plan screen. The BSR Application Plan must meet the following requirements:

- The plan must include one or more interflow VDNs that are associated with vectors that include the vectors steps necessary for successful invocation of the AT&T In-Band Transfer and Connect feature.
- The **Net Redir?** field associated with a location where AT&T In-Band Transfer and Connect is used must be set to **n**.

Example BSR implementation: The following example shows how BSR can be used with AT&T In-Band Transfer and Connect to implement call redirection. In the example scenario, *local* rather than *remote* interflow VDN numbers are assigned to the BSR application plan screen.

The example application plan is shown below:

```
                  BEST SERVICE ROUTING APPLICATION

Number: 1    Name:        Maximum Suppression Time: 60      Lock? y

Num  Location Name   Switch Node     Status Poll VDN   Interflow VDN    Net Redir?
1    Omaha             320           95022011          4004                n
2    Paris             320           95022111          4005                n
3    Sydney            320           95032211          4006                n
```

The example application plan shown above lists VDN extension numbers that are local to the communication server. Each of the VDNs are associated with vectors that are designed to invoke AT&T In-Band Transfer and Connect operations.

Each of the vectors associated with the interflow VDNs listed in the application plan includes the elements shown in the following example.

```
1. announcement 1234 [*8 plus PSTN number for remote site]
2. wait 2 seconds hearing silence
3. disconnect after announcement none
```

In the vector example shown above, step 1 provides the extension for an announcement that plays the DTMF digits, as described in Setting up DTMF announcements for AT&T In-Band Transfer and Connect on page 406.

Step 2 provides a wait step that is included to give the PSTN switch sufficient time to process the in-band information (sent by the announcement in the preceding step) before the call is disconnected at step 3. The **disconnect** command in step 3 sends an ISDN DISCONNECT message that includes the Information Forwarding data for the call in a codeset 0 or 7 UUI IE element. For more information about Information Forwarding, see NCR and Information Forwarding on page 387.

Important:
The type of Information Forwarding data sent to the PSTN depends on how the **UUI IE Treatment** field on the **TRUNK FEATURES** page of the Trunk Group screen is administered:

If the **UUI IE Treatment** field is set to **Service Provider**, the ASAI user data is forwarded to the PSTN in the ISDN DISCONNECT message. If the **UUI IE Treatment** field is set **Shared**, the call center-related data described in NCR and Information Forwarding on page 387 is forwarded to the PSTN in the ISDN DISCONNECT message.

Note that the same above call vector example that invokes the AT&T In-Band Transfer and Connect feature using BSR *VDN Interflow* vector processing can also be used to invoke the AT&T In-Band Transfer and Connect feature with non-BSR vector programming. The same guidelines and notes related to the **T announcement* format apply to executing this example vector in a non-BSR vector call-flow.

NCR troubleshooting

You can use the following methods and resources to analyze NCR problems:

- When NCR and BSR are both implemented, your first troubleshooting step should be to verify that no problems exist with BSR polling and interflow operations when NCR is *not* administered on the BSR Best Routing Application screen. After any problems are identified and resolved, set the **Net Redir?** files to **y** on this screen for all locations where NCR is used, and then verify that NCR works properly.
- The ISDN message trace information provided by the Message Sequence Tool (MST) for the ISDN trunk D-channel associated with NCR invocation attempts. The steps to configure MST for NCR troubleshooting are as follows:
  - Enter the **ch MST** Switch Administration Terminal command, then on page 1 set the **ISDN-PRI?** field to **Y**, and on page 2 set the ISDN-PRI Filter Data **Port Type** field to **d-channel** and the **Port** field to the DS1 D-channel switch equipment location associated with the PRI trunk being used with the NCR feature.

- Use the **enable mst** and the **list mist cont** Switch Administration Terminal commands to see NCR-related MST trace data.
- When a NCR NCT-type invocation is initiated by vector processing operation or by a manual call-transfer or call-conference/release operation, a D-channel message is sent to the PSTN switch by the Communication Manager to initiate the merging of the two B-channels associated with the first and second call-legs of a trunk-to-trunk call. The following MST trace example is for a NCR Two B-Channel Transfer D-channel invocation message that has the same general format as for the MCI NCT, ETSI ECT, or NCD protocols:

```
<msg #> 62 <time stamp> 40 01 18 0F 08 02 80 02 62 1C 09 91 A1 06 02 01
                04 02 01 06
```

  Look for the **91 A1** data-byte sequence shown in bold characters above to verify that a NCR invocation D-channel message is being sent by the Communication Manager.

- If the NCR NCT-type invocation is successful, the PSTN switch will return a D-channel message to the Communication Manager that has the following general format:

```
<msg #> 60 <time stamp> 00 00 4B 17 08 02 93 E5 62 1C 06 91 A2 03 02 01
               01
```

  Look for the **91 A2** data-byte sequence shown in bold characters above to verify that the PSTN switch accepted the NCR invocation request. A D-channel message instead sent by the PSTN switch that has 91 A3 or 91 A4 data-byte sequence indicates the NCR invocation attempt was rejected. Use the **display events** System Administration Terminal command to see vector events that will explain why the NCR invocation failed.

- For the NCR ETSI ECT protocol, a NCR `Request LinkID` D-channel message is first sent to the PSTN switch by the communication server to determine which D-channel to use for this NCR ETSI ECT invocation: This will result in the PSTN sending a `Returned LinkID` D-channel message to the communication server, where an example of an Ericsson AXE-10 single-byte LinkID MST message is as follows:

```
<msg #> 60 <time stamp> 40 01 18 0F 08 02 00 57 62 1C 13 91 A2 10 02 01
              0B 30 0B 06 06 04 00 82 71 01 04 02 01 FE
```

- The Communication Manager next will send an `Invoke Explicit ECT` D-channel message to the PSTN switch using the LinkID returned by the PSTN switch, where an example Ericsson AXE-10 single-byte LinkID MST message is as follows:

```
<msg #> 62 <time stamp> 40 01 18 0F 08 02 01 92 62 1C 11 91 A1 0E 02 01
               0C 06 06 04 00 82 71 01 01 02 01 FE
```

— For any of the NCR NCT-type protocols, a successful invocation results in both legs of the trunk-to-trunk connection being dropped by the PSTN switch after the B-channels are merged. An example of the PSTN switch first dropping the second call-leg by sending a `Disconnect`, the Avaya switch sending back a `Release`, and the PSTN switch sending a `Release Complete` D-channel message is as follows:

```
<msg #> 60 <time stamp> D40 01 18 0F 08 02 81 92 45 08 02 82 90 1C 23 91
                        DA1 20 02 02 00 80 02 01 22 30 17 A1 0F 30 06 02
                        D01 00 02 01 01 30 05 05 00 02 01 02 82 01 00 83
                        D01 00 1C 06 91 A2 03 02 01 0C
<msg #> 62 <time stamp> 40 01 18 0F 08 02 01 92 4D
<msg #> 60 <time stamp> 40 01 18 0F 08 02 81 92 5A
```

An example of the PSTN switch completing the NCR call-redirection operation by dropping the first call-leg by sending a `Disconnect`, the Avaya switch sending a `Release`, and the PSTN switch sending a `Release Complete` D-channel message.To verify the called number information associated with the NCR setup of the second call-leg is correct and to see trunk-related denial events that may be generated if the NCR fails, use the **`list trace tac <trunk group number>`** Switch Administration Terminal command

- To see the behavior of a particular VDN or vector, use the **`list trace vdn`** and **`list trace vector`** Switch Administration Terminal commands to check for NCR errors.
- To check for NCR errors using BSR processing:
  — If you are logged in at the Switch Administration Terminal (SAT) using the init login, enter **`go tcm`**
  — When the `tcm1>` prompt is received, enter the **`rdd:dp_mgr Bsr_applloc`** command to see the total NCR attempts, internal errors, network errors, successful redirections, and disconnects peg counts that are associated with BSR call interflows where NCR was invoked.

  These peg counts are free-running and are only reset when the BSR Best Service Routing Application screen is accessed using the **`ch best`** SAT command for a particular BSR application number.
- If NCR vector invocation by call vectoring has failed for previous calls, use the **`display events`** SAT command to obtain a real-time display of vector events that may be logged for call redirection attempts. The possible NCR vector events are as follows:
  — 68: Adjunct Route via NCT failed
  — 310 NCR: Invoke trunk not ISDN
  — 311 NCR: Bad NCR trunk admin
  — 312 NCR: No NCT PSTN service
  — 313 NCR: No NCT outgoing trk

- 314 NCR: NCT outgo trk drop
- 315 NCR: PSTN NCT invoke err
- 316 NCR: PSTN NCT netwrk err
- 317 NCR: Used NCT trk-to-trk
- 318 NCR: No NCD PSTN service
- 319 NCR: NCD invalid PSTN nmbr
- 320 NCR: NCD call connect err
- 321 NCR: PSTN NCD invoke err
- 322 NCR: PSTN NCD netwrk err
- 323 NCR: PSTN NCD max redirs
- 324 NCR: PSTN NCD no disc
- 325 NCR: Internal system err

Other things to know about NCR failure scenarios

The following items may be relevant in case of failed NCR operations:

Failure to invoke NCT-type operations: If an NCT-type operation is invoked and the PSTN switch rejects the NCT-type operation, the call will revert to a trunk-to-trunk transfer. In this case, the call is not lost but further vector processing stops with the failure to invoke the redirection operations.

Failure to invoke Network Call Deflection: If NCD is invoked and the PSTN switch rejects the call, the call is disconnected from the redirecting communication server and no further vector processing occurs.

Network Call Redirection with SIP

This section includes the following topics:

- About NCR support with SIP on page 413
- SIP Network Call Redirection protocols on page 414
- SIP limitations on call redirection on page 414
- SIP NCR and Information Forwarding on page 414
- SIP NCR feature interactions on page 415
- Summary of call vectoring-activated NCR operations on page 417
- For NCR applications see NCR implementation methods on page 390
- BSR queue-to best vector step activation of NCR on page 417
- NCR administration for SIP on page 419
- NCR troubleshooting on page 420

About NCR support with SIP

Network Call Redirection (NCR) provides an Avaya communication server call routing method between sites on a public network or a Virtual Private Network (VPN) that can reduce trunking costs. These cost reductions are particularly valuable in enterprises or multi-site call center environments where trunk costs are high.

When an incoming call arrives at an Avaya communication server that has the NCR feature enabled, call redirection is managed by the SIP service provider or VPN switch instead of the local Avaya server. As a result, trunks that the server would otherwise retain to accomplish a trunk-to-trunk transfer are released after the call redirection takes place.

The cost reductions associated with reduced trunk use can be significant particularly when Avaya virtual routing features, such as Best Service Routing (BSR) with Expected Wait Time (EWT), are implemented. The cost-savings are achieved by the Avaya customer requiring fewer trunks to handle the same number of incoming/outgoing calls after the NCR feature is implemented within the local communication server.

SIP Network Call Redirection protocols

NCR can occur over SIP trunks. The SIP REFER or SIP 302 Moved Temporarily messages contain all the information needed for Network Call Redirection. When the call is answered by an agent or call vectoring event, a SIP REFER message is sent. If a SIP call is not answered by an agent or a call vector process, a SIP 302 Moved Temporarily message is sent.

SIP limitations on call redirection

You should understand the following items that pertain to SIP limitations on the NCR feature:

NCR feature support: SIP service provider support for NCR varies with geographical location and may be limited or absent in some areas. Consult your Avaya account team to determine SIP service provider availability of one of the NCR protocols in your area.

Allowable number of redirection per call: There may be limits placed on the number of times a call may be redirected over the public network. These limits are imposed by the public network service provider. There might be additional charges associated with redirected calls.

User-to-User information forwarding support: Some public network service providers do not support forwarding of User-to-User Information (UUI), including Adjunct Switch Application Interface (ASAI) user data, collected digits, VDN name, the VDN in-time (as reflected by the NETINTIME database items), and the UCID. In such situations, Information Forwarding will be lost and the second leg of the redirected call will look like an entirely new call to the redirected-to server at the second location.

One of the data items lost is the VDN name, which is rerouted to the originally called service (DNIS) information. The indication that the call has been forwarded can be achieved by using dedicated VDNs for call forwarding, but this strategy loses the benefits of Information Forwarding inherent with NCR and limits use of CTI applications.

SIP NCR and Information Forwarding

The Avaya Information Forwarding feature is supported with NCR when the SIP service provider supports UUI transport in conjunction with the specific network redirection protocol used by the switch.

This section includes the following topics:

- UUI data included in Information Forwarding on page 415

- UUI data forwarding on page 415

UUI data included in Information Forwarding

Information Forwarding forwards the following call center-related data (as User-to-User Information) with a SIP call:

- Adjunct Switch Application Interface (ASAI) user data
- Universal Call ID (UCID)
- Collected digits
- In-VDN time
- VDN name.

UUI data forwarding

When NCR is used, the UUI is forwarded by the Avaya communication server in the SIP REFER or 302 Moved Temporarily messages.

The UUI is forwarded by the Avaya communication server in a SIP REFER message if the call has been answered. A call can be answered for example, if an agent answers the call, a vector plays an announcement or music, or the call is processed by a command that provides an answer such as collect digits. The REFER is sent back to the caller, causing the first call to be dropped after a second call is issued and established with the the next location.

The UUI is forwarded by the Avaya communication server in a 302 Temporarily Moved message if NCR is inoked before the call is answered. The first call is redirected to the next location.

SIP NCR feature interactions

NCR interacts with the following call center features:

- Attendant Vectoring — Attendant Vectoring can use the **`route-to number`** vector step to route calls to attendants located at another communication server node. The operation of the NCR feature using the network call redirection features to accomplish the call redirection is exactly the same as for redirecting ACD calls.

  For more information, see Using the route-to command for NCR on page 623.

- BCMS — No change is made to BCMS for support of NCR. Redirected calls are tracked as completed calls since the SIP service provider disconnects the incoming facility of the original call when the call is redirected to another site.

- Enhanced Information Forwarding — For the NCR feature, Enhanced Information Forwarding transports User-to-User information (UUI) for the incoming call to the SIP service provider endpoint that receives the redirected call. The use of the Enhanced Information Forwarding capability with NCR (the recommended configuration) requires that the incoming call trunk group be assigned as shared (i.e., the **UUI treatment** field is set to shared). However, if the trunk group is set up as service provider, only the ASAI user information (or user information provided by the incoming call) will be included in the UUI sent on a non-shared basis to the redirected-to SIP service provider endpoint.
- Look-Ahead Interflow — NCR activation using the **route-to number** vector step does not require Look-Ahead Interflow to be active to provide multi-site capabilities, which are required for considering remote locations and access to the BSR Application Plan screen.
- Service Observing by VDN — If the Service Observing by VDN feature is used to service observe a VDN, where the NCR feature is used to redirect incoming calls, the service-observer will hear the same tones, music, and/or announcements heard by the incoming caller before the NCR feature reroutes the call to another SIP service provider endpoint. When the NCR operation is completed, the service-observer will be dropped as an observer of the incoming call and placed in the service-observing queue associated with the VDN.
- Trunk-to-Trunk Transfer — If the NCR feature is optioned and the ASAI Third-Party make Call/transfer operation is used to redirect an incoming call to a SIP service provider endpoint, the **Trunk-to-Trunk Transfer** field on the System-Parameter Features form must be enabled for the call redirection to succeed. If the **route-to number** or BSR **queue-to best** vector step uses the NCR feature to redirect an incoming call to a SIP service provider endpoint, the Trunk-to-Trunk Transfer customer option does not need to be set to y.

  For more information, see Using the route-to command for NCR on page 623.
- VDN Return Destination — If the VDN Return Destination feature is administered for the VDN that is associated with a vector that causes the NCR feature to be invoked, the VDN Return Destination feature will be canceled when the call is redirected by NCR.
- CMS database items — The following Avaya CMS database items are affected by NCR:
  - DEFLECTCALLS: In the VDN CMS database tables, the DEFLECTCALLS item includes the number of calls that are redirected using NCR through the BSR feature by using the **route-to number** or **queue-to best** commands. Successful NCR attempts are pegged as DEFLECTCALLS.
  - INTERFLOWCALLS: In the VDN CMS database tables, the INTERFLOWCALLS item includes successful BSR interflows using NCR redirections.
  - LOOKATTEMPTS: In the VDN CMS database tables, the LOOKATTEMPTS item includes the number of times the Look-Ahead Interflow or BSR interflow was attempted for calls in the vector. Successful Look-Ahead Interflow or BSR attempts are also counted. NCR invoke attempts are also reflected in LOOKFLOWCALLS.

– LOOKFLOWCALLS: In the VDN CMS database tables, the LOOKFLOWCALLS item includes the number of INTERFLOWCALLS that were redirected by the Look-Ahead Interflow or BSR features. LOOKFLOWCALLS is a subset of INTERFLOWCALLS and includes LOOKATTEMPTS for the Look-Ahead Interflow or BSR interflows. With BSR interflow using trunk-to-trunk transfer or NCR, every LOOKATTEMPT will also be counted as a LOOKFLOWCALLS unless a failure occurs.

Summary of call vectoring-activated NCR operations

The processes by which NCR is implemented by a call vectoring method with SIP is summarized in the following steps:

1. The SIP service provider switch sends an incoming call to the Avaya communication server, where the call enters vector processing.
2. Call processing proceeds to either a **`route-to number ~r`** or BSR **`queue-to best`** vector step.

BSR queue-to best vector step activation of NCR

NCR is especially useful for multi-site call center operations in which the Best Service Routing feature is enabled, since the number of trunk resources needed for call interflows is reduced. This method provides the best approach for balancing loads across a multi-site environment and is more cost effective and accurate than pre-delivery routers.The **`queue-to best`** vector step can be used to interflow calls between communication servers over the SIP service provider network. You can use SIP to interflow BSR calls, but BSR polling is not supported over SIP trunks. Instead BSR polling can be done using other methods, such as Polling Over IP without B-Channel. For more information about BSR, see Best Service Routing (BSR) on page 297.

NCR is activated by the **`queue-to best`** vector step when the BSR feature determines a BSR location is the best BSR location and that location is administered with the **Net Redir?** option set to **y** on the BSR Application Table screen. Note that the administered **Interflow VDN** field on the Best Service Routing Application screen must be a SIP service provider or VPN endpoint number without a trunk/ARS/AAR access codes included. For some SIP service provider switch dialing plans, the long-distance access code (for example, a "1" in the United States) must be prefixed to the SIP service provider number for the call to be succesfully routed by the SIP service provider switch.

As shown in the following example, the Best Service Routing Application Plan screen must include locations that have the **Net Redir?** field set to **y**.

```
                  BEST SERVICE ROUTING APPLICATION

Number: 1    Name:        Maximum Suppression Time: 60      Lock? y

Num  Location Name   Switch Node    Status Poll VDN  Interflow VDN    Net Redir?
1    Omaha                          95552011       3035551211           y
2    Paris                          95552022       18005551234          y
3    Sydney                         95552033       18665553456          y
```

An appropriate vector is then used to identify a BSR best location and NCR is activated by the **queue-to-best** vector step.

```
wait 2 seconds hearing ringback
consider skill 1 pri l adjust-by 0
consider location 1 adjust-by 20
consider location 2 adjust-by 40
consider location 3 adjust-by 20
queue-to best
```

NCR administration for SIP

NCR administration - verify customer options for NCR

Administration command:	`display system-parameters customer options`	
Page name:	Optional Features	
Required field(s):	G3 Version	V15 or later
	ISDN/SIP Network Call Redirection	y

NCR administration - BSR screen[1]

Administration command:	`change best-service-routing x`	
Page name:	Best Service Routing Application	
Required field(s):	Net Redir?	y

1. Required only if the BSR feature is enabled.

NCR administration - trunk group screen

Administration command:	`change trunk-group x`	
Page name:	Trunk Group	
Required field(s):	Direction:	two-way
	Group Type:	SIP
Page name:	Protocol Variations	
Required field(s):	Network Call Redirection?	y

NCR troubleshooting

You can use the following methods and resources to analyze NCR problems:

- When a NCR NCT-type invocation is initiated by a vector processing operation or by a manual call-transfer or call-conference/release operation, a SIP REFER or 302 Moved Temporarily message is sent to the SIP service provider by the Communication Manager to initiate the redirection operation.

  You can determine what User-to-User data was sent in the SIP REFER or 302 Moved Temporarily messages, but the absense of User-to-User information does not mean that an invocation failed, there might not be any User-to-User info at the time the message was generated.

  The following examples are for SIP NCR invocation messages:

  — A successful SIP NCR invocation with a REFER message:

```
REFER sip:30341@135.9.72.61;transport=tcp SIP/2.0^M
From: "3322" <sip:3322@avaya.com>;tag=0b6ce3afb70dc14f047524900^M
To: "ISDN 2" <sip:30341@avaya.com>;tag=0b6ce3afb70dc19a0474ff5700^M
Call-ID: 0b6ce3afb70dc19b0474ff5700^M
CSeq: 1 REFER^M
Max-Forwards: 70^M
Route: <sip:123.4.72.61;lr;transport=tcp>^M
Via: SIP/2.0/TCP 123.4.72.136:5062;branch=z9hG4bK8010313dfb70dc150047524900^M
User-Agent: Avaya CM/R015x.00.0.822.f^M
Contact: "test vdn" <sip:3322@135.9.72.136:5062;transport=tcp>^M
Refer-To: <sip:825030340@avaya.com?User-to-User=00C810313233343536373838393031323
343536F7020007F803042143%3Bencoding%3Dhex>^M
Content-Length: 0^M
^M
```

  A successful invocation is indicated by the User-to-User information for the `Refer-To` header.

  The REFER method of NCR invocation is used for the case where the call has already been answered either in the vector or by an answering station.

  Failure of the SIP NCR invocation with vector processing will results in continuation of vector processing at the following step.

— A successful SIP NCR invocation with a 302 Moved Temporarily response:

```
SIP/2.0 302 Moved Temporarily^M
From: "ISDN 2" <sip:30341@avaya.com>;tag=0f0a1affb70dc1ac0474ff5700^M
To: "3322" <sip:3322@avaya.com>;tag=0f0a1affb70dc15e047524900^M
Call-ID: 0f0a1affb70dc1ad0474ff5700^M
CSeq: 1 INVITE^M
Via: SIP/2.0/TCP 123.4.72.61;branch=z9hG4bK0f0a1affb70dc1ae0474ff5700^M
Server: Avaya CM/R015x.00.0.822.f^M
Contact:
<sip:825030340@avaya.com?User-to-User=00C810313233343536373839303132333435
36F7020002%3Bencoding%3Dhex>^M
Content-Length: 0^M
^M
```

A successful invocation is indicated by the User-to-User information for the `Contact` header.

- The 302 Moved Temporarily method of NCR invocation is used for the before answer case that only applies within vector processing. Failure of the SIP NCR invocation in this case will result in the stopping of vector processing.
- To verify the called number information associated with the NCR setup of the second call-leg is correct and to see trunk-related denial events that may be generated if the NCR fails, use the **`list trace tac <`*`trunk group number`*`>`** Switch Administration Terminal command
- To see the behavior of a particular VDN or vector, use the **`list trace vdn`** and **`list trace vector`** Switch Administration Terminal commands to check for NCR errors.
- If NCR vector invocation by call vectoring has failed for previous calls, use the **`display events`** SAT command to obtain a real-time display of vector events that may be logged for call redirection attempts. The possible NCR vector events are:
  - -68: Adj Route via NCR failed
  - -311: Bad NCR trunk admin
  - -312: NCR: No NCT service
  - -316: NCR: NCT netwrk err
  - -318: NCR: No service
  - -319: NCR: invalid num
  - -322: NCR: netwrk err
  - -325: NCR: Internal system err
  - -327: NCR: Caller not SIP trk

Attendant Vectoring

This section includes the following topics:

- About Attendant Vectoring on page 423
- Command set on page 424
- Overview on page 430
- Attendant Vectoring and attendant VDNs on page 437
- Attendant Vectoring and multiple queueing on page 438
- Considerations on page 439

About Attendant Vectoring

The Attendant Vectoring feature enables a set of commands that can be used to write call vectors for calls to be routed in non-call center environments. When Attendant Vectoring is enabled, all attendant-seeking or dial 0 calls are processed using the call vectors, not the normal attendant console call routing.

The main reason to use Attendant Vectoring is to allow flexible routing of attendant-seeking calls. If users are instructed to dial an attendant VDN, the call could be answered by an attendant, but it may also be covered to the voice mailbox of a night station. Training users to understand these different call routing options is something you should consider before using Attendant Vectoring.

If you use Attendant Vectoring and night service to route calls to a voice mail system, you can also use the Automatic Message Waiting (AMW) feature to notify after-hours personnel that there are messages in the night service station mailbox by assigning an AMW lamp on one or more backup telephones. When personnel see that there are new messages, they can check those messages after hours and act upon them as needed.

Command set

The following table lists the commands associated with Attendant Vectoring.

Attendant vectoring command set

Command category	Action taken	Command
Treatment		
	Play an announcement.	`announcement`
	Play a busy tone and stop vector processing.	`busy`
	Disconnect the call.	`disconnect`
	Delay with audible feedback of silence, ringback, system music, or alternate audio/ music source.	`wait-time`
Routing		
	Queue the call to an attendant group.	`queue-to attd-group`
	Queue the call to an attendant extension.	`queue-to attendant`
	Queue the call to a hunt group.	`queue-to hunt-group`
	Route the call to a specific extension number.	`route-to number`
Branching/programming		
	Go to a vector step.	`goto step`
	Go to another vector.	`goto vector`
	Stop vector processing.	`stop`

Treatment commands

Attendant Vectoring allows use of several TREATMENT commands, including:

- *announcement command on page 425*
- *busy command on page 425*
- *disconnect command on page 425*
- wait-time command on page 425

The following sections detail the syntax that can be used for these commands and any information that is specific to their use in Attendant Vectoring.

announcement command

Syntax

```
announcement <extension>
```

The usage for the **announcement** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

busy command

Syntax

```
busy
```

The usage for the **busy** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

disconnect command

Syntax

```
disconnect after announcement <extension>
```

The usage for the **disconnect** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

wait-time command

Syntax

```
wait-time <time> secs hearing <silence, ringback, music>
```

This use of the **wait-time** command was slightly modified for attendant vector usage. The **i-silent** treatment choice was removed because it does not pertain to attendant vectoring. The **wait-time <seconds> secs hearing <extension> then <silence, ringback, music, continue>** command was left unchanged. No other changes or attendant specific considerations apply, so these commands work as they do in Basic Call Vectoring.

Routing commands

Attendant Vectoring allows use of several ROUTING commands, including:

- *queue-to attd-group command on page 426*
- *queue-to attendant command on page 427*
- *queue-to hunt-group command on page 428*
- route-to number command on page 428

Note:
A **wait-time 0 secs hearing ringback** step should be used to give immediate feedback to the caller. The **queue-to** command does not provide ringback until the call is actually ringing the attendant. The **wait-time** step should be implemented as the first vector step or as the step immediately before the **queue-to** step.

The following sections detail the syntax that can be used for these commands and any information that is specific to their use in Attendant Vectoring.

queue-to attd-group command

Syntax

queue-to attd-group

The **queue-to attd-group** vectoring command is available only for attendant vectors. If an attendant group call is redirected to vector processing that queues the call to the attendant group, the group to which the call gets queued is determined by the TN assignment that is associated with the call. If an attendant in the group is available to take the call, it is terminated to the attendant, not queued, and vector processing terminates.

Attendant group based on tenant number

When attendant group calls are redirected to vector processing and are programmed to queue to the attendant group, the attendant group is the group that is designated for the call's associated tenant number.

If an attendant group call is redirected to vector processing that queues the call to the attendant group, the call is placed in the queue using the priority that is assigned for the call. Attendant queue priorities are assigned on a system-wide basis, not on an individual partition basis.

Attendant group queue

Calls that are queued to the attendant group by way of attendant vector processing are queued with the system-administered priority for the call. If an attempt is made to queue the call and it fails, the vector event for queue failure is logged.

As with other vector queue commands, vector processing continues with the next step following the **queue-to attd-group** command regardless of success or failure. The **goto step if queue-fail** command is provided for handling failure conditions. Otherwise, on success, announcements or other feedback can be applied while the call is in queue. Other than the provision of caller feedback, attendant queue functionality is unchanged. If no commands follow a successful queue step, the call is left in the queue with no feedback. If no commands follow a failed queue step, the call is dropped. Anytime the end of vector processing is reached without the call being placed in queue, it is dropped and an event is logged.

queue-to attendant command

Syntax

queue-to attendant <extension>

The **queue-to attendant** vectoring command is available only for attendant vectors. If an attendant group call is redirected to vector processing that queues the call to an individual attendant, the attendant to whom the call gets queued must be a member of the attendant group that is indicated by the TN assignment associated with the call. If the attendant is available to take the call, the call is terminated to the attendant, not queued, and vector processing terminates.

The success of this command depends on having individual attendant access. These calls are queued based on the priority that is assigned to individual attendant access calls.

Individual attendant queue

Calls that are queued to the individual attendant using attendant vector processing are queued with the system-administered priority for individual attendant access calls. If the indicated attendant is not a member of the associated attendant group, the command is considered failed and vector processing continues with the next vector step. If an attempt is made to queue the call and it fails, a vector event is logged.

As with other vector queue commands, vector processing continues with the next step following the **queue-to attendant** command regardless of success or failure. The **goto step if queue-fail** command is provided for handling failure conditions. Otherwise, on success, announcements or other feedback can be applied while the call is in the queue. If no commands follow a successful queue step, the call is left in the queue with no feedback. If no commands follow a failed queue step, the call is dropped. Anytime the end of vector processing is reached without the call being placed in queue, the call is dropped and an event is logged.

queue-to hunt-group command

Syntax

queue-to hunt-group <#> pri <l (low), m (medium), h (high), t (top)>

This vectoring command is available only for attendant vectors. However, it is the functional equivalent of the split queueing command. As such, a call can be queued to up to three hunt groups. If an attendant group call is redirected to vector processing that queues the call to a hunt group, the call is queued with the indicated priority. If a hunt group member is available to take the call, it is terminated to the member, not queued, and vector processing terminates. In order to use a hunt group in vectoring, it must be administered as a vector controlled group. However, it can be any type of hunt group, including UCD, ACD, and so forth.

Hunt group queue

Calls that are queued to a hunt group by way of attendant vector processing are queued with the indicated priority for the call. If an attempt is made to queue the call and it fails, a vector event is logged.

As with other vector queue commands, vector processing continues with the next step following the **queue-to hunt-group** command regardless of success or failure. The **goto step if queue-fail** command is provided for handling failure conditions. Otherwise, on success, announcements or other feedback can be applied while the call is in the queue. Since these hunt groups are required to be vector-controlled, announcements are provided by way of vectoring commands and hunt group-specific forced announcements do not apply. If no commands follow a successful queue step, the call is left in the queue with no feedback and vector processing terminates. If no commands follow a failed queue step, the call is dropped. Anytime the end of vector processing is reached without the call being placed in the queue, it is dropped.

route-to number command

Syntax

route-to <number> with cov <y, n> if <unconditionally>

This command is slightly modified from standard usage when used for attendant vectoring and **unconditionally** is the only available option. Existing choices allow routing with **if unconditionally**, **digit**, **name**, or **interflow-qpos**. Since digit comparison and interflow do not pertain to attendant vectoring, the options are not available. No other changes or attendant specific considerations apply. This command works as it does in standard usage. This command is provided by administration that is defined on the Console Parameters screen. Therefore, call processing requirements are not needed.

Syntax

route-to ~r<number>

For incoming calls to the communication server, NCR can be activated using the route-to number vector step, where the number field in the vector step has a ~r in the first digit position. This allows for the route-to number vector step to interflow an incoming attendant call to another communication server over the PSTN since no trunks are tied up at the redirecting switch.

Branching/programming commands

Attendant Vectoring allows use of several branching/ programming commands, including:

- *goto step command on page 429*
- *goto vector command on page 430*
- stop command on page 430

The following sections detail the syntax that can be used for these commands and any information that is specific to their use in Attendant Vectoring.

goto step command

Syntax 1

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

This use of the **goto step** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

Syntax 2

```
goto step <step #> if <unconditionally>
```

This use of the **goto step** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

Syntax 3

```
goto step <step #> if queue-fail and goto vector <vector #> if
  queue-fail
```

These vectoring conditionals are available only for attendant vectors. Any time an attempt is made to queue a call and it cannot be queued, these commands can be used to direct vector processing. For attendant vectoring, there is no attempt to determine whether a call can be queued before attempting to do so. Therefore, one of these commands can be used to provide alternate processing when calls cannot be queued. Some examples of why calls can fail to queue are as follows, but this is not a complete list of the causes of failure:

- The queue is full
- The attendant group is in night service and there is no night console
- The individual attendant is not a member of the associated attendant group
- There were invalid multiple queue attempts. For more information, see Attendant Vectoring and multiple queueing on page 438.

Failure to queue

The queue failure conditional is set following a queue command that fails to queue the call. It always indicates the result of the most recent queue command. If the failure conditional is set, vector processing is redirected as indicated.

goto vector command

Syntax 1

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

The use of the **goto step** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

Syntax 2

```
goto vector <vector #> if unconditionally
```

The use of the **goto step** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

stop command

The use of the **stop** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

Overview

The Attendant Vectoring capability enables you to use certain vector commands in a non-call center environment. For example applications of Attendant Vectoring see Call Vectoring applications on page 57.

Attendant Vectoring is available in non distributed attendant environments and distributed attendant environments for IAS and QSIG CAS.

Vector screen

The following example shows the Call Vector screen with the Attendant Vectoring field enabled.

Call Vector screen

```
change vector xxx                                        page 1 of 3
                     CALL VECTOR

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

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

The Attendant Vectoring field appears only when Attendant Vectoring is enabled on the Customer Options screen. If either Basic Vectoring or Prompting are set to **y**, the Attendant Vectoring field defaults to **n**. If Basic Vectoring, Prompting, and Enhanced Conference are not enabled on the Customer Options screen, the Attendant Vectoring field defaults to **y**, and it cannot be changed to **n**. When the Attendant Vectoring field on the Call Vector screen is set to **y**, that vector is used as an attendant vector.

To associate VDNs and vectors for attendant vectoring, a field on the VDN and the call vectoring screens indicates attendant vectoring. When attendant vectoring is indicated for VDNs and vectors, all call center-associated fields (such as **Skills** and **BSR**) are not displayed.

Console Parameters screen

When Attendant Vectoring is enabled, a field on the Console Parameters screen identifies the assigned Attendant Vectoring VDN. The following examples show the Console Parameters screens.

Console Parameters screen (Page 1)

```
change console-parameters                                    Page   1 of   4
                              CONSOLE PARAMETERS
         Attendant Group Name: OPERATOR
                         COS: 1                                  COR: 1
       Calls in Queue Warning: 1                    Attendant Lockout? y
        Ext Alert Port (TAAS): 01A1216
                         CAS: none
                                                Night Service Act. Ext.: 195
                IAS (Branch)? n                 IAS Tie Trunk Group No.:
          IAS Att. access Code:                   Alternate FRL Station:
               Backup Alerting? y          DID-LDN Only to LDN Night Ext? n
      Attendant Vectoring VDN: 2000
```

Console Parameters screen (Page 2)

```
change console-parameters                                    Page   2 of   4
                              CONSOLE PARAMETERS

TIMING
  Time Reminder on Hold (sec): 30           Return Call Timeout (sec): 30
  Time in Queue Warning (sec): 15

  INCOMING CALL REMINDERS
      No Answer Timeout (sec): 10                       Alerting (sec): 10
                               Secondary Alert on Held Reminder Calls? y

ABBREVIATED DIALING
     List1:                   List2:                   List3: system

                          COMMON SHARED EXTENSIONS
            Starting Extension: 670              Count: 3
```

Console Parameters screen (Page 3)

```
change console-parameters                                    Page   3 of   4
                              CONSOLE PARAMETERS

QUEUE PRIORITIES

               Emergency Access: 1
                Assistance Call: 2
                        CO Call: 2
               DID to Attendant: 2
                       Tie Call: 2
           Redirected DID Call: 2
               Redirected Call: 2
                   Return Call: 2
                   Serial Call: 2
    Individual Attendant Access: 2
                Interpositional: 2
       VIP Wakeup Reminder Call: 2
             Miscellaneous Call: 2

       Call-Type Ordering Within Priority Levels? n
```

Console Parameters screen (Page 4)

```
change console-parameters                                    Page   4 of   4
                              CONSOLE PARAMETERS

ASSIGNED MEMBERS ( Installed attendant consoles )
              Type       Grp  TN            Type       Grp  TN
           1: principal   1   1          9:
           2:                           10:
           3:                           11:
           4:                           12:
           5:                           13:
           6:                           14:
           7:                           15:
           8:                           16:
```

TN assignments

Just as TN assignment determines the attendant group to which calls are terminated, the TN assignment also determines the VDN to which calls are redirected. If a VDN is administered, attendant group calls are redirected to the VDN rather than the attendant group. If a VDN is not assigned, calls terminate to the associated attendant group.

The selected TN for calls that are covered to an attendant group is the called user's TN, not the calling user's TN. When Tenant Partitioning is not administered, the system can have only one partition and attendant group. All attendant group calls are directed to attendant group 1. The screen to administer TN associations is not accessible, so system-wide console assignments apply. To follow the existing principals of this administration, the attendant vectoring VDN assignment appears on the Console Parameters screen when partitioning is turned off. When it is turned on, the field is removed from the console screen and the contents are automatically copied to TN 1.

Restrictions

No restrictions apply to attendant and non attendant vectoring. For example, an attendant VDN can point to a non attendant vector and vice versa. The same is true for vector commands.

For example, an attendant VDN that points to an attendant vector can have a vector step that routes to another non attendant VDN. In this case, the call is removed from the queue and treated as though it just entered vector processing rather than as a continuation from one VDN to another. The reverse is also true if a non attendant VDN is routed to an attendant VDN.

Attendant queue

If attendant vectoring results in putting a call in the attendant queue, it is placed in queue with the priority as administered on the console parameter screen. There are no changes made to the attendant priority queue for attendant vectoring. Even when partitioning is turned on and multiple attendant groups exist, all queues have the same priority assignments. Priority queue administration also applies for calls to an individual attendant, by way of the assigned extension.

Hunt group queue

If attendant vectoring results in putting a call in the hunt group queue, it is placed in the queue with the indicated priority. To use this command, the hunt group must be vector controlled.

Redirecting calls to attendant VDNs

Because it is not possible to apply vector commands or specialized administration to specific types of attendant group calls, the following can not be redirected to the attendant VDN.

Emergency Access: These calls are still sent directly to the attendant group. However, an attendant vectoring VDN can be assigned as the emergency access redirection extension.

Attendant return calls: These calls are still sent to the original attendant if the original attendant is available or will be placed into the attendant group queue if no attendants are available.

Serial calls: As with return calls, serial calls are still returned to the original attendant if the original attendant is available and are placed into the attendant queue if no attendants are available.

VIP Wakeup calls: These reminder calls are still sent directly to the attendant group.

Call Park time-out: These calls result in a conference (caller, principal, and attendant) and call vectoring does not allow conferenced calls to be vectored.

Call Transfer time-out: These calls are controlled by the attendant return call timer and are processed as though they are attendant extended calls, in other words, actual attendant return calls.

Night service

There is no additional night service functionality provided for attendant vectoring. Night service routing can be provided using the existing night station service in conjunction with attendant vectoring. All existing night service rules remain in place (for example, night console service supersedes night station service, which supersedes TAAS). Attendant group calls are not redirected to attendant vectoring when the system is in night service unless a night console is available. Otherwise, they continue to be redirected to the applicable night service processing. To achieve attendant vectoring for calls when the system is in night service without a night console, the night station service extensions must be attendant vectoring VDN extensions.

Attendant VDNs

The fact that VDN extensions can be dialed directly or calls can be transferred to VDN extensions is unchanged for attendant VDNs.

Currently, VDN extensions can be assigned to:

Hunt group night destination : An attendant vectoring VDN can be assigned as a hunt group's night destination. Calls to that hunt group when it is in night service are redirected to the VDN and attendant vectoring applies.Hunt group night service does not apply if the hunt group is vector controlled. When **vector?** on the Hunt Group screen is **y**, the **night service destination** field is removed from the screen. In order for a hunt group to be available in vectoring for the **`queue-to hunt-group`** command, the hunt group must be vector controlled. The hunt group in the **`route-to`** command could be in night service and the call would then terminate to the indicated night service destination. If the hunt group is accessed using the **`queue-to hunt-group`** command no night service applies.

LDN and trunk night destination: One or all trunk groups can be placed into night service and an attendant vectoring VDN can be assigned as the group's night service destination. If a night destination is assigned for LDN calls, it overrides (for LDN calls) the trunk group's night destination. Either of these destinations can be an attendant vectoring VDN. However, if Tenant Partitioning is administered and the trunk group night service destination is the attendant group, the call is redirected to the VDN that is associated with the trunk group's TN. If, instead, the night service destination is explicitly assigned to a particular attendant vectoring VDN, it may or may not be the VDN that would have resulted had the night destination been the attendant group.

Tenant night destination: For Tenant Partitioning, each partition can be assigned a night destination. When Tenant Partitioning is turned off, local attendant group calls are sent to the LDN night destination. When partitioning is turned on, local attendant seeking calls are sent to the partition's night destination.

Trunk group incoming destination: The incoming destination can be an attendant vectoring VDN except for RLT trunk groups. As in trunk group night service, an assigned incoming destination to an attendant vector could result in the call being sent to a different VDN than if the destination had been assigned to the attendant group.

Last coverage point in a coverage path: An attendant VDN can be assigned as a coverage point. If an Attendant VDN is assigned as a coverage point, it should be the last point in the coverage path.

Abbreviated dialing lists: Attendant VDNs can be assigned to abbreviated dialing lists.

Emergency access redirection: An attendant VDN can be assigned to emergency access redirection. When the attendant's emergency queue overflows or when the attendant group is in night service, all emergency calls are redirected to this VDN. Careful thought should be given to routing these calls off-switch.

QSIG CAS number for attendant group calls : An attendant VDN can be assigned to this number which determines where attendant group calls at a QSIG Branch are processed. This allows local vectoring at a Branch prior to routing the calls to the Main or elsewhere.

Auxiliary data for the following button assignments: In keeping with existing procedures, attendant VDNs will not be denied as auxiliary button data for:

- Facility busy indication. Visual indication of busy or idle status for the associated extension.
- Manual message waiting indication. Lights a message waiting lamp on the station that is associated with the button.
- Manual signaling. Rings the station that is associated with the button.
- Remote message waiting indicator. Message waiting status lamp automatically lights when a LWC message is stored in the system for the associated extension.

Attendant Vectoring and attendant VDNs

When Attendant Vectoring is administered and if an attendant VDN is assigned, attendant group calls are intercepted and sent through vector processing. The attendant VDN can be assigned on the Console Parameters screen if Tenant Partitioning is turned off or on the Tenant screen if partitioning is turned on. If an attendant VDN is assigned, the call is redirected to the VDN for vector processing. If a VDN is not assigned, the call is directed to the attendant group. Attendant group calls can only be redirected to attendant VDNs.

Intercept attendant group calls

When calls are placed to the attendant group or become attendant group calls for the reasons listed below, a check is made for an assigned attendant VDN. If an attendant VDN is assigned and either the system is not in night service or the system is in night service and a night console is available, the call is redirected to the VDN for subsequent vector processing. Otherwise, the call is treated with typical attendant group procedures.

The following occurrences can cause a call to become an attendant group call:

- Listed Directory Number (LDN)
- Attendant group in coverage path
- Attendant control of trunk group access
- Calls forwarded to attendant group
- Controlled Restriction
- Dialed attendant access code
- DID/Tie/ISDN intercept treatment
- DID time-out due to Unanswered DID Call Timer expiration
- DID busy treatment
- Security Violation Notification (SVN)
- Multi frequency signaling with attendant group as terminating destination
- CDR buffer full with attendant group as Call Record Handling Option
- Trunk incoming destination is attendant group
- Trunk group night service destination is attendant group
- Hunt group night service destination is attendant group
- Automatic Circuit Assurance (ACA) referral
- VDN routes to the attendant access code.

Vector override always applies to attendant VDNs. The **Allow VDN Override?** field will not be available so **yes** is assumed.

Allow override

VDN override always applies to attendant VDNs.

To provide the most flexibility possible, there are no restrictions placed on the vector that is assigned to a VDN. A non attendant vector can be assigned to an attendant VDN and an attendant vector can be assigned to a non attendant VDN. Obviously, doing so is not recommended. Assigning an attendant vector to a non attendant VDN severely restricts processing for basic call vectoring since only limited vectoring commands are available in attendant vectors. Assigning a non attendant vector to an attendant VDN also severely restricts attendant vectoring since the attendant-specific commands are not available in basic call vectoring. In addition, it removes basic call vectoring information from attendant VDNs. Also, there are no restrictions in vector chaining between attendant and non attendant vectors (for example, using the **goto vector** or **route-to number** commands).

Interflow between vectors

When calls interflow from one type of vector processing to another, they are removed from the queue (if applicable) and treated as new calls to vectoring, not continuations of vectoring.

Tenant Partitioning assignments apply to attendant VDNs the same as they do for non attendant VDNs. Therefore, care must be taken that a VDN assignment on the partitioning screen has a compatible TN number assigned to the VDN. For example, tenant partition 1 can be assigned a VDN which belongs to tenant partition 2 so long as partition 1's permissions allow access to partition 2. However, music source selection is based on the tenant partition where the VDN is assigned rather than the partition to which the VDN belongs.

Music source

When music is to be provided for attendant vectored calls, the source that is assigned to the tenant partition of the attendant seeking call is used rather than the source that is assigned to the partition of the VDN.

Attendant Vectoring and multiple queueing

Calls can exist in only one type of queue, which can be an attendant group, and individual attendant, or a hunt queue, and cannot be moved from one queue to another. For example, if a call is queued to the attendant group and a subsequent command attempts to queue the call to an individual attendant or hunt group, it is considered a failed queue attempt.

Restrict queueing to only one type of queue

Once a call is queued to the attendant group, individual attendant, or hunt group, any attempt to queue the call to another type of queue is considered a failed queue attempt.

Multiple attempts to queue to attendant groups or individual attendants are also considered failed queue attempts. For example, if a call is queued to attendant X and a subsequent command attempts to queue the call to attendant Y, the second queue command fails.

Allow multiple priority queueing within hunt queues

Since hunt group queueing is based on the indicated priority, multiple queue attempts are valid. There is no limitation on the number of attempts to queue to a particular hunt group so long as the command changes the priority at which a call is to be queued. For example, a call can be queued at low priority and subsequently requeued at medium and/or high priority. However, a second attempt to queue a call at the same priority for which it was previously queued is considered a failed queue attempt. Hunt group queueing is the functional equivalent to split queueing. As such, calls can be queued to a maximum of three different hunt groups at the same time.

Once a call is queued to a hunt group, any subsequent attempt to queue with a different priority results in the call being requeued with the new priority. Any subsequent attempt to queue with the same priority at which the call is already queued is considered a failed queue attempt.

Allow multiple hunt group queueing

A call can be queued to a maximum of three different hunt groups. Once this maximum is reached, any subsequent attempt to queue a call to a different hunt group is considered a failed queue attempt.

Considerations

The main consideration with Attendant Vectoring is training users to understand that calls placed to an attendant console may not always be answered by a live operator. If users are instructed to dial an attendant VDN, the call could be answered by an attendant, but it may also be covered to the voice mailbox of a night station. Training users to understand these different call routing options is something you should consider before using Attendant Vectoring.

If you use Attendant Vectoring and night service to route calls to a voice mail system, you can also use the Automatic Message Waiting feature to notify after-hours personnel that there are messages in the night service station mailbox by assigning an AMW lamp on one or more backup telephones. When personnel see that there are new messages, they can check those messages after hours and act upon them as needed.

Meet-me Conference

This section includes the following topics:

- About Meet-me Conference on page 441
- Command set on page 441
- Administering Meet-me Conference on page 445
- Meet-me Conference call processing scenario on page 451
- Troubleshooting on page 453

About Meet-me Conference

The Meet-me Conference feature allows you to set up a dial-in conference of up to six parties. The Meet-me Conference feature uses Call Vectoring to process the setup of the conference call.

Meet-me Conference can be optionally assigned to require an access code. If an access code is assigned, and if the vector is programmed to expect an access code, each user dialing in to the conference call must enter the correct access code to be added to the call.

The Meet-me Conference extension can be dialed by any internal or remote access users, and by external parties if the extension number is part of the customer's DID block.

Command set

The following table lists the commands associated with Meet-me Conference.

Meet-me Conference command set

Command category	Action taken	Command
Information collection		
	Collect information from the calling party.	`collect digits`

Meet-me Conference command set (continued)

Command category	Action taken	Command
Treatment		
	Play an announcement.	`announcement`
	Play a busy tone and stop vector processing.	`busy`
	Disconnect the call.	`disconnect`
	Delay with audible feedback of silence, ringback, system music, or alternate audio or music source.	`wait-time`
Routing		
	Route to the appropriate meet-me conference and stop vector processing.	`route-to`
Branching/Programming		
	Go to a vector step.	`goto step`
	Go to another vector.	`goto vector`
	Stop vector processing.	`stop`

Information collection commands

The following section details the syntax that can be used for this command and any information that is specific to the Meet-me Conference feature.

collect command

Syntax

```
collect 6 digits after announcement <extension>
```

When the **Meet-me Conf** field is enabled, the `collect` vector step has been modified to collect the next six digits and use those digits as the access code for a Meet-me Conference call. Though not required, the digits can be collected after a recorded announcement.

Treatment commands

Attendant Vectoring allows use of several treatment commands, including:

- announcement command on page 443
- busy command on page 443
- disconnect command on page 443
- wait-time command on page 443

The following sections detail the syntax that can be used for these commands and any information that is specific to the Meet-me Conference feature.

announcement command

Syntax

```
announcement <extension>
```

The usage for the **announcement** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

busy command

Syntax

```
busy
```

The usage for the **busy** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

disconnect command

Syntax

```
disconnect after announcement <extension>
```

The usage for the **disconnect** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

wait-time command

Syntax

```
wait-time <time> secs hearing <silence, ringback, music>
```

The usage for the **wait-time** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

Routing commands

The following section details the syntax that can be used for this command and any information that is specific to the Meet-me Conference feature.

route-to meetme command

Syntax

```
route-to meetme
```

The **route-to** vector step uses the condition **meetme** only for the Meet-me Conference feature. When successful, this condition adds the caller to the Meet-me Conference call and all parties on the call hear an entry tone to signify that another caller has joined the conference. This condition is valid when the caller has entered the correct access code and there are not already six parties on the call.

If the **route to meetme** step ever fails, vector processing stops and the caller hears busy tone.

Branching/programming commands

Meet-me Conference uses several branching/ programming commands, including:

- goto step command on page 444
- stop command on page 445

The following sections detail the syntax that can be used for these commands and any information that is specific to their use in Attendant Vectoring.

goto step command

Syntax 1

```
goto step <step #> if meet-me-idle
```

Syntax 2

```
goto step <step #> if meet-me-full
```

The **goto step** vector step has two conditions used for the Meet-me Conference feature:

- meet-me-idle
- meet-me-full

The **meet-me-idle** condition routes the first caller accessing a Meet-me Conference to the conference call. An announcement step saying they are the first party to access the call can be given to the caller.

The **meet-me-full** condition is used when the Meet-me Conference already has the maximum of six parties on the call.

Syntax 3

```
goto step <step #> if digits = meet-me-access
```

The **goto step** vector step supports the option, **meet-me access**, for the **digits** condition to verify that the access code is valid. If the access code entered by the caller equals the access code administered for the VDN, vector processing continues.

stop command

The use of the **stop** command is the same as in Basic Call Vectoring. For details on using this command, see the Basic Call Vectoring section.

Administering Meet-me Conference

This section includes the following topics:

- Activating the Meet-me Conference feature on page 445
- Creating a Meet-me Conference VDN on page 446
- Creating a Meet-me Conference vector on page 447
- Interactions on page 448
- Security issues on page 450
- Capacity issues on page 450

Activating the Meet-me Conference feature

Meet-me Conference is available for all switch models that support the R11 call processing software.

To enable the Meet-me Conference feature:

- The G3 Version field of the Customer Options screen must be set to V11 or later.
- The Enhanced Conferencing field of the Customer Options screen must be enabled. This feature has an RTU cost and must be enabled through the License File process.

Creating a Meet-me Conference VDN

To create a Meet-me Conference VDN (using example VDN 36090):

1. Enter:

   add vdn 36090

   The system displays the VDN screen:

```
add vdn 36090                                          Page   1 of   3   SPE A
                            VECTOR DIRECTORY NUMBER

                             Extension: 36090
                                  Name: Enhanced Conf. Meet-me VDN
                         Vector Number: 90

                    Meet-me Conferencing? y
```

2. Enter a name, a vector number, and enter **y** in the **Meet-me Conferencing** field.
3. Press **NEXTPAGE** to display page 2.

   The system displays page 2 of the VDN screen:

```
add vdn 36090                                        Page   2 of   3   SPE A
                            VECTOR DIRECTORY NUMBER

                        MEET-ME CONFERENCE PARAMETERS
       Conference Access Code: 937821
        Conference Controller: 80378
              Conference Type: 6-party
              Route-to Number:
```

4. Enter a conference access code. If you do not want an access code, leave the field blank. Once an access code is assigned, an asterisk displays in this field for subsequent change, display, or remove operations by all users except the *init* super user login.

   SECURITY ALERT:
   You should always assign an access code to a Meet-me Conference VDN.

5. Enter a conference controller extension. If an extension number is entered, a user at that extension can change the access code for the Meet-me Conference VDN using a feature access code. If this field is blank, only a station user that is assigned with console permissions can change the access code for the Meet-me Conference VDN using a feature access code. In addition, remote access users can change a Meet-me Conference access code using the feature access code.

6. Enter the conference type. This field can have the following values:
   - 6-party - Enter this value to administer a regular 6-party conference. This value is the default.
   - expanded - Enter this value if you want to administer up to a 300-party conference.
7. If you set the **Conference Type** field to **expanded**, use the **Route-to Number** field to administer the ARS/AAR Feature Access Code, the routing digits, and the conference ID digits for the VDN.
8. Press **ENTER** to submit the VDN.

Creating a Meet-me Conference vector

To create a Meet-me Conference vector (using example vector number 90):

1. Enter:

   change vector 90

   The system displays the CALL VECTOR screen.
2. Enter **y** in the **Meet-me Conf** field. This designates the vector as a Meet-me Conference vector.

3. Create a vector as shown in the following example:

```
change vector 90                                        Page   1 of   3   SPE A
                                CALL VECTOR

    Number: 90               Name: Meet-me Vec
Multimedia? n      Attendant Vectoring? n    Meet-me Conf? y           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? n

01 collect      6    digits after announcement 12340
02 goto         step   6   if digits           =      meet-me-access
03 collect      6    digits after announcement 12341
04 goto         step   6   if digits           =      meet-me-access
05 disconnect   after announcement 12342
06 goto         step   11  if meet-me-idle
07 goto         step   14  if meet-me-full
08 announcement 12343
09 route-to     meetme
10 stop
11 announcement 12344
```

```
change vector 90                                        Page   2 of   3   SPE A
                                CALL VECTOR

12 route-to     meetme
13 stop
14 disconnect   after announcement 12345
15 stop
16
17
18
19
20
21
22
```

4. Press **ENTER** to submit the vector.

Interactions

The following are administration interactions for Meet-me Conference.

General

Both Attendant Vectoring and Meet-me Conference cannot be enabled at the same time.

If Enhanced Conferencing is enabled, but no other vectoring customer options are enabled, only Meet-me Conference vectors can be assigned.

A non Meet-me Conference vector cannot be assigned to a Meet-me Conference VDN and a Meet-me Conference vector cannot be assigned to a non Meet-me Conference VDN.

There will be no restrictions in vector chaining between Meet-me Conference and non Meet-me Conference vectors (for example, using the **goto vector** or **route-to number** commands). When calls interflow from one type of vector processing to another, they will be removed from any queue (if applicable) and treated as new calls to vectoring, not a continuation of vectoring.

Call Detail Recording

As parties join a Meet-me Conference, a call record is created if required by system administration. If a record is required, the called party will be the Meet-me Conference VDN number and the duration will be the length of time that the party was included in the call. There will be an individual record for each party that will be output when the party drops from the call. One option that will record all calls to Meet-me Conference VDNs is to activate the Intra-switch CDR feature and populate all the Meet-me Conference VDN numbers in the system.

If the Intra-switch CDR feature is used with the Meet-me Conference VDNs, the condition code should be set to C for all call records as is done with traditional conference calls when Intra-switch CDR is active.

If Intra-switch CDR feature is not active for Meet-me Conference VDNs, the creation and contents of call records will depend on the trunk group translations for external callers to the Meet-me Conference. Internal callers to the Meet-me Conference will not generate any records if the Intra-switch CDR feature is not active for either the Meet-me Conference VDN or the calling extension.

Changing vector types

To change a Meet-me Conference vector to a non Meet-me Conference vector, the administrator must first remove all vector steps. To change a non Meet-me Conference vector to a Meet-me Conference vector, the administrator must first remove all vector steps. If either of these conditions exist, a warning message displays that states *VDNs currently assigned to this vector may not operate as expected.* The next time the administrator tries to submit a change to the Meet-me Conference VDN, they would be forced to assign the VDN to a Meet-me Conference vector.

Direct Inward Dialing (DID)

If the VDN extension is part of the customer's DID block, external users will be able to access the conference VDN. If the VDN extension is not part of the customer's DID block, only internal callers on the customer's network (including DCS or QSIG) or remote access callers can access the conference VDN.

Disabling Enhanced Conferencing

If Meet-me Conference VDNs are assigned when disabling the Enhanced Conferencing option, the change is not allowed and the message, *Must first remove all Meet-me Conf VDNs and vectors*, is displayed. The administrator must remove those VDNs and vectors before the option can be disabled.

Removing stations

A station that is administered as a controlling station for a Meet-me Conference VDN cannot be removed without first removing the assignment on the VDN. The following message displays:

```
Must first remove as conference controller on VDN form.
```

Security issues

The Meet-me Conference feature is a potential security problem. If Meet-me Conference VDNs are assigned without access codes, hackers could tie up Meet-me Conference facilities, keeping others from conducting legitimate business, and could potentially access the switch and use the switch to make unauthorized calls. Therefore, we should recommend that all Meet-me Conference VDNs have access codes that are known only to administrators and users on a need to know basis. We should also recommend that access codes be changed on a regular basis to reduce the risk of unauthorized access to the switch.

If a user tries to change the access code of a Meet-me Conference and is unsuccessful, or if a user tries to access a Meet-me Conference and uses an invalid access code, a meet-me event is logged. For more information, see Tracking unexpected events on page 707.

Capacity issues

Meet-me Conference calls count towards the maximum number of 3-way and 6-way conference calls.

Users cannot add more parties to a conference call once the system maximum is reached.

For Category A, the number of Meet-me Conference VDNs is a subset of the total number of VDNs allowed in the system.

For Category B, the total number of VDNs and vectors is doubled from the normal limit if both Call Vectoring and Enhanced Conferencing are enabled. However, the maximum number of VDNs and vectors available for call center applications is unchanged.

Meet-me Conference call processing scenario

Joe Davis has a sales review scheduled with four associates located in different cities. He has reserved Meet-me Conference telephone number 865-253-6090. In switch administration, this number has been assigned to vector 90. See the following screen.

```
add vdn 36090                                          Page   1 of  3   SPE A
                            VECTOR DIRECTORY NUMBER

                             Extension: 36090
                                  Name: Meet-me VDN
                          Vector Number: 90

                       Meet-me Conference? y
```

VDN 36090 is administered with an access code of 835944.

When each associate calls the Meet-me Conference telephone number, the following vector processing occurs:

```
change vector 90                                        Page   1 of   3   SPE A
                                CALL VECTOR

    Number: 90               Name: Meet-me Vec
                  Attendant Vectoring? n    Meet-me Conf? y           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? n

01 collect      6    digits after announcement 12340
02 goto         step   6   if digits           =      meet-me-access
03 collect      6    digits after announcement 12341
04 goto         step   6   if digits           =      meet-me-access
05 disconnect   after announcement 12342
06 goto         step   11  if meet-me-idle
07 goto         step   14  if meet-me-full
08 announcement 12343
09 route-to     meetme
10 stop
11 announcement 12344
```

```
change vector 90                                        Page   2 of   3   SPE A
                                CALL VECTOR

12 route-to     meetme
13 stop
14 disconnect   after announcement 12345
15 stop
16
17
18
19
20
21
22
```

Each caller hears announcement 12340, which says something similar to *Welcome to the Meet-me Conferencing service. Enter your conference access code.* Each caller enters the access code 835944.

The **collect** vector step 1 collects the access code digits. If the access code is valid, the vector processing continues with vector step 6. If the access code is invalid, the vector processing continues with vector step 3, which plays announcement 12341. Announcement 12341 says something similar to *This access code is invalid. Please enter the access code again.* If the caller enters the wrong access code again, the vector processing continues with vector step 5, which plays announcement 12342. Announcement 12342 says something similar to *This access code is invalid. Please contact the conference call coordinator to make sure you have the correct conference telephone number and access code. Good-bye.*

Vector step 6 is only valid for the first caller into the Meet-me Conference. The meet-me-idle condition routes the first caller to announcement 12344 (vector step 11). The recorded announcement says something similar to, *You are the first party to join the call.* The caller is then routed to the Meet-me Conference call by vector step 12 and vector processing stops.

Vector step 7 is used when the Meet-me Conference already has the maximum of six parties on the call. The meet-me-full condition disconnects the caller after playing announcement 12345 (vector step 14). The recorded announcement says something similar to, *This Meet-me Conference is filled to capacity. Please contact the conference call coordinator for assistance. Good-bye.*

If a caller enters the correct access code, is not the first caller, and the conference call is not full, vector processing continues with vector step 8, which plays announcement 12343. The announcement says something similar to *Your conference call is already in progress.* The caller is then routed to the Meet-me Conference call by vector step 9 and vector processing stops. As each caller enters the conference call, all parties on the call will hear an entry tone.

When the conference call is over and callers drop out of the conference call, any remaining parties on the call will hear an exit tone.

Troubleshooting

This section describes common problems and possible resolutions for the Meet-Me Conference feature. Topics described in this section include:

- Conference call drops on page 453
- Sound volume is too low on page 454

Conference call drops

The conference call drops abruptly for no apparent reason.

Possible reason: The Vector Disconnect Timer on the System-Parameters Features screen is set to a value that does is shorter than the duration of the Meet-Me Conference session.

Solution: Increase the Vector Disconnect Timer value.

Sound volume is too low

Voice volume levels for some conference participants is too low.

Possible reason: The affected conference participants connect through international trunks in which Central Office (CO) loss plans are set for too much loss.

Solution: In the System-Parameters Country Options screen, go to Tone & Country Loss Plans (page 3) and change the values specified in the **End-to-End total loss (dB) in a n-party conference** field.

Expert Agent Selection

This section describes Expert Agent Selection (EAS), discusses EAS upgrades, and provides examples that show how EAS is implemented. This section includes the following topics:

- What is EAS? on page 455
- EAS benefits on page 456
- EAS considerations on page 459
- Expert Agent Selection (EAS) terminology on page 460
- EAS-PHD - 60 skills/16 skill levels on page 461
- Switch administration for the EAS feature on page 462
- Identifying caller needs on page 467
- Functions and examples on page 473
- EAS feature interactions on page 493
- EAS adjunct interactions on page 498
- Upgrading to the EAS environment on page 502

What is EAS?

Expert Agent Selection (EAS) helps call center managers provide the best possible telephone service to callers by matching the needs of the callers with the skills or talents of the agents. Caller needs and agent skills are matched using Call Vectoring. All the Call Vectoring features described in this guide can be used with EAS.

Matching the call to an agent with the appropriate skills reduces transfers and call-holding time. Accordingly, customer satisfaction is increased. Also, since an entire agent group need not be trained at the same time for the same skills, employee satisfaction is increased.

In addition to matching the skills that are required for a call to an agent with one of those skills, EAS provides other capabilities:

- Logical Agent associates hardware (the telephone) with an agent only when the agent is logged in. While the agent is logged in, calls to the agent login ID are directed to the agent. For more details, see Logical Agent capability on page 481.
- Direct Agent Calling (DAC) allows a user to call a particular agent and have the call treated as an ACD call. For more details, see Direct Agent Calling on page 470.

Most EAS administration can be completed before you activate it, thus minimizing the down time for upgrading to EAS.

EAS requires ACD and Call Vectoring. All of the existing ACD features and Call Vectoring capabilities can be used within EAS applications.

As with Call Vectoring calls, EAS calls are directed to VDNs, which in turn point to vectors. However, unlike Basic Call Vectoring, skills can be assigned in EAS to VDNs, or they can be associated with vector steps to represent caller needs. As for Call Vectoring calls, EAS calls are queued to ACD hunt groups. However, with EAS enabled, ACD hunt groups are called *skill hunt groups* instead of splits.

Skill hunt groups deliver calls to EAS agents. Agent skills are administered on the Agent Login ID screen.

Note:
These are the same login IDs that are used by Avaya Call Management System (CMS) and Basic Call Management System (BCMS).

Logical Agent implies that telephones are no longer preassigned to hunt groups. When the agent logs, the telephone becomes associated with all of the skill hunt groups that are assigned to that agent login ID.

With EAS optioned and enabled, ACD calls can also be directed to a particular agent, instead of to the skill hunt group, by using the DAC feature. The direct agent call is treated like an ACD call, but it waits in queue for a specific agent to become available. direct agent calls have a higher priority than skill hunt group calls.

EAS benefits

This section includes the following topics:

- About EAS benefits on page 457
- Skill-based call distribution on page 457
- Greatest need call distribution on page 458
- Percent allocation call distribution on page 458
- ACD queuing and vector commands on page 458

About EAS benefits

Because you can match caller needs to an agent who has the appropriate skills to handle the call, your call center can achieve the following:

- Maximum profitability.
- Greater customer satisfaction because the caller reaches, on the first call, an agent with the necessary skills to handle the call.
- Greater responsiveness to customer needs because you can base call distribution on either skill level or greatest need.
- Improved agent performance and satisfaction because agents handle calls they are most familiar and most comfortable with.
- Improved agent performance because supervisors have the option to have agents handle calls based on either skill level or greatest need. For agents, it offers an opportunity to learn new skills.
- Ability to track the number of calls that are handled by particular skills from the VDN perspective. You can see whether vectors are performing as expected.

Skill-based call distribution

With EAS, call distribution is based on agent skills. Caller needs are determined by the VDN called or by voice prompting.

An agent who has at least one of the skills that a caller requires is selected to handle the call. You assign skills and skill levels to agents to determine which types of calls go to which agents and to determine the order in which agents serve waiting calls.

Interruptible Aux

If a skill's designated service level is not met, unavailable EAS agents who are in Auxiliary (AUX) work mode and have an interruptible reason code can be made available. Using this feature, for example, during the call volume spikes, you can use agents in Auxiliary (AUX) work mode to maintain the service level.

For more information on Interruptible Aux, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Greatest need call distribution

With EAS, you have the option of basing call distribution on greatest need instead of skill level. You can distribute the highest-priority, oldest call waiting to an agent with an appropriate skill, even if that skill is not the agent's highest-priority skill.

Percent allocation call distribution

Percent allocation enables you to assign a percentage of an agent's time to each of the agent's assigned skills, to comprise a total of 100% of the agent's staffed tome. Percent allocation then selects the call that is the best match for an agent's administered skill percentages.

Percent allocation is available with Avaya Business Advocate. For more information, see *Avaya Business Advocate User Guide*.

Percentage allocation routing

Based on specified percent allocation, percentage allocation routing helps you to distribute calls among a set of call centers or local/remote destinations using VDNs. Percentage allocation routing enables desired distribution of work by routing the incoming calls according to the percentage specified for each of the destinations.

For more information on Percentage allocation routing, see Percentage allocation routing on page 373.

ACD queuing and vector commands

ACD queuing and the vector commands **`queue to skill`** and **`check skill`** are used to route a call to an agent with the appropriate skill to handle the call.

Route calls by skill level

Under agent surplus conditions, this feature allows you to request selection of an agent with a particular skill level. Using vector processing, you can apply skill level as well as skill (and other factors) in agent selection. The **`check skill`** vector command is used to route calls to an agent with a particular skill. For more information on check skill vector command, see Check skill for available agents with level preference on page 552 and *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* . For more information on how to route calls by skill level, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

EAS considerations

When you implement the EAS feature, be aware of the following considerations:

- With EAS, skill hunt groups replace splits. You cannot administer both skills and splits on the same switch. All ACD hunt groups must be administered as either splits or skills. If EAS is optioned, all ACD hunt groups are skill hunt groups.
- With EAS, all skill hunt groups except for messaging-system hunt groups must be vector controlled.
- With EAS, non-ACD hunt groups are allowed, but they cannot be vector controlled.
- Agent login IDs are extensions in the dial plan, and they decrease the total number of stations that can be administered.
- With EAS, agents have a different login procedure and a single set of work mode buttons, regardless of the number of skills that are assigned to the agents.
- Skill hunt groups can distribute a call to the most-idle agent (UCD) or to the most-idle agent with the highest skill level for that skill (EAD). In either of these cases, the call can route to the most-idle agent for the specified skill, or to the most-idle agent in all of the skills. Direct Department Call (DDC) distribution is not allowed for skill hunt groups.
- With either UCD or EAD distribution, the system can be administered to deliver calls based either on greatest need or agent skill level. This is the Call Handling Preference that is administered on the Agent LoginID screen. When calls are in the queue, greatest need delivers the highest priority oldest call waiting for any of the agent's skills. With skill level administration, the system delivers the highest priority oldest call waiting for the agent's highest level skill with calls in the queue.
- The EAS-PHD customer option adds additional capabilities to the basic EAS capabilities.
  - It increases the number of skills an agent can log in to from 4 to 20
  - It increases the number of agent skill priority levels from 2 to 16

For information on converting a call center to EAS, see Converting a call center to EAS on page 843.

Expert Agent Selection (EAS) terminology

The following terms have special significance in the EAS environment.

Agent skill	The type of call that a particular agent can handle. With EAS, an agent can be assigned up to four skills each, with a primary (level 1) or secondary (level 2) skill level. With the following releases of Communication Manager for EAS-PHD: • Prior to 2.0, an agent can be assigned as many as 20 skills • Later than 2.0, an agent can be assigned up to 60 skills
Caller needs	The reasons why customers call your call center. Caller needs are determined by the VDN number that the caller dialed, by Call Prompting, or by Automatic Number Identification (ANI) database lookup. You define caller requirements in the vector in order to route calls to an ACD agent with particular skills to match the needs of the caller. These caller needs, which translate to skills, become active for an ACD call whenever a queue to the main skill or check backup skill vector command is executed and the threshold condition is met.
Skill	A specific caller or business need of your call center. You define your skills based on the needs of your customers and your call center. You specify skills by skill numbers, which are assigned to agents and are referenced in vectors to match caller needs with an agent who is skilled to handle those needs. When configuring your call center for skills, a particular skill number always has the same meaning, whether it is an agent skill, VDN skill, or skill hunt group.
Skill hunt group	Calls are routed to specific skill hunt groups that are usually based on caller needs. Agents are not assigned to a skill group; instead, they are assigned specific skills that become active when they log in.

Skill level	For each agent skill, a skill level may be assigned. With EAS-PHD, skill levels can range from 1 to 16, with 1 being the highest skill level (also known as the highest-priority skill). Without EAS-PHD, skill levels may be defined as primary (level 1) or secondary (level 2), with the primary being the highest-priority skill. When calls are queued for more than one of the agent's skills and the agent's call-handling preference is by skill level, the agent receives the oldest call waiting for the agent's highest level skill. If an agent's call-handling preference is by greatest need, then the agent receives the highest-priority, oldest call waiting for any of that agent's skills, regardless of skill level.
Top agent	An agent in a given skill who has the skill assigned as top skill.
Top skill	For EAS-PHD, an agent's first-administered, highest-priority skill. For EAS, an agent's first-administered primary skill (or first-administered secondary skill if the agent has no primary skill assigned). With call-handling preference by skill level, this is the skill for which the agent is most likely to receive a call.
VDN skill preference	Up to three skills can be assigned to a VDN. Calls use VDN skills for routing based on the preferences that you specify in the vector. VDN skill preferences are referred to in the vector as 1st, 2nd, and 3rd.

EAS-PHD - 60 skills/16 skill levels

EAS-PHD is a feature that allows an agent to be assigned to as many as 60 skills. For each skill, one of the 16 skill levels can be assigned, with 1 being the highest skill level and 16 being the lowest skill level.

If calls are waiting for some of the agent's skills and the agent's call-handling preference is by skill level, the agent receives the call that requires the agent's highest-priority skill. For an agent, the first-administered, highest-priority skill is known as the agent's top skill. The top skill represents the skill for which the agent is most likely to receive a call.

If an agent's call-handling preference is by greatest need, the top skill is not useful, because the agent receives the highest-priority, oldest call waiting that requires any of the agent's skills, regardless of skill level.

Switch administration for the EAS feature

Before activating EAS in your call center, you need to complete the appropriate screens on your Avaya communication server as described in the following sections.

This section includes the following topics:

- EAS administration screens on page 462
- Other screens that support EAS Agent LoginID on page 463

EAS administration screens

The following table lists the screens used to administer EAS. For more information about the screens listed below, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

EAS administration screens

Screen	Use
System-Parameters Customer-Options	The **Expert Agent Selection Enabled?** field on this screen changes to **y** when EAS is installed. If you purchased EAS-PHD, the **Expert Agent Selection-Preference Handling Distribution (EAS-PHD) Enabled?** field changes to **y**.
Dial plan	Use this screen to change the dial plan. It is recommended that login IDs start with a unique digit in the dial plan (for example, 5111, 5123, 5432). It is preferable to dedicate a block of numbers for login IDs. If your login IDs do not have the same first digit and the login IDs are four digits long, consider changing to a 5-digit number for login IDs. This may require a modification to the CMS login ID if the current ID is not a valid extension number or cannot be made available in the switch dial plan. Agent login IDs must be different from assigned telephone extensions.
VDN	Use this screen to add or change VDNs and to designate skill preferences.
Vector	Use this screen to change vectors.
Hunt Group	Use this screen to add or change skill hunt groups. The **Skill?**, **ACD?** and **Vector?** fields must be all **y** or all **n**. Hunt group types should be either UCD or EAD. You cannot administer agents on this screen when EAS is enabled.

EAS administration screens ***(continued)***

Screen	Use
Agent Login ID	Use this screen to add or change agent login IDs and skill assignments. If you add or change skills on the switch, the agent must log out and then log in again before the changes take effect. You must use the Agent Login ID screen to select call-handling preferences for agent login IDs. The **Call Handling Preference** field must be set to either skill level or greatest need. The default is skill level. You also may enter a direct agent skill number in the **Direct Agent Skill** field. The skill entered in this field must be one of the agent's administered skills or the field is left blank. If no direct agent skill is administered and the agent receives a direct agent call, the call is delivered to the agent's first-administered, highest-level skill.
Station	Only a single set of work mode buttons is needed with EAS. Use this screen to remove additional sets of buttons if you are administering agents in multiple splits.

Other screens that support EAS Agent LoginID

The following table lists switch administration screens that can have an EAS Agent loginID administered on them.

EAS loginID table

Feature	Accepts loginID?
Abbreviated Dialing Buttons	
7103A	Yes
Enhanced	Yes
Group	Yes
Personal	Yes
System	Yes
Agent-LoginID	
Port Extension	No
Announcements	No
Buttons	

EAS loginID table (continued)

Feature	Accepts loginID?
abrdg_app	No
aut-msg-wt	Yes
brdg_app	No
busy-ind	Yes
data_ext	No
man_msg_wt	No
q-calls	No
q-time	No
signal	No
Call Processing	
Auto-Callback	No
Call Forward from Agent Login ID	No
Call Forward to Agent Login ID	Yes
Call Park	Yes
Hundreds group	No
LWC Retriever gets lagt msgs	Yes
Service observ Agent Login ID	Yes
CDR Parameters	
Primary Extension	No
Secondary Extension	No
Code-Calling	Yes
Communication Link screen	
Communication Link Digits	No
Console Parameters	
CAS-backup ext	No
IAS Att Access Code	No

EAS loginID table (continued)

Feature	Accepts loginID?
Coverage Groups	
Answer Group Member	No
Path	Yes
Measured Principals	
Coverage Measurement	No
Feature-Related Parameters	
ACA-referral dest.	No
ACA - long holding	No
ACA - short holding	No
Controlled out restriction	No
Controlled Terminal	No
Controlled Stn-to-Stn	No
DAA Extension	No
DID/Tie/ISDN announcement	No
Emergency Access Redirection	No
CDR output extension	No
SVN referral destination (announcement)	Yes
System LWC retriever	No
System Printer	No
Hospitality Parameters	
Journal Printer	No
LWC wakeup	No
PMS ext	No
PMS log	No
Routing on Voice Synthesis	No

EAS loginID table (continued)

Feature	Accepts loginID?
Hunt Group screen	
Announcement extension	No
ASAI link	No
AUDIX extension	No
Calls Warning extension	No
Member	No
Night Service	No
Supervisor	Yes
Time Warning extension	No
Intercom Group Member	No
Intra-switch CDR	Yes
Listed Directory Number	
Member	No
Night Destination	Yes
Malicious Call Trace	
MCT Member	No
Permanent Switched Calls	No
Personal CO Line	No
Pickup Group Member	No
Remote Access Extension	No
Term Extension Group Member	No
Trunk Group	
Night Service	Yes
Incoming Destination	Yes
Member Night Service	Yes

EAS loginID table (continued)

Feature	Accepts loginID?
Vector Administration	
adjunct extension	No
announcement	No
messaging	Yes
route-to	Yes

Identifying caller needs

This section includes the following topics:

- About identifying caller needs on page 467
- DNIS/ISDN called party on page 469
- Call Prompting/VRU Digits/CINFO digits on page 469
- Host database lookup on page 470
- Direct Agent Calling on page 470

About identifying caller needs

Caller needs for a particular call can be identified by any of the following methods:

- Interpreting information that is passed from the network in the screen of DNIS digits or ISDN messages.
- Processing Call Prompting digits, digits entered at a Voice Response Unit (VRU), or CINFO digits that are forwarded by the network.
- Using Adjunct Switch Application Interface (ASAI) or a VRU such as Avaya Interactive Response in a host database lookup.

To show how a call center manager might match caller needs and agent skills (which can be viewed as capabilities needed from the caller's perspective), assume that a call center receives inbound calls from automobile club members who speak Spanish or English. The callers in this case either need to plan a vacation route or have trouble with their car and are calling for assistance. The following table provides example associations between caller needs and agent capabilities.

Example of caller need-to-agent skill matching

Caller need	Capability needed
Tourist information	Knowledge of the region
To speak Spanish	Bilingual
Emergency assistance	Handle stressful callers
Tow truck	Access to dispatch systems

The following list looks at the call center manager's strategy in matching the caller needs to the capabilities of the agent:

- Tourist information/knowledge of the region

  Travelers may need information while traveling or regarding a future trip. All assigned agents can provide this information.

- To speak Spanish/bilingual

  Separate numbers are published and used as part of Spanish membership information, or Call Prompting is used after a general number is dialed.

- Emergency assistance/handle stressful callers

  Separate emergency road service numbers are published and used, or Call Prompting is used after a general number is dialed. For example, a number is provided for towing.

Note that the call center chose to implement Call Prompting to identify Spanish-speaking callers and callers who require emergency assistance. This allows for quicker and more specialized treatment and therefore better satisfies the caller's needs.

In addition, some customers might prefer to speak to the agent that he or she spoke to on a previous call. To accommodate this request, a call center manager can implement Direct Inward Dialing (DID) at the call center. Also, Direct Agent Calling (DAC) can be used to direct a call to a specific agent.

The following sections explain further how caller needs are identified.

DNIS/ISDN called party

A set of DNIS digits can be interpreted as a VDN. The following table presents four services and their corresponding telephone number including DNIS digits that might be provided to the caller.

Examples of services and corresponding DNIS digits

Service	Telephone number	Corresponding DNIS
Emergency road service (English)	800-765-1111	6001
Emergency road service (Spanish)	800-765-2222	6002
Route planning (English)	800-765-3333	6003
Route planning (Spanish)	800-765-4444	6004
General (Call Prompting)	800-765-5555	6005

Note:
DNIS digits must be extensions that are reflected in the dial plan.

Call Prompting/VRU Digits/CINFO digits

The Call Prompting/VRU/CINFO digits are entered by the caller in response to any recorded question about a caller's needs, or in the case of CINFO ced or cdpd digits, are provided by the call center host computer. For example, a hotline for a product may request that a product code be entered, or a travel service may request a 2-digit state code to indicate the state to which the caller would like to travel. The following table provides a prompt that encourages the caller to enter the appropriate Call Prompting digit for the needed service from the automobile club.

Example of a prompt for entering Call Prompting digits

For emergency road service, dial 1.
Para asistencia con su automovil, marque el dos.
For travel route directions, dial 3.
Para informacion sobre rutas, marque el cuatro.

Host database lookup

A host database lookup uses DNIS and ANI (calling party's number) to determine what skills are required or even the agent desired. For example, the database may show that the caller speaks Spanish and has been working with Agent 1367. To access host information, either Adjunct Switch Application Interface (ASAI) or a VRU in conjunction with a **converse-on skill** step is used.

Direct Agent Calling

This section includes the following topics:

- About DAC on page 470
- Advantages of DAC on page 471
- How DAC works on page 471
- Administering DAC on page 472
- Administering DAA on page 473

About DAC

Direct Agent Calling (DAC) is an EAS feature that lets a caller:

- Contact a specific agent instead of a skill hunt group
- Queue for the agent if the agent is on a call
- Use Agent LoginID for callbacks and transfers
- Hear system wide direct agent delay announcement while holding
- Follow the agent's coverage path, if the call is not answered immediately

DAC allows a call to a specific ACD agent to be treated as an ACD call. Zip-tone answer, ACW, and other ACD features can be used with direct agent calls.

If an agent is logged in but is not available, the call queues for that agent. If the agent is not logged in, the call follows the agent's coverage path.

EAS Direct Agent Calling is accomplished by dialing the login with the proper class of restriction (COR) settings. Both the caller (that is, trunk, VND, or station) and the agent must have the direct agent COR settings.

Customers might call an agent directly using Direct Inward Dialing (DID) if the agent's login ID is a published number, or customers might dial a toll-free number and be prompted for the agent's login ID extension. Vectors can be designed to handle the Call Prompting function.

Note:
DAC requires CallVisor Adjunct-Switch Application Interface (ASAI) or EAS. Both originating and called party Class of Restrictions (CORs) must be set to allow Direct Agent Dialing.

Advantages of DAC

Direct agent calls have two important advantages:

- They reduce the need to transfer callers who want or need to speak with a certain agent, such as the agent spoken to on a previous call.
- They provide more accurate reporting of calls, because CMS counts direct agent calls as ACD calls. In this way, agents get proper credit for taking them. By comparison, calls transferred to an agent are not counted as ACD calls.

How DAC works

DAC works as described below:

- Callers can dial the agent's login ID as part of a DID or from auto attendant as an extension number.
- Direct agent calls have a special ringing sound, regardless of the agent's work state, and the current work mode button on the agent's telephone flashes.
- If the agent is on a call, he or she can use multiple call handling to decide whether to put the call on hold in order to take the direct agent call.
- If the agent is available, the call is delivered according to the answering and ringing options.
- If the agent is not available, or if multiple call handling is not used, call coverage or RONA routes the call to backup.
- While on direct agent calls, agents are unavailable for subsequent ACD calls. If the agent logs off by unplugging the headset, he or she can still answer a direct agent call in the queue by logging back in and becoming available. Agents who have direct agent calls waiting are allowed to log off. Direct Agent calls left in-queue for any logged-out agent will remain in-queue until coverage takes over (unless the Direct Agent call was made with the Priority Calling feature), or until the calling party abandons. If the agent is in Manual In mode or pushes the After Call Work (ACW) button while on a direct agent call, the agent goes to ACW mode.

Generally, direct agent calls are queued and served in first-in, first-out order before other calls, including priority calls. However, if you administer a skill level for Call Handling Preference, direct agent calls must be assigned the highest priority for them to be delivered before other ACD calls. Otherwise, calls with a higher skill level are distributed before direct agent calls.

Note that you can use Multiple Call Handling (MCH) to allow agents to answer a direct agent call with another ACD call active.

Direct agent calls follow the receiving agent's coverage and call forwarding paths, if these features are administered. Once a call goes to coverage or is forwarded, the call is no longer treated as a direct agent call, and CMS is informed that the call has been forwarded.

Administering DAC

To administer DAC:

1. On the Agent LoginID screen, enter the agent's direct agent skill.
2. Use the Hunt Group screen to set up a skill for all DA calls.

   This skill will:

   - Tell the switch how to handle calls to the skill.
   - Show report users how much time each agent has spent on DA calls.

   Note:
   Any agent who will receive direct agent calls should have at least one non-reserve skill assigned to the agent loginID.
3. Add the skill to the agent's administered skills on the Hunt Group screen.

   Whenever an outside caller dials the agent's extension, the switch looks at the entry in that field to determine the skill for tracking call data.
4. On page 8 of the Feature-Related System Parameters screen, you may specify:

   - A Direct Agent Announcement Extension that plays an announcement to direct agent callers waiting in queue.
   - Amount of delay, in seconds, before the announcement.
5. Administer a Class of Restriction (COR) for DA calls.
6. Use the Trunk Group screen to administer Direct Inward Dialing (DID).
7. On the second page of the Hunt Group screen, you can administer Multiple Call Handling On-Request for this hunt group.

   This feature will enable agents to see that the incoming call is a DA call and put the current call on hold to answer the DA call.
8. If there is no answer after a certain number of rings, use RONA to redirect the caller to a VDN that points to a vector. You can set up the vector to provide appropriate routing and treatment for the call.
9. On page 3 of the Hunt Group screen, administer messaging for the DA hunt group.
10. Assign this hunt group to agents who need to receive DA calls.

Administering DAA

Direct Agent Announcement (DAA) enhances Direct Agent Calling (DAC) capabilities for CallVisor Adjunct-Switch Application Interface (ASAI) and Expert Agent Selection (EAS). It plays an announcement to DAC waiting in a queue. The following screens should be administered for DAA.

You must also have enabled either Expert Agent Selection (EAS) or ASAI Adjunct Routing (or both).

Screen	Field
System-Parameters Customer-Options	• ACD • Vectoring (Basic) • Expert Agent Selection (EAS) • or • ASAI Adjunct Routing
Feature-Related System Parameters	• Direct Agent Announcement Delay • Direct Agent Announcement Extension
Announcements/Audio Sources	All

Functions and examples

This section includes the following topics:

- Administering skills on page 473
- Preference Handling Distribution on page 481
- Logical Agent capability on page 481
- Delivering the call to the skill queue on page 482
- Routing the call to an agent on page 487

Administering skills

A skill is an attribute that is:

- Administered as a skill hunt group
- Administered to VDNs (VDN skill preference)
- Assigned to agents (agent skill)

A skill hunt group is administered for each skill. A skill hunt group is a set of agents trained to meet particular customer needs.

Generally, if the ability *Spanish speaking* is assigned to skill 127, for example, it follows that Agent skill 127 and VDN skill 127 both signify *Spanish speaking*. However, note that the agent skill might be assigned a skill term that is broader than that for the corresponding VDN skill. For example, Agent skill 127 might be labeled, *bilingual,* for agents that can handle calls in English as well as Spanish.

Skills for an application are shown in the following table, which presents a very abbreviated example of such a skill distribution for an automobile club.

Example of a skill table for an automobile club

Supergroup-99	
Emergency road service-bilingual-22	Route planning-bilingual-44
Emergency road service-English-11	Route planning-English-33

In the table shown above, five skills are defined. Each skill indicates knowledge or an ability on the part of the agent or a need for knowledge on the part of the caller. One or more of these skills can be attributed to the agent according to the agent's expertise with the corresponding highway services and his or her language-speaking ability. Similarly, one or more of these skills can be considered needs on the part of the caller.

The table shown above, is arranged in such a manner that the agents at the top level have the broadest knowledge, that is, these agents can handle emergency road service and route planning calls and can speak Spanish. The top level (skill group) here is called Supergroup, and it contains agents who, as a group, can take any type of call regarding the automobile club. Accordingly, this skill group serves as a backup skill group. As you descend through the table, each sublevel corresponds to a group of agents who have more specific skills and can therefore take more specialized calls.

Calls can be distributed to the most-idle agent by using either the Uniform Call Distribution (UCD) option or the Expert Agent Distribution (EAD) option. UCD distributes calls from the skill hunt group to the most-idle agent who has this skill assigned at any priority level. This scenario provides a more even distribution to calls and therefore keeps agents equally busy. EAD distributes calls from the skill hunt group to agents to an available agent who has the highest skill level. Skills that are assigned to an agent at higher skill levels indicate a higher level of expertise or preference by the agent than any lower skill level skills that are assigned to that agent. EAD distribution provides the caller with the best or most expert agent match.

Agents are usually given a preference for higher skill level calls. However, the system can be administered to give agents a preference for the greatest need call. The greatest need call is the highest priority oldest call waiting for any of the agent's skills.

Multiple Call Handling on Request and Forced Multiple Call Handling make it possible for an agent to receive additional ACD calls either after putting a call on hold, or when active on another ACD call. Forced Multiple Call Handling can be used to give priority to an ACD call over an in-progress non-ACD call, or to give priority to a call from one skill over an in-progress call from a different skill. For more information, see *Feature Description and Implementation for Avaya Communication Manager*.

To administer skills, set the Skill, ACD, and Vector fields to y. Instructions for completing the Hunt Group screen are included in *Administrator Guide for Avaya Communication Manager*.

VDN skills

EAS enhances the Call Vectoring and Automatic Call Distribution features of the switch by distributing incoming calls based on:

- Specific skills that are assigned to a VDN or used in a vector, and
- Skills that are assigned to an agent

For example, a caller dials a particular number (VDN). The VDN uses a vector to queue the call to an agent with a skill that matches the VDN skill.

You can assign up to three different skills to a VDN in an order that meets your callers' needs. The first skill assigned to a VDN might be the skill that is required to best meet the needs of the customer who called the VDN. The second and third skills assigned to the VDN might represent backup skills that can also meet the callers' needs.

Skills that are administered to a VDN are commonly called VDN skill preferences. VDN skill preferences are labeled 1st, 2nd, and 3rd.

Note:
While skills can be optionally assigned to VDNs, the vector controls when and to what VDN skill the call queues. The application of VDN skills is described later.

The following table shows how skill preferences can be assigned to the five VDNs that are used for the automobile club that we discussed earlier. For each VDN, the corresponding call type and the number of the vector to which the VDN points are indicated. For a description of each skill, see Example of a skill table for an automobile club on page 474.

Example of VDN skill preferences assignments

Call type	Skill Preferences				
	VDN	1st	2nd	3rd	Vector
General number	6005				1
Emergency Road Service (English)	6001	11	22	99	3
Emergency Road Service (Spanish)	6002	22		99	2

Example of VDN skill preferences assignments (continued)

Call type	Skill Preferences				
	VDN	1st	2nd	3rd	Vector
Route Planning (English)	6003	33	44	99	3
Route Planning (Spanish)	6004	44	99		2

In the table shown above, note that two VDNs point to Vector 3, two VDNs point to Vector 2, and one VDN points to Vector 1. Note also that a 1st and 3rd VDN skill Preference, but no 2nd VDN skill Preference, are assigned to VDN 2222. This implies that the call to this VDN (if not already answered) will wait longer before queuing to the backup skill (Supergroup-99, in our example), provided that the vector is designed to execute accordingly.

The following table shows the skill preferences that are assigned for one specific VDN (6003) that is used for the automobile club:

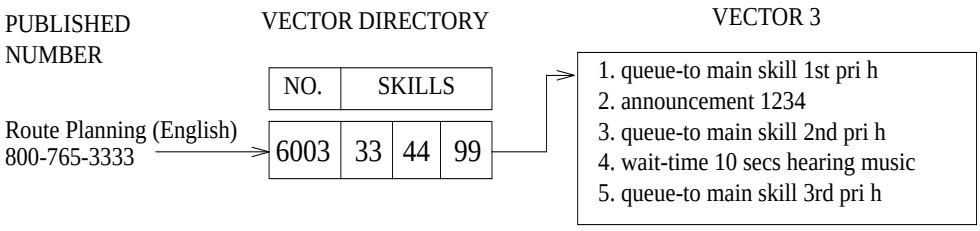
Skill preferences assignments for VDN 6003

Preference	Number	Description
1st:	33	Directed to an agent who is knowledgeable about Route Planning and speaks English
2nd:	44	Directed to an agent who is knowledgeable about Route Planning and is bilingual
3rd:	99	Directed to an agent who can field all calls

In the table shown above, the first VDN skill preference corresponds to a knowledge area that could be considered a subset of the knowledge area that is represented by the second and the third preference. Similarly, the second VDN skill Preference corresponds to a knowledge area that could be considered to be a subset of the knowledge area that is represented by the third preference. Such an approach is commonly used to assign VDN skill preferences. The result of this approach is that the longer a call waits, the larger the pool of agents that the ACD considers for handling the call.

Recall that the vector numbers for each VDN associated with the automobile club are listed in Example of VDN skill preferences assignments on page 475. VDN 6003 points to Vector 3. As such, the skill requirements that are associated with the VDN are forwarded to the vector. This process is shown in the following figure.

Example of VDN skill implementation



Assume that the English-speaking caller needs information on route planning and dials the appropriate number (800-765-3333). Network 800 features direct the call to 6003 (a VDN), the call enters the switch and is directed to VDN 6003, which points to the appropriate vector. As shown in Skill preferences assignments for VDN 6003 on page 476, VDN skill Preferences 33, 44, and 99 are administered as the 1st, 2nd, and 3rd skill preferences, respectively, for VDN 6003.

Vector processing of this application is described in Delivering the call to the skill queue on page 482.

Vector Directory Number (VDN) screen

The Vector Directory Number (VDN) screen shown in the following example is used to administer VDN skills.

Vector Directory Number (VDN) screen, page 1

```
change vdn xxxxx                                                    page 1 of 2
VECTOR DIRECTORY NUMBER

                            Extension: 2001
                                 Name: vdn 2001
                       Vector Number: 1

                 Attendant Vectoring? n
                  Allow VDN Override? n
                                COR: 1
                                 TN: 1
                           Measured: internal
       Acceptable Service Level (sec): 20
              Service Objective (sec):

        VDN of Origin Annc. Extension:
                            1st Skill:
                            2nd Skill:
                            3rd Skill:
```

Vector Directory Number (VDN) screen, page 2

```
change vdn xxxxx                                                    page 2 of 2
VECTOR DIRECTORY NUMBER

                             Audix Name:
                Messaging Server Name:
                   Return Destination:
                VDN Timed ACW Interval:
                       BSR Application:
         BSR Available Agent Strategy: 1st-found
             Observe on Agent Answer?: n
```

Note:

Skills can be optionally assigned to VDNs, however, the vector controls when and to what VDN skill the call queues.

Complete instructions for completing the screen are included in *Administrator Guide for Avaya Communication Manager*.

Call Vector screen

Completion of the Call Vector screen is required for using vectors with EAS. The screen contains three pages. However, if the vector contains 11 or fewer instructions, you need to complete only the first page of the screen, as shown in the following example.

Call Vector screen (Page 1 of 3)

```
change vector 20                                          Page 1 of 3
                           CALL VECTOR
 Number: 20                Name:_____________________
Multimedia? n     Attendant Vectoring? n           Lock? y
     Basic? y  EAS? y   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 ________________
```

Note:
Skills can be optionally assigned to VDNs, however, the vector controls when and to what VDN skill the call queues.

Instructions for completing the Call Vector screen are provided in *Administrator Guide for Avaya Communication Manager*, and in Creating and editing call vectors on page 217.

Agent skills

Agents are trained or hired to accommodate specific caller needs. Agent skills represent and define the ability of the agent to handle calls that require these skills. Agents are assigned skill numbers that are based on such characteristics as training or knowledge, access to systems or information, language ability, and interpersonal traits. Examples of agent skills include the following: speaks Spanish, knows about widget X, can handle complaint calls, has access to a database, and so forth.

You can assign up to 60 skills (with EAS-PHD) or 4 skills (without EAS-PHD). Each of these skills can be designated a skill level between 1 and 6 (EAS-PHD) or 1 and 2 (EAS), with 1 being the highest skill level, which is the highest-priority skill.

If an agent has multiple skills, a single skill group can be created for each set of skills. Agent skills are assigned to agents by completing the Agent Login ID screen. For more information, see the ACD login ID dialing.

It is highly recommended that you create a separate skill hunt group for direct agent calls. Direct agent calls are queued to the skill that is administered as the direct agent skill on the Agent LoginID screen. If an agent is not able to log in to his or her direct agent skill, direct agent calls are queued to the first-administered highest-level skill.

The following table shows the assignment of agent skills. For a description of the skills, see Example of VDN skill preferences assignments on page 475.

Example of agent skill assignments

Agent	Skills assigned			
Jan O'Hara	22 (L1)	44 (L2)		
Sam Lopez	99 (L1)			
Sue Carlson	22 (L1)	11 (L1)	44 (L2)	33 (L2)
Mark Davis	44 (L1)			
Amy Brown	44 (L1)	22 (L2)		

Without EAS-PHD a maximum of four agent skills may be assigned to any one agent with one of two preference levels. With EAS-PHD up to 60 skills can be assigned to each agent with one of sixteen preference levels. The skill assignments table shows that four agent skills (22, 11, 44, 33) are assigned to Sue Carlson. These assignments indicate that Sue is bilingual and can service callers who need emergency road service or information on route planning. Only one agent skill (99-Supergroup) is assigned to Sam Lopez. This means that Sam is serving only as a backup.

A L1 or L2 next to the skill number indicates whether the agent skill is assigned as a level 1 or level 2 skill. For example, Jan O'Hara has *Emergency Road Service-Bilingual* as a level one skill and *Route Planning-Bilingual* as a level two skill. This means that whenever Jan O'Hara becomes available for an ACD call, provided that the Call Handling Preference is skill-level, the ACD software first looks for English-speaking callers who are requesting information on emergency road service from the agent. Only if there are no callers requesting emergency road service does the ACD software look for English-speaking callers who are requesting information on route planning. If the Call Handling Preference is greatest-need, Jan O'Hara receives the highest priority, oldest call waiting for either *emergency road service* or *route-planning bilingual* each time that she becomes available.

For any given application, EAS puts no restrictions on which agent skills can be assigned to an agent.

Note:

Agent skills are administered by completing the Agent Login ID screen. This screen is shown in ACD login ID dialing on page 490. Complete instructions for completing the screen are provided in *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Preference Handling Distribution

Preference Handling Distribution enables an agent to take calls based on either skill level or greatest need.

If an agent's call-handling preference is by skill level, the agent receives the call that requires the skill for which the agent's skill level is highest.

If an agent's call-handling preference is by greatest need, the agent receives the highest-priority, oldest call waiting that requires any of the agent's skills.

It is recommended that in any skill, all agents have the same call handling preference. This ensures the most consistent distribution of calls by either greatest need or skill level.

Preference Handling Distribution Examples

The following table is an example of how calls queue with Preference Handling Distribution.

Preference Handling Distribution

Agent is assigned skills and skill levels...	These calls are in queue...
Skill 11; skill level 1	Waiting 15 seconds; priority medium
Skill 21; skill level 8	Waiting 30 seconds; priority low
Skill 31; skill level 16	Waiting 45 seconds; priority medium

Logical Agent capability

With Logical Agent and EAS, calls are routed to agents based on the login ID instead of the extension number that is assigned to the telephone. The agent's login ID must be consistent with the dial plan of the switch. When an agent logs in to an extension, the login ID overrides the extension as far as ACD tracking and characteristics, such as name and class of restriction (COR) are concerned.

When a specific login ID is called, the switch routes the call to the telephone that the agent is currently logged in to. Logical Agent allows agents to be called regardless of the telephone the agent is using. Calls to agent login IDs can be delivered as direct agent calls with the proper COR set for both the originating and the receiving login ID/facility.

Agents are not assigned to skill hunt groups with Logical Agent. Instead, an agent has specific skills that are assigned to his or her login ID. When an agent logs in, the agent is associated with the assigned skill hunt groups and tracking begins for the assigned skills.

Note:
Avaya CMS automatically measures a logical agent who is administered with at least one measured skill when the agent logs in.

Logical Agent uses a single set of work-mode buttons for all skills. This means that an agent is available or in AUX work for all skills at the same time. An agent cannot be available in some skills and in AUX work in others.

The telephone's button assignments and automatic answer options do not follow the agent because they are associated with the physical extension and not the agent login ID.

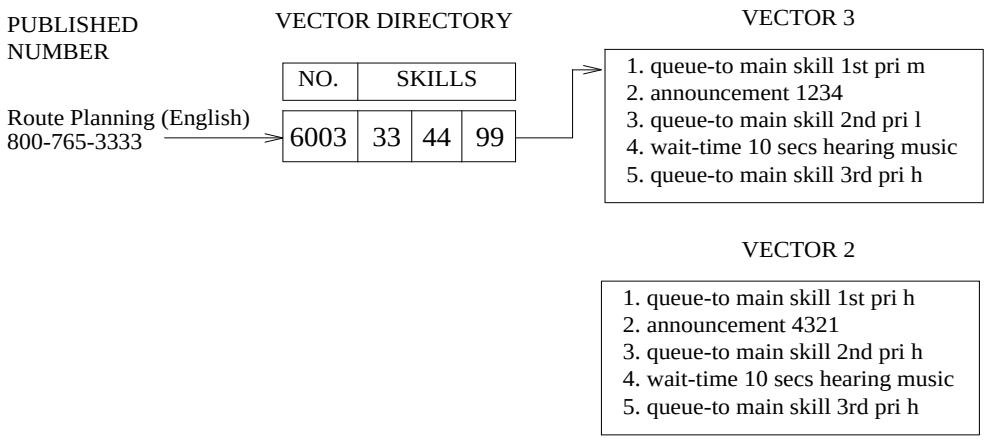
Note:
Converting to EAS may require a change to the CMS login ID if the current ID is not a valid extension number or cannot be made available in the switch dial plan. Agent login IDs are assigned names from the Dictionary-Login Identification window by way of Avaya Supervisor. Login IDs must be different from the telephone extensions.

Delivering the call to the skill queue

This example shows how a call is delivered to a skill hunt group queue by vector processing.

The skills that are assigned to a VDN define the requirements in the vector for routing calls to an ACD agent with a particular set of skills. These skills become active for an ACD call whenever a **queue-to skill** command is executed. The skills also become active whenever a **check skill** command is executed and the threshold condition is met. Once a skill is active for an ACD caller, the call cannot be delivered to an available ACD agent unless the agent also has one of the active VDN skills.

Process for delivery of a call to a skill queue



The process shown above assumes that an English-speaking caller needs information on route planning and dials the appropriate number (800-765-3333). In this case, the call enters the switch and is directed to VDN 6003, which points to Vector 3. Once vector processing starts, the **queue-to skill** command in step 1 queues the call to the skill hunt group that corresponds to the 1st VDN skill (33-Route Planning-English). If an agent with skill 33 is available, this agent answers the call. If such an agent is not available, the call is eventually queued to the skill hunt group that corresponds to the 2nd VDN skill (44-Route Planning-Bilingual) by the **queue-to skill** command in step 3. This time, if an agent with skill 44 is available, this agent answers the call. If the call is still not answered, the call is eventually queued to the skill hunt group that corresponds to the 3rd VDN skill (99-Supergroup) by the **queue-to skill** command in step 5.

In the process shown above, Vector 2 would be executed if a Spanish-speaking caller had called into the switch. Accordingly, the announcement that is provided in Vector 2 is in Spanish, whereas the announcement in Vector 3, which is executed in our example, is in English.

Note also that each of the **queue-to skill** commands in Vector 2 queues the call at a high priority, whereas only one of the **queue-to skill** commands in Vector 3 queues the call at this high a priority level. The strategy presented here is valuable when there is a limited number of bilingual agents because the bilingual such agents will be available more quickly to service callers who speak only Spanish.

VDN skills can also be used in **check skill, messaging skill,** and **converse-on skill** commands. Within any of these commands, a specific skill number can be used instead of a VDN skill Preference, provided that the relevant skill hunt group is correctly administered. For example, step 5 might have read **queue-to skill 99 pri h**. This concept is discussed further in Super agent pool on page 485.

Procedure using Call Prompting

The procedure that is described in the previous section can be enhanced by using Call Prompting. For example, the user can dial a general telephone number whose VDN points to a Call Prompting vector.

Staying with our automobile club example, recall that in Examples of services and corresponding DNIS digits on page 469, we define 800-765-5555 as the general telephone number for the service. Recall also that in Example of VDN skill preferences assignments on page 475 we identify 6005 as the VDN for this 800 number. Also, we indicate that VDN 6005 points to Vector 1.

The following vector shows how Vector 1 might appear.

Call Prompting vector for the automobile club

```
1.  wait-time 0 seconds hearing ringback
2.  collect 1 digits after announcement 5678
    [For emergency road service, dial 1.
    Para asistencia con su automovil, marque el dos.
    For travel route directions, dial 3.
    Para informacion sobre rutas, marque el cuatro.]
3. route-to number 6001 with cov n if digit = 1
    [English Emergency Road Service VDN]
4. route-to number 6002 with cov n if digit = 2
    [Bilingual Emergency Road Service VDN])
5. route-to number 6003 with cov n if digit = 3
    (English Route Planning VDN)
6. route-to number 6004 with cov n if digit = 4
    (Bilingual Route Planning VDN)
7. route-to number 6002 with cov n if unconditionally
    [Bilingual Emergency Road Service VDN]
```

Once the caller dials 800-765-5555, the call enters the switch and is directed to VDN 6005, which points to our Call Prompting vector. At this point, vector processing begins. Step 1 provides ringback if the caller has to queue for the announcement in step 2. The **collect digits** command in step 2 first provides an announcement that requests the caller to dial 1, 2, 3, or 4, depending upon the caller need and the caller's language speaking ability. If the caller dials a digit that is other than one of the four specified, each of the **route-to...if digits** commands in steps 3 through 6 fails, and control is passed to the **route-to...if unconditionally** command in step 7, which unconditionally routes the call to VDN 6002. This VDN is assigned the *bilingual emergency road service* skill and points to Vector 2, which is provided in the previous section.

Now we return to the **collect digits** step and assume that the caller dials 4. In this case, steps 3 through 5 fail because the required digit (1, 2, or 3, respectively) was not dialed. Thereafter, control is passed to step 6, where the **route to...if digit** command finds a digit match and consequently routes the call to VDN 6004. This VDN is assigned the *bilingual route planning* skill and also points to Vector 2, which is provided in the previous section.

Note:

VDN Override applies to the skills that are assigned to the VDN. For more information, see VDN Override on page 39.

Super agent pool

EAS allows a skill hunt group to function as a super agent pool. A super agent pool is a backup group of one or more agents that is able to handle many if not all types of calls coming into the application. In our automobile club examples, Skill Hunt Group 99 (Supergroup) serves as a super agent pool. Also, you might recall that 99 appears as both a VDN skill and an Agent skill. However, a super agent pool can be assigned a skill hunt group number that is not assigned to a VDN skill. This can and should be done whenever the application requires four levels within the skill table distribution, as shown in the following table.

Modified skill table for the automobile club

Supergroup-99			
Emergency road service- bilingual-88		**Route planning-bilingual-77**	
English-66	Spanish-55	English-44	Spanish-33
Bostonian-11	Castilian-13	Bostonian-15	Castilian-17
New Yorker-12	South American-14	New Yorker-16	South American-18

Besides a new skill numbering scheme, our modified skill table has four levels instead of the three levels that are provided in Example of a skill table for an automobile club on page 474. Except for the skill numbering scheme, the top two levels (Supergroup-99 and Emergency Road Service-Bilingual-88/Route Planning-Bilingual-77) remain unchanged. However, note that the next level is reorganized into segments to indicate the ability to speak English or Spanish. Finally, note that a new level is added to denote particular types of accents or pronunciation in English and Spanish.

The following table shows how some of the skills in Modified skill table for the automobile club on page 485 are administered to one relevant VDN (VDN 1616).

VDN 1616 - Skill preferences		
1st:	16	Knows about Route Planning, speaks English, has New York accent
2nd:	44	Knows about Route Planning, speaks English
3rd:	77	Knows about Route Planning, is bilingual

Now we are ready to consider the following vector to accommodate a super agent pool.

Modified vector to accommodate a super agent pool

```
1. queue-to skill 1st pri m
2. announcement 4555
3. queue-to skill 2nd pri l
4. wait-time 10 seconds hearing music
5. check skill 3rd pri l if calls-queued < 3
6. announcement 4666
7. check skill 99 pri l if available-agents > 0
```

Assume an English-speaking caller needs information on route planning and want to speak to an agent with a New York accent. In this case, the caller dials the appropriate number (800-765-1616, for example). Accordingly, the call enters the switch and is directed to VDN 1616, which points to the vector in the previous screen. 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 (New Yorker-16). If an agent with skill 16 is available, this agent answers the call. If such an agent is not available, the call is eventually queued to the skill group that corresponds to the 2nd VDN skill (English-44) by the **queue to main skill** command in step 3. This time, if an agent with skill 44 is available, this agent answers the call. If the call is still not answered, the **check skill** command in step 5 attempts to queue the call according to the parameter indicated (if calls-queued < 3) to the skill group that corresponds to the 3rd VDN skill (Route Planning-Bilingual-77). If the call is queued, and if an agent with skill 77 is available, this agent answers the call. If the call is not queued, or if it is queued and an agent with skill 77 is not available, the **check skill** command in step 7 is executed.

Before we discuss the execution of step 7, note that a specific skill hunt group number (99) and not a VDN skill Preference designation (1st, 2nd, or 3rd) is included within the **check skill** command. Since the skill table for the application involves four levels of skills, and since there can be no more than three VDN skills, the specific skill group number (99) for the super agent pool must be included within the queuing command to allow caller access to the pool. Whereas a VDN skill is always represented in a vector by the term 1st, 2nd, or 3rd, a super agent pool is always represented by a whole number according to the parameters of the relevant switch. For the queueing commands, see Call Vectoring commands on page 521.

Returning to the vector execution, the **check skill** command in step 7 attempts to queue the call according to the parameter that is indicated (if available-agents > 0) to the super agent pool (Supergroup-99). If the call is queued, and if an agent in the super agent pool is available, this agent answers the call.

Note:

If the call has already queued to all three VDN skill hunt group preferences, it does not queue to the specific skill hunt group. This reflects the restriction that a call can only queue to a maximum of three splits or skills. The best approach is to test the splits/skills first to determine where to queue the call. Also see Expected Wait Time (EWT) on page 259.

Routing the call to an agent

With EAS optioned, an agent becomes associated at login with one or more skill hunt groups. A single set of work mode buttons applies to all the skills that are assigned to a logged-in agent. For example, if the agent selects Aux Work, the agent is in Aux Work for all the skills associated with the agent. Therefore, logged-in agents need only a single set of work-mode buttons for all relevant skill hunt groups.

Calls can be routed to the agent from a skill hunt group by dialing an agent login ID or by dialing an agent telephone extension directly. The following sections discuss these procedures.

Delivery from a skill hunt group

An incoming call is matched to an agent who has at least one of the three VDN skills that are required to handle the call. This matching is done by ACD queuing and the **`queue-to skill`**, **`check skill`**, **`messaging skill`**, or **`converse-on skill`** commands in the vector. If more than one agent is available for a call, the call is delivered according to whether EAD or UCD is administered for the skill hunt group.

For any one login session, an agent can have a maximum of four skills, or a maximum of twenty skills with EAS-PHD. Each agent skill is administered with a skill level.

Remember that when the Call Handling Preference is administered as greatest need, the agent receives the highest priority oldest call waiting for any of the agent's skills. If the Call Handling Preference is skill-level, the ACD software distributes the call that is waiting for the agent's highest skill-level skills whenever the agent becomes available. If no calls are waiting for the highest skills, the queued calls for the next highest skills are distributed to the agent, and so on. The following scenario describes call distribution when the Call Handling Preference is skill level.

Once an agent becomes available, he or she receives a waiting call in the following order:

1. Oldest direct agent call waiting for the agent if the direct agent skill is administered at the agent's highest skill level
2. Oldest call waiting at the highest priority for the highest skill-level skill
3. Oldest call waiting at the next highest skill-level skill, and so on.

For example, assume that Jill is the only agent with skills 22 (L1), 13 (L1), 23 (L1) and 47 (L2). Also assume that, while Jill is in AUX work mode, five calls are queued, as shown in The following table, which also shows the skill level and priority level that are associated with each call:

Example of skill call queue sequence

Call	Time in queue	Skill number	Priority level
A	8:00	13	Medium
B	8:01	47	Top
C	8:02	23	Direct Agent
D	8:03	22	Top
E	8:04	22	Medium

Given this scenario, the next table indicates and explains the order in which Jill handles the five calls.

Example of skill call distribution for a single agent

Call handled	Reason
C	Only direct agent call queued at highest level skill.
D	Oldest call waiting at the highest priority for highest skill-level skills (Call B has the same priority level (Top), but it is assigned a lower skill level (47). Also, Call E has the same skill (22), but it has a lower priority level (Medium) and has not been waiting as long as Call D).
A	Oldest call waiting at the highest priority level for highest skill-level skills (Call E also has a primary skill (22) and the same priority level as Call A, but Call A has been waiting four minutes longer than Call E).
E	Only remaining call with the highest skill level (22) (Call B has a lower skill level (47)).
B	Last remaining call, and the only one that has the lower skill level (47).

If no calls are waiting when an agent becomes available, the agent is placed into the agent queue according to the call distribution method that is in effect. For UCD, the agent is placed at the bottom of the most-idle agent queue. For EAD, the agent is placed at the bottom of the agents with the same skill level.

The following table shows a call scenario that is valid for either UCD or EAD.

Example of UCD/EAD call scenario

Time	Event	Skills
9:00	Jill logs in	22(L1), 13(L1), 47(L2)
9:01	Jill available	22(L1), 13(L1), 47(L2)
9:02	Jack logs in	22(L1), 47(L1)
9:03	Jack available	22(L1), 47(L1)
9:04	Call A arrives	47
9:05	Call A drops	47
9:06	Call B arrives	13
9:07	Call B drops	13
9:08	Call C arrives	22

Given the scenario presented above, the following table shows how Calls A, B, and C are distributed by UCD and EAD:

Example of call distribution by UCD and EAD

Time	UCD **or** EAD**?**	Result	Reason
9:04	UCD	Jill receives Call A.	Jill is the most idle agent for skill 47.
	EAD	Jack receives Call A.	Jack is the more expert agent because he has skill 47 as a level 1 skill whereas Jill has skill 47 as a level 2 skill.
9:06	UCD	Jill receives Call B.	Jill is the only agent who is logged in to skill 13.
	EAD	Jill receives Call B.	Jill is the only agent with skill 13.
9:08	UCD	Jill receives Call C.	Jill is the most idle agent for skill 22. She receives Call C even if she handled Call A.
	EAD	Jill receives Call C.	Both Jill and Jack have skill 22 as a level 1 skill, but Jill has been logged in 2 minutes longer than Jack; that is, she is the most idle agent.

ACD login ID dialing

The ACD login IDs used in EAS are extension numbers that are included in a station numbering plan but not administered as stations. These IDs are administered by using the Agent Login ID screen, as shown in the following example. If EAS-PHD is not optioned, you can only administer four skills.

Agent Login ID screen

```
add agent-loginID 9011                                        Page  1 of  1
                              AGENT LOGINID

                Login ID: 9011_                                  AAS? _
                    Name: ____________________________         AUDIX? _
                      TN: 1__                          LWC Reception: spe
                     COR: 1                  AUDIX Name for Messaging: _______
           Coverage Path: ____    Messaging Server Name for Messaging: _______
           Security Code: ____                LoginID for ISDN Display? n
      Direct Agent Skill: ___                                Password: _________
Call Handling Preference: skill-level           Password (enter again): _________
       Service Objective? _                                Auto Answer: ______

   SN  RL SL  PA      SN  RL SL  PA       SN  RL SL  PA       SN  RL SL  PA
1: __  _  __  ___  6: __  _  __  ___  11: __  _  __  ___  16: __  _  __  ___
2: __  _  __  ___  7: __  _  __  ___  12: __  _  __  ___  17: __  _  __  ___
3: __  _  __  ___  8: __  _  __  ___  13: __  _  __  ___  18: __  _  __  ___
4: __  _  __  ___  9: __  _  __  ___  14: __  _  __  ___  19: __  _  __  ___
5. __  _  __  ___ 10: __  _  __  ___  15: __  _  __  ___  20: __  _  __  ___

      WARNING:  Agent must log in again before skill changes take effect
```

With EAS, an agent's ACD login ID is associated with a specific telephone only when the agent actually logs in at that telephone. When the agent logs off, the association of the agent's ACD login ID with a specific telephone is removed. If an agent does not answer a call, or if the agent is logged out, the call goes to the busy points on the coverage path.

When the agent logs in, the telephone display indicates the agent's skill assignments.

The agent logs in by doing the following:

- Going off-hook or selecting a line appearance
- Upon hearing the dial tone, entering the login Feature Access Code (FAC) or selecting the Login Abbreviated Dialing button
- Upon hearing the dial tone, entering the 1-digit to 5-digit login ID

  Note:
  If someone is already logged in at that telephone, the agent hears an intercept tone.

- Upon hearing the dial tone, entering (optionally) the 0-digit to 9-digit password.

Note:
If the agent is using a DCP telephone (such as a Callmaster), then the password digits are not shown unless an abbreviated dial button is used. BRI telephones show the password digits.

Once the login is accepted, confirmation tone is given. Also, the skills that are assigned are displayed for 5 seconds on the telephone display. If more skills are assigned than can be displayed, a plus sign (+) appears at the end of the display. If a skill is administered but the agent was not logged in to the skill, the skill number is displayed with a star (*). The previous login sequence allows an ACD call to be directed to a specific agent and to have that call tracked and treated as an ACD call.

When an EAS agent logs in to a station with the station administered for audible message waiting, the agent receives an Audible Message Waiting tone only when calls are waiting for the agent login ID extension. When the agent logs out, Audible Message Waiting tone then applies again to messages that are waiting for the physical extension. This field has no impact on whether an agent hears the EAS Login-ID Message Waiting tone during the login process.

The message waiting lamp by default tracks the status of messages that are waiting for the logged-in EAS agent LoginID rather than messages for the physical telephone. The operation of the Message Waiting Lamp can be changed so that it tracks the status of messages that are waiting for the physical telephone where the agent is logged in. For more information, see the Feature-Related System-Parameters screen in *Administrator Guide for Avaya Communication Manager*.

Other agent login capabilities

In addition to skill assignments, the following capabilities are associated with agents' login IDs.

Call routing : A call to the login ID reaches the agent independent of the telephone that the agent is currently using. In other words, such a call is sent to the telephone at which the agent is currently logged in.

If the proper Class of Restrictions (COR) is set, callers can initiate a direct agent call either by dialing the login ID extension directly or by calling a VDN that points to a vector that contains first a prompt for the login ID and then a **route-to digits** command. This allows external callbacks by way of Direct Inward Dialing (DID) or an 800 number. Both the receiving agent's login ID COR and the originator's (caller's) COR must have Direct Agent Calling (DAC) set to y. The caller's COR is for the following:

- Telephone extension (for internal calls or transfers)
- Trunk group (for DID calls)
- VDN (for prompted calls)

If the call covers or is forwarded, the COR of the originator (or VDN) and the final agent is used. All feature functionality for ACD calls, except Queue Status indications, is available for direct agent calls.

Internal and external users can originate direct agent calls by dialing the agent's login ID. Also, DAC can be used to transfer ACD calls from one agent to another agent.

If an agent who is receiving the direct agent call is staffed but unavailable, the call waits in front of the skill calls in the skill that is administered as the agent's direct agent skill until either the call is answered or a coverage timeout occurs. Also, the caller hears an optional direct-agent announcement that is followed by music or silence. There is one direct agent announcement per system. The agent, on the other hand, receives a ring-ping, and the current work mode button flashes. If the agent is available, the call is delivered to the agent according to the answering and ringing options. Calls are answered and handled in the same manner as ACD calls. For more information, see the Feature-Related System-Parameters screen in *Administrator Guide for Avaya Communication Manager*.

Login ID name on the telephone display: A call to a logged-in EAS login ID by default displays the name associated with the login ID and not the name that is associated with the telephone. This is also true on the receiving party's display for a call that is made from a telephone with an agent logged in. However, the user can display the name of the physical telephone where the EAS agent is logged in. The user must be active on a call with the agent, and must have a telephone with an alphanumeric display and an inspect button. When the inspect button is pressed during a call to or from the EAS agent, the physical telephone name of the agent is displayed.

Coverage path: Call coverage can occur whether or not the agent is logged in. If the agent is not logged in, the busy criteria is met and the call follows the points on the coverage path. If the agent is logged in but fails to answer, the don't answer criteria is met and the call follows the points on the coverage path. A call to the login ID goes to the coverage path that is assigned to the login ID rather than to the coverage path that is assigned to the telephone extension.

Agent restrictions: A call to the login ID or from the agent uses the restrictions that are associated with the agent and not the telephone.

Telephones are fully functional if an agent is not logged in. The restrictions, coverage, and name revert to the telephone administration when the agent logs out.

If a number of users are sharing one telephone (due to job sharing or shifts, for example), a unique login ID extension is assigned to each user. Therefore, whenever a user is logged out, any calls to that user (login ID) are sent to his or her coverage path. As a result, login IDs can be used to reach people independent of where they happen to be. Such people include those who use more than one phone because they have more than one office or (in the case of security guards, for example) sit at more than one desk.

Because AAS/messaging-system ports are not mobile, these ports are administered to agent login IDs. Whenever the **AAS** or **AUDIX** field is set to **y**, a field that requests the port number is brought up, and the **password** field disappears.

EAS feature interactions

This section discusses the feature interactions that involve EAS. Unless otherwise specified, the feature interactions for skill hunt groups are the same as for vector-controlled splits.

Abbreviated Dialing: Abbreviated Dialing is used to log in or log out EAS agents. Abbreviated Dialing lists or buttons can be administered only for stations.

Administration Without Hardware: Although EAS login IDs are extensions without hardware, they are not a part of the Administration Without Hardware (AWOH) feature.

Agents in multiple splits feature: With EAS, the Agents in Multiple Splits feature is called Agents in Multiple Skills. This feature allows an EAS agent to be logged in to multiple skills.

Agent work modes: With EAS optioned, an agent can be in only a single work mode for all skills at any one time. For example, an agent cannot be in AUX work mode in one skill hunt group and also available in another skill hunt group. Also, if the After Call Work (ACW) mode button is selected, the agent is placed into ACW for the first skill that is administered and logged in to.

Assist: This feature is used for skill hunt groups (that is, there is one supervisor per skill hunt group). A telephone can be administered with one or more Assist buttons for each skill that agents who are using the telephone might have. An Assist button can also be administered with no associated skill. In this case, the supervisor for the skill that the agent is currently active on is called. If the agent is not active on any skill, the supervisor for the agent's first skill is called.

Any assist button that is selected is tracked as an assist for the current call, regardless of any skill that is assigned to the button. The administered association of an Assist button with a particular skill and assigned supervisor is not affected when an EAS agent logs in to that station.

Audible message waiting: If messages are waiting for an EAS agent login-ID extension, an agent hears a special 5-burst EAS Login-ID Message Waiting tone (instead of confirmation tone) after successfully logging in. This does not require Audible Message Waiting to be assigned to the telephone or the system.

If Audible Message Waiting is optioned for the system and assigned to an agent's telephone, and messages are waiting for the agent login ID extension, the agent hears the Audible Message Waiting tone whenever the agent goes off-hook, or selects a line appearance and hears dial tone. Messages that are waiting for the physical extension do not cause an Audible Message Waiting tone when an EAS agent is logged in.

Auto-Available Skills: If a skill hunt group is administered as an Auto-Available Skill (AAS), the EAS login IDs that are assigned to this skill must also be administered as Auto-Available. When the switch reinitializes, these login IDs are automatically logged in with the auto-in work mode. If any switch features attempt to change the work mode to anything except auto-in, this attempt is denied. Agents cannot have both Auto-Available and Non-Auto-Available Skills. This feature is not intended for human agents.

Automatic answering with zip tone: This feature can be administered only for a physical extension. The feature is not associated with a LoginID.

BCMS: The BCMS user interface remains the same when EAS is optioned. The only change is that the labeling of the headings is changed from split to skill. When EAS is enabled, BCMS agent reports are based on the agent login IDs.

BCMS tracks direct agent calls as skill calls. direct agent calls affect ACD talk time, ACW time, and Average Speed of Answer. Whenever direct agent calls are waiting, BCMS displays an asterisk (*) immediately after the CALLS WAITING column.

Best Service Routing (BSR): EAS VDN skills (1st, 2nd, 3rd) can be used in **`consider split/skill`** commands. EAS skills levels are used for the EAD-MIA and EAD-LOA BSR Available Agent Strategies.

Bridging: ACD calls do not alert on bridged appearances. However, bridged users can activate features on behalf of agents. Features that can be activated include log in, log out, change work modes, and assist.

Call coverage: Call coverage can occur whether or not the agent is logged in. If the agent is not logged in, the busy criteria is met and the call follows the points on the coverage path. If the agent is logged in but fails to answer, the don't answer criteria is met and the call follows the points on the coverage path. A call to the login ID goes to the coverage path that is assigned to the login ID rather than to the coverage path that is assigned to the telephone extension.

Call Detail Recording (CDR): For skill calls, the **called party** field can optionally be the agent login ID.

Call forwarding: Since they are vector-controlled, skill hunt groups cannot be call forwarded. EAS agent login IDs cannot be forwarded, but the physical extension where the EAS agent is logged in can be forwarded. If another station with console permissions tries to forward an EAS login ID, an intercept tone is given.

Call park: To retrieve a parked call by a Feature Access Code (FAC), the agent dials the Answer-Back FAC and the extension where the call is parked. If the person who is unparking the call dials the Answer-Back FAC and the physical extension of the station where the call is parked, he or she is connected to the parked call.

In some cases, the person who is unparking the call may also be able to dial the Answer-Back FAC and the logical agent extension of the agent who parked the call. This operation is possible if the Class of Restriction (COR) of both the agent parking the call and the telephone or agent who is unparking the call have a COR with the DAC flag set to y. If the telephone that is unparking the call is not a logged-in agent, the telephone must have a COR with DAC set to y. If the station that is unparking the call is a logged in agent, then the COR of the logical agent extension must have DAC set to y.

Call pickup: Skill hunt group extensions and EAS login ID extensions cannot be members of a call pickup group.

Class of Restriction: Skill hunt groups do have a Class of Restriction (COR). The COR is used if the skill hunt group extension is called directly. The COR for an EAS agent login ID overrides the physical extension's COR of the telephone that an agent logged in to.

Class of Service: EAS agents do not have a COS associated with their login ID. Instead, the COS is associated with the physical extension. Therefore, the COS of the telephone is not affected when an EAS agent logs in to that telephone.

Dial plan: Agent login IDs are part of the dial plan, and they reduce the total number of stations.

Direct Agent Calling (DAC): If a called EAS Agent login ID and the call originator (extension, trunk, or VDN) both have a COR that allows direct agent calls, the call to the login ID is treated as a direct agent call. A call to the telephone extension where an EAS agent is logged in, or a call to an EAS agent login ID where either the originator's or the login ID's COR does not allow direct agent calls, is treated as a personal (non-ACD) call.

Leave Word Calling: When an EAS agent is logged into a station, the agent can only retrieve LWC messages left for that agent's login ID. To retrieve LWC messages left for that station, the agent must log out.

When an EAS agent is logged into a station, its Message lamp defaults to tracking the status of LWC messages waiting for the station. However, you can assign the Message lamp to track the status of LWC messages waiting for the agent's login ID.

Look-Ahead Interflow: Skills are not sent to another system when a call interflows using Look-Ahead Interflow (LAI). If skills have the same meaning on both ACDs, a LAI command to a VDN with the same skills assigned can provide a mapping of the skills.

Multiple Split Queuing: When EAS is enabled, the Multiple Split Queuing feature is called Multiple Skill Queuing, which has the same functionality. With Multiple Split/Skill Queuing, a call can queue to a maximum of 3 splits/skills.

OCM/EAS: If EAS is enabled on the switch, the Outbound Call Management (OCM)/Expert Agent Selection (EAS) feature is required for a CallVisor ASAI adjunct application to launch predictive Outbound Call Management (OCM) calls. Predictive Calling is an OCM feature that is often used in applications, such as sales or *cold calling*, where it does not matter which agent is accessed by a caller and for which it is important to keep the agents utilized fully.

While OCM predictive calling is an outbound call management application, the EAS environment provides a number of desirable features for inbound call handling. The OCM/EAS feature allows the customer to enable both types of call handling on the switch. From a technical standpoint, if EAS is enabled, the feature is needed for the following reasons:

- All skill hunt groups are vector controlled. However, to launch a predictive OCM call in a traditional ACD environment, the ACD split cannot be vector-controlled.
- The traditional ACD environment and EAS cannot be enabled on the switch at the same time.

The OCM/EAS feature extends the ASAI features to include launching predictive OCM calls from a VDN extension. Previously, ASAI hosts could launch predictive calls only from ACD split extensions. A limited number of Call Vectoring commands are supported in the VDNs that are used to launch or process OCM predictive calls. These commands are listed in the following section.

Commands for OCM predictive calls

Vectors that are intended for processing predictive calls must be designed in such a manner that the vectors are limited to the supported steps.

The following table lists vector commands available for processing OCM Predictive Calls and provides a brief comment for each command.

Commands for OCM predictive calls

Command	Comment
queue-to skill (single occurrence)	This command queues the call for handling by an agent in the skill pool.
announcement	This command plays an announcement if there are no agents available and if the queue (if any) is full.
stop	This command ends vector processing. The command also disconnects any call that is not queued.
adjunct routing link	EAS supports adjunct routing to any of the following: skill extension, direct agent call, announcement, or local extension. The command does not involve routing to an external number.
wait-time	NOTE: This command is used with the **adjunct routing link** step to determine how long the switch waits for an adjunct route before continuing with vector processing.

Queue Status Indications: Physical extensions can be administered with Queue Status Indicator buttons and lamps for skill hunt groups that operate in the same manner as split Queue Status Indicators for traditional ACD splits. As long as enough buttons are available, Queue Status Indicators can be administered for all skills that are needed by agents who are using that physical extension. Also, any waiting direct agent calls are not reflected in the queue status indicators.

Reason codes: With Reason Codes, an EAS agent can enter a numeric code that identifies his or her reason for entering AUX work mode, or for logging out.

Remote Service Observing: Remote access to the Service Observing (SO) FACs can be provided by the Remote Access feature or through Service Observing vectors. For additional information, see Service Observing routing on page 625 and Creating Service Observing vectors on page 248.

Service Observing: Service Observing is activated in the EAS environment by dialing either the physical extension of the telephone where an EAS agent is logged in, or the EAS agent login ID.

Telephone displays: When an EAS agent is logged in, the display for originators who call the login ID shows the login ID and agent name as they are administered on the Agent Login ID screen. Calls that are originated by the agent show the agent login ID and agent name at the receiving telephone's display. However, the user can display the name of the physical telephone where the EAS agent is logged in. The user must be active on a call with the agent, and must have a telephone with an alphanumeric display and an inspect button. When the inspect button is pressed during a call to or from the EAS agent, the physical telephone name of the agent is displayed. Calls to the physical extension show the physical extension's number and name on the originator's display.

Tenant Partitioning and agent skills: The Tenant Partitioning feature was designed to support multiple customers using the same Communication Manager server. Tenant Partitioning separates entities, thereby avoiding or reducing interactions between entities in different partitions.

Assign the same partition number to agents, groups, and entities to avoid blocking calls and to avoid any unexpected interactions that result from mixing tenant partitions. When Tenant Partitioning is active and used for restriction of service, assign the same partition number to:

- ACD agents
- Hunt groups (splits or skills)
- Other entities that are involved with ACD agents and hunt groups, such as VDNs and announcements

Note:
An agent's skill set should contain only skills belonging to the *same* tenant partition; not doing so can result in unintended behavior.

VDN Override: If VDN Override is set to y (yes) on the previous VDN, the VDN skills of the current VDN are used. If VDN Override is set to n (no) on the previous VDN, the VDN skills of the previous VDN are used.

VuStats: VuStats can display information for all 20 agent skills.

Work mode buttons: Only a single set of agent work mode buttons is needed. If multiple buttons are assigned, all lamps for that work mode, for example, manual-in, light whenever any one button is pushed.

EAS adjunct interactions

This sections describes adjunct interactions with the EAS feature and includes the following topics:

- ASAI interactions with EAS on page 498
- Messaging system on page 501
- CMS on page 501
- Speech-processing adjuncts on page 501

ASAI interactions with EAS

ASAI support for EAS may be organized into the following categories: call control, feature requests, value queries, event notification, and adjunct-controlled skills. This section provides a high-level overview of the behavior of ASAI in the EAS environment and includes the following topics:

- Call control on page 498
- Feature requests on page 499
- Multiple monitors on page 500
- Value queries on page 500
- Event notification on page 500
- Adjunct-controlled skills on page 500

Call control

Call-control capabilities work exactly the same in the EAS environment as in the traditional ACD environment except for the following:

- User-classified third-party make calls (calls classified by the originator) may originate from an EAS login ID and terminate to a login ID. User-classified calls that terminate to a login ID are given the same direct agent treatment that is provided for such calls that are dialed from a station extension.
- Switch-classified third-party make calls, which are classified by a call classifier board and delivered (when answered) to the originating hunt group, may originate from or terminate to EAS login IDs.

- Direct agent third-party make calls, which are ACD calls that are terminated to a selected member of an ACD skill group, may be requested by including a direct agent option, an agent's physical extension and a skill group extension (compatibility mode), or by requesting a user-classified third-party make call with a login ID destination. The primary differences between the two methods of requesting direct agent calls are that the compatibility mode allows the adjunct to specify the skill hunt group to which a given direct agent call is queued and that the non compatibility mode allows the adjunct to direct the call to a login ID, regardless of which station an agent is logged in to. Direct agent third-party make calls may not originate from an EAS login ID.
- Supervisor assist third-party make calls, which are supervisor assist calls that are originated by a selected member of an ACD split, may originate from an EAS login ID, and they may terminate to an EAS login ID. Unlike dialed direct agent calls, supervisor assist calls that are terminated to a login ID behave as though they have been previously directed to the requested login ID's physical extension. For example, they do not cover if the requested agent is not logged in and if the originator's display shows the agent's physical extension and not the agent's login ID.
- Extension (Domain) control may not be requested for an EAS login ID, but it may be requested on behalf of a Logical Agent's physical extension. Auto-dial calls, which are calls that are initiated by an extension-controlled station, may be terminated to an EAS login ID, in which case the call is given direct agent treatment.
- Adjunct-routing calls, which are vector calls that are routed by an ASAI adjunct by the **`adjunct routing link`** Call Vectoring command, are similar to third party make calls. Such calls may include a direct agent option, an ACD agent's physical extension, and a skill extension. If this is true, these calls are given compatibility mode direct agent treatment and may be terminated to an EAS login ID (in which case they behave like dialed direct agent calls.
- If EAS is optioned, ASAI launches OCM switch-classified or predictive calls from a VDN extension by the OCM/EAS feature. To launch a predictive call in a traditional ACD environment, an adjunct OCM application sends an ASAI request to the switch with an ACD split number as the originating number. The application also sends flags that identify the call as a switch-classified call. In the traditional ACD environment, the ACD split cannot be vector-controlled.

Feature requests

In the EAS environment, agent login, logout and change work-mode requests are fully supported. Agent login requests must contain an EAS agent login ID and optional password (delimited by '#') in the login request's user code IE. Agent logout requests and change work-mode requests may contain the desired agent's physical extension or login ID. Call Forwarding and Send all Calls feature requests are denied for EAS login IDs but may be requested for EAS physical extensions where an EAS agent is logged in.

Multiple monitors

Multiple Monitors provides the ability for up to three ASAI applications to monitor the same ACD Split or VDN domain.

This is helpful in environments were OCM is primary and it can also be used to add an OCM application to launch calls at off-peak times without disrupting the primary application in any way. Multiple Monitors can also be used to monitor an ACD split over 2 links in call environments where ASAI link failure recovery is important.

Value queries

Value queries function identically in the EAS and traditional environments, except that the Extension Type/Class Information Query returns a new indication that a requested extension is an EAS login ID along with an indication of whether the login ID is currently logged in and where, in other words, at which physical extension.

Event notification

Because all skill hunt groups are vector controlled, event notification may not be requested on the basis of a skill hunt group extension. Event notification may, however, be requested on the basis of a controlling VDN extension. Generally, all event reports that involve EAS agents contain the agent's physical extension rather than the agent's login ID.

Adjunct-controlled skills

Agents with adjunct-controlled skills are considered to be adjunct-controlled agents. Adjunct-controlled agents exhibit the same behavior as agents within adjunct-controlled splits in the traditional ACD environment. The following list provides more details:

- Stations are locked for all logged-in adjunct-controlled agents. The only action an agent can take from the station is to go on hook (or unplug the headset) from an auto-answer station, which causes the agent to be logged out.
- Stations are unlocked whenever the controlling adjunct's ASAI link stops functioning. Stations are locked again when the adjunct's link is reestablished.
- The adjunct controls all skill and agent activities such as login, logout, and change work-mode (with the exception of agent logout using the telephone hook).
- Only adjunct-controlled calls can terminate to the extension of an adjunct-controlled agent.
- Only adjunct-controlled calls can terminate to an adjunct-controlled skill hunt group extension.
- Adjunct-controlled EAS Agents can be administered with only one skill. Accordingly, EAS agents may not mix adjunct-controlled and non-adjunct-controlled skills.

Messaging system

Calls to the EAS agent login ID can cover to the messaging system. Each agent must enter his or her agent login ID when calling the messaging system to obtain messages.

Messaging-system agents are assigned to EAS agent extensions. These login IDs are used for CMS and BCMS tracking if the associated messaging-system skill hunt group is externally measured. The aut-msg-wt button or message waiting light can be used to indicate that the login ID has a message.

An agent cannot have both messaging-system and non messaging-system skills.

CMS

Note:
CMS reports show only the first 15 skills that an agent is logged into.

The following items apply to Avaya CMS Agent Tables:

- Separate direct agent database items starting with *DA_* are tracked.
- Standard reports combine statistics for direct agent calls and skill calls. However, reports can be customized to separate these statistical groupings.

The following is true for the CMS Skill Tables:

- Skill queues can be monitored for direct agent calls on the Queue/Agent Summary report.
- Direct agent calls are not tracked.
- Agent time while on a direct agent call is tracked as other time.
- Non-ACD calls while in direct agent ACW are tracked.

The following item is true for the CMS VDN/Vector Tables:

- Direct agent calls and skill calls are combined as ACD calls.

Speech-processing adjuncts

Speech-processing adjuncts that have a line interface to the communication server are able to initiate direct agent calls by dialing the login ID for an agent.

Listing Agents Logged into a Split or Skill: When administering a split or skill, you can use the **`list members hunt-group`** command to verify that all agents are logged out and to identify any agents who are logged in. You can list all logged in agents for a split or skill, or limit the list to a range of login IDs or physical extensions:

- list members hunt-group <hunt group nnn>

- list members hunt-group <loginid nnnn to-loginid nnnn>
- list members hunt-group <ext nnnn to-ext nnnn>

For example, to list the agents logged in to skill 37:

1. Type **`list members hunt-group 37`** and press Return.

   The List Hunt Group Members report screen appears.

```
list members hunt-group 37

                          HUNT GROUP MEMBERS

Group Number: 37     Group Name: Platinum Card    Group Extension: 3002
  Group Type: ucd-mia       ACD? y     Skill? y           Members: 4

    Phys  Phys            Login Login          Agt     Per       Wrk
    Ext   Name            Ext   Name           Prf Lvl All SO DF Tim Occ
 1: 1002  1002-Al MacInni 2902  Agent 2902     grt 04      y      10  33
 2: 1022  1022-Kelly Chas 2901  Agent 2901     lvl 14      n      15  55
 3: 1001  1001-Chris Pron 2904  Agent 2904     pal R2         n    0   0
 4: 1021  1021-Maria Esta 2903  Agent 2903     pal 08   30    y   18  45
```

Note:
This screen shows a system using EAS and Avaya Business Advocate. For systems without either of these features, the related columns will be blank.

You can also use this command to list the agents *administered* in non-ACD hunt groups. However, since non-ACD hunt groups don't use agent logins the report will not identify agents who are currently active.

Upgrading to the EAS environment

For information on converting a call center to EAS, refer to Appendix P: Converting a call center to EAS on page 843.

Service Level Maximizer

Service Level Maximizer (SLM) is an optional Avaya Communication Manager Call Routing feature introduced in Release 2.0 that is used with Expert Agent Selection (EAS), and without Avaya Business Advocate.

SLM ensures that a defined service level of *X*% of calls are answered in *Y* seconds. When SLM is active, the software verifies that inbound calls are matched with agents in a way that makes sure that the administered service level is met.

This section includes the following topics:

- SLM requirements on page 503
- SLM operations on page 504
- SLM administration on page 510
- SLM algorithms on page 511
- SLM reporting on page 512
- SLM feature interactions on page 513

SLM requirements

SLM works on all platforms and operating systems that are supported by an Avaya communication server. SLM has the following licensing and system requirements:

- The Call Center Elite package.
- The Call Center Release field on the System-Parameters Customer-Options screen must be set to 12.0 or later.
- To obtain CMS reports that include information related to SLM, you must use CMS Release 12 or later. For more information about how to use CMS reports to evaluate SLM operations, see SLM reporting on page 512.
- SLM and Avaya Business Advocate cannot be simultaneously enabled on the System-Parameters Customer-Options screen. Therefore, SLM and Advocate can not both be used on the same system. Avaya Business Advocate provides a more flexible and functional form of achieving service level targets.

SLM operations

This section provides an overview of the SLM feature, and includes the following topics:

- SLM agent selection on page 504
- SLM call selection on page 504
- SLM target service levels and agent opportunity costs on page 505
- SLM benefits on page 507
- Auto reserve agents on page 507
- Agent selection rules in mixed skill environments on page 509

SLM agent selection

Agent selection methods are used when a call is queued to a skill and two or more agents are available to take the call. This is known as an agent surplus condition.

The algorithm for agent selection:

- Selects the agent with only one skill first.
- For agents with more than one skill, the agent's skills that are *not* needed for the incoming call are reviewed and the agent with skills that are less likely to be needed for future calls is selected. This is another way of saying that the algorithm selects the agent with the lowest opportunity cost.
- In a tie, the most idle agent is selected.

For details, see SLM target service levels and agent opportunity costs on page 505.

The agent selection algorithm does not consider the agent's skill level, or agent occupancy.

SLM call selection

Call selection applies when an agent becomes available and there are calls waiting in queue in two or more of the agent's skills. This is known as a call surplus condition.

Call selection compares all of the agent's assigned skills that have a call in queue and picks the one with the lowest ART value. If any ART values are negative, the call in the skill with the highest negative ART value is chosen.

For more information about ART values, see SLM target service levels and agent opportunity costs on page 505.

Note:
The SLM call selection method is applied to agents having at least one skill administered as **slm**.

The call selection algorithm does not consider an agent's skill level or call priorities.

SLM target service levels and agent opportunity costs

The SLM agent selection method is based on user-defined *target service levels* for SLM-administered skills and the concept of *agent opportunity costs*.

Target service level: You define specific target service level goals for each SLM skill based on the following format:

```
SLM target service level = x percent calls answered in y seconds
```

For purposes of SLM reporting, estimates of service level compliance for a skill are expressed as the Actual service level Relative to the Target service level (ART). At any point in time, an SLM skill can be below, equal to, or above its specified target service level. For example, if a skill has a target service level of 80% of all calls answered within 20 seconds and the current service level is 75% of all calls to the skill answered within 20 seconds, then the current ART value is -5%. Alternately, if the current service level indicates that 90% of all calls are being answered within 20 seconds, then the current ART value is +10%.

For information about how to administer service target levels for a skill, see SLM administration on page 510 and for information about evaluation of skill service level data, see Evaluating target service level compliance on page 512.

Opportunity costs: SLM compares actual call service levels to target service levels for each SLM skill, so that when an incoming call arrives at a skill, service level data can be used as the basis to develop agent *opportunity cost* estimates. The opportunity cost for an agent at a given point in time is represented as a weighted estimate that considers the status of the agents skills relative to the target service levels of each skill.

The process that SLM uses to derive agent opportunity cost estimates can be summarized as follows:

- An incoming call arrives for an SLM skill and agents that are both assigned to that skill and currently available are identified.
- All skills to which the available agents are assigned are also identified. For each of the assigned skills (excluding the skill associated with the incoming call), a current service level estimate is calculated and compared to the target service level.

Note:
The opportunity costs for a single-skill agent is always equal to zero, since they can always be selected for an incoming call in their assigned skill with no impact on the service level status of any other skills.

- Based on the current overall service level for the skills of each available agent, SLM derives a weighted estimate that identifies which of the available agents is currently the least needed for their other assigned skills, where the *need* of a skill is (approximately) defined as the difference between the current service level and the target service level. This agent has the lowest overall opportunity cost.

Because of the way that SLM estimates agent opportunity costs in the agent selection process, available agents whose skills are currently closest to matching their specified target service levels are selected first, while agents whose skills are furthest from matching their specified target service level are selected last. This strategy maximizes the possibility that an agent will be available when a call arrives at a skill whose target service level is at risk.

For example, consider a simplified scenario in which agents A and B, are both assigned to Skill 4 as well as two other skills. When an incoming call arrives at Skill 4 and both agents are available, SLM compares the current service level to the target service level for each of the skills to which the agents are assigned. The agent who currently has the lowest opportunity cost is identified and selected to receive the incoming call in Skill 4.

The following table shows how the agent with the lowest opportunity costs is selected in two different call service level scenarios:

Note:
To simplify this example, the service level states for each skill are represented as ART values. The actual agent selection algorithms used by SLM are complex and do not rely directly on ART data.

Skill Assignments	SLM Skill 1	SLM Skill 2	SLM Skill 3	Skill 4[1] (Incoming Call)
Agent A	X	X		X
Agent B		X	X	X
	For Skill 1, if...	For Skill 2, if...	For Skill 3, if...	Then... Agent with lowest opportunity cost for incoming call is:
Scenario 1	ART[2] = -5%	ART = +2%	ART = +2%	Agent B
Scenario 2	ART = -1%	ART = +5%	ART= -6%	Agent A

1. SLM agent opportunity cost estimates do not include service level data of the receiving skill for the incoming call.
2. ART = Actual service level relative to Target service level, where the service level is defined as x% calls answered in y seconds. For more information, see Evaluating target service level compliance on page 512.

In scenario 1 in this table, Agent B has the lowest opportunity cost compared to Agent A because the skills other than skill 4 assigned to Agent B (skills 2 and 3) are both above target service level. At the same time, of Agent A's skills (skill 1 and skill 2), skill 1 is below target. Agent A is selected for skill 1. Therefore, of Agents A and B, it is better to select Agent B for the incoming call to handle skill 4.

In scenario 2, of Agent B's other skills (2 and 3) skill 2 is above target level but skill 3 is below target by 6%. At the same time, of Agent A's other skills (1 and 2), skill 1 is only below target by 1%. Therefore, in this scenario, Agent A has the lowest opportunity cost compared to Agent B, since Agent B has a skill in worse shape than Agent A.

SLM benefits

Because SLM is able to differentiate skills in terms of their current call service demands, it provides the following advantages over other agent selection methods:

- Since agent resource needs for each skill are assessed in real-time, you can use SLM to allocate agent resources to those skills that have the greatest call service demand in a dynamic manner, thereby reducing overall call response times.
- Potential problems associated with staffing exceptions, or fluctuating, intra-day call service demands are also reduced.
- SLM is especially useful for call center operations that are bound by contract or other legal obligation to meet specific service level requirements.

Auto reserve agents

This section includes the following topics:

- About auto reserve agents on page 508
- How auto reserve works on page 508
- Considerations for allocating auto reserve agents on page 508
- Rules for auto reserve agents on page 509

For information about administration of the auto reserve option, see SLM administration on page 510. For information about how to use CMS reports to evaluate auto reserve operations, see Evaluating auto reserve rates on page 513.

About auto reserve agents

Auto-reserve is an added feature you can use to ensure that the service level is met in critical skills. When a critical skill is not meeting its service level, auto-reserve puts agents in standby for their other skills to ensure there is an available agent when the next call arrives for the critical skill. When an agent becomes available, all of his or her assigned skills are checked to see if any auto-reserve skills are not meeting their target service level. If so, the agent is made available only in those skills.

How auto reserve works

SLM also allows you to specify auto reserve agents for a skill to ensure that the desired service level is met in critical skills. When an agent becomes available, the agent can be reserved for SLM skills that have a weighted service level below their assigned targets. When the agent is reserved in one or more of his or her assigned skills, that agent is made available to receive calls only from those skills.

With SLM, an agent becomes reserved for an SLM skill, which has a Group Type of *slm*, when the agent becomes available. At that time, the SLM software checks all the agent's assigned skills to determine if any have a weighted service level below the target service level. Before the agent is automatically reserved for one or more of those skills:

- The skills must have a maximum auto-reserve setting of greater than 0 - as set on the Hunt Group screen for the skill
- The limit of reserved agents has not been exceeded for the skill

In other words, the agent is only available in that skill. The agent is made unavailable in the above-target skills, thus reserving each agent for the neediest skills.

Considerations for allocating auto reserve agents

Since auto reserve agents are kept unavailable in other skills, auto reserve agents should only be used in skills for which achievement of service level targets is considered to be critical. The addition of even a single auto reserve agent to a skill can have a significant impact on the service level that is realized. Therefore, Avaya recommends that you initially set the number of auto reserve agents on the Hunt Group screen to **0** or **1**, observe the impact on the service level, and if necessary, gradually increase the number of auto reserve agents by increments of one at a time until you have determined that your service level goals are reliably achieved.

Rules for auto reserve agents

For agents that are assigned to any skills that use the auto reserve option, the following rules apply when an agent becomes available:

- If any of the auto reserve-enabled skills to which an agent is assigned are currently below their specified target service level, the agent is available only in those skills.
- The designation of auto reserve agents for a skill is continuously assessed as agents become available. If the maximum number of auto reserve agents has already been reached, a single-skill agent who becomes available replaces the multi-skilled agent who has the highest opportunity cost.
- If one or more of an agent's auto reserve-enabled skills are currently below his specified target service level, a *multi-skill* agent is put into the auto reserve state if one of the following conditions are met:
  - The maximum number of auto reserve agents for the skill is not yet filled.
  - The maximum number of auto reserve agents for the skill *is* filled, but the opportunity cost for an idle, multi-skilled agent is lower than the opportunity cost of a multi-skilled agent who is currently in the auto reserve state. In this case, the agent with the highest opportunity cost is released from the auto reserve state.

Agent selection rules in mixed skill environments

SLM skills can be co-resident on the system with skills that use other agent selection methods, such as LOA. However, situations may arise in which a skill is not administered as an SLM skill, but includes agents that are also assigned to one or more SLM skills. In such a mixed skill environment, the following rules apply:

- If a non-SLM administered skill does not include any agents who are also assigned to SLM skills, then agent selection is based on the agent selection method that is administered for that skill.
- If a non-SLM administered skill includes one or more agents who are also assigned to SLM skills, a current service level value of 100% is applied to the non-SLM skill for purposes of SLM service level and agent opportunity cost calculations.

Important:
In a mixed skill environment, the service level for non-SLM hunt groups should be administered so that it reflects the importance of the hunt group to your business. For example, if it is permissible for inbound callers to wait for longer amounts of time, you might set the service level to be 75% (of calls answered) in 180 seconds. In other cases, when an extended wait time is not expected, but target service level compliance is not critical, you might set the service level to be 45% in 15 seconds.

SLM administration

Administration of the SLM feature is relatively simple. This section lists the administration screens and settings that are required for SLM administration.

SLM administration - verify that SLM feature is enabled[1]

Administration command:	`display system-parameters customer options`	
Page name:	Call Center Optional Features	
Required fields:	Call Center Release	12[2]
	Business Advocate?	n[3]
	Service Level Maximizer?	y
	Expert Agent Selection?	y

1. Contact your Avaya account representative if this screen indicates that any of the required feature selections are not enabled.
2. Call center release 12 or later.
3. SLM *cannot* be activated with Avaya Business Advocate.

SLM administration - set skill Group Type, Service Level Target and Auto Reserve Agents

Administration command:	`change hunt-group`	
Page name:	`Hunt Group`	
Required fields:	Group Type:	slm
	Service Level Target (% in sec):	_ in _ [1]
	Maximum Auto Reserve Agents:	___[2]

1. Default service level target values are set to 80% in 20 seconds.
2. The default value for this field is set to 0. Valid entries range from 0 to 9.

SLM algorithms

Starting with Avaya Communication Manager Release 3.1.2 (load 632), you can choose an alternative algorithm for selecting agents and delivering calls to maximize service level targets. The original Weighted Service Level (WSL) algorithm used for maintaining the service level targets has been changed to an Actual Service Level (ASL) algorithm that works better with low staff or low traffic level conditions. The ASL algorithm also handles the higher staff and traffic conditions and is recommended. ASL is determined as a percentage on a hunt group basis using the number of *accepted* calls in the current interval divided by the total calls in the current interval. A call is counted as *accepted* if it is answered within the target service level time period. You can still select the WSL algorithm on a system basis when desired. The WSL algorithm is based on a weighting calculation that uses the difference between the target time and the estimated wait time.

Criteria for choosing algorithms

The ASL algorithm is an improved algorithm for maintaining service level targets. Use ASL for most situations, in addition to low staff levels or low traffic level conditions unless actual experience indicates that WSL provides better performance for your installation.

Administering the ASL algorithm

To use the ASL algorithm:

1. From the System Administration Terminal (SAT), enter **`display system-parameters customer options`**.
2. On the System Parameters Customer-Options screen, make sure the **Service Level Maximizer** field is set to **y**.
3. Enter **`change system-parameters features`**.
4. On the Feature-Related System Parameters screen, make sure the **Service Level Maximizer Algorithm** field is set to **actual**.
5. Enter **`change hunt-group`**.
6. On the Hunt Group screen, set the **SLM Count Abandoned Calls** field to **n** or **y** (the default), depending on whether or not you want to include abandoned calls in the ASL algorithm calculations for SLM.
7. On the Hunt Group screen, set the **Service Level Interval** field to the time interval when you want ASL calculations to run. The default value is **daily**. The other choices are **hourly** or **weekly**.

SLM reporting

This section provides an overview of new Avaya CMS Supervisor report features that allow you to evaluate various aspects of SLM performance.

For detailed information about:

- CMS database items that are related to SLM or MAO, see *Avaya CMS Database Items and Calculations*
- ART reports, see the Avaya CMS Supervisor online help

This section includes the following topics:

- Evaluating target service level compliance on page 512
- Evaluating auto reserve rates on page 513

Evaluating target service level compliance

CMS includes database items that you can use in CMS Supervisor to evaluate how well your target service levels are met by SLM operations.

Note:
The service level used by the communication server to route calls is based on a prediction of a call being answered in the target service level. The service level calculated by CMS is the actual service level being achieved.

ART reports: Supervisor provides several types of Actual Relative to Target (ART) reports that compares actual service levels to target service levels and expresses the difference on a percent basis in a graphical format.

Note:
If your service level targets are based on contractual agreements, verify that your assessment of service level performance is based on a time frame (days, weeks, months) that is appropriate for the terms of your contract.

A percent value that exceeds zero means that actual service levels exceed the target, while percent values less than zero mean that the service level is not being achieved. When actual and target service levels correspond closely, the percent difference between the two data sets that are displayed in ART reports will tend to be close to zero, which is an indication that staffing levels are consistent with call service goals.

Service level calculations: Service level calculations can also be used to evaluate service level compliance. In R12 new database items have been added to track the number of calls answered (TARGETACDCALLS), abandoned (TARGETABNS) and outflowed (TARGETOUTFLOWS) within the service level administered on the communication server.

CMS uses the target service level that is administered on the communication server to generate these items. The advantage to using these items is that if the target service level is changed, CMS receives the new service level value and automatically adjusts how these items are computed. These items can be included in custom reports.

Note:
The existing CMS service level calculation can be used only if the acceptable service level on CMS Split/Skill Call Profile matches the Target Service Level administered on the communication server. If the target service level is modified on the communication server, the CMS service level must be manually modified to match that value.

Evaluating auto reserve rates

Avaya CMS Supervisor includes a %Skills Available column in historical Agent Summary Reports. The %Skills Available value is 100% when an agent spends no time in the auto reserve state. All values less than 100% indicate agent time spent in the auto reserve state.

SLM feature interactions

Before you use SLM, you should understand the feature interactions described below.

Avaya Business Advocate : SLM and Avaya Business Advocate cannot both be enabled on the System-Parameters Customer-Options screen.

BCMS Reporting Desktop VuStats: If BCMS Reporting Desktop VuStats is used to display acceptable service level report data, the displayed value is identical to the **seconds** value that is set in the Target Service Level (% in sec) field on the Hunt Group screen.

For more information about administration of SLM skills, see SLM administration on page 510.

Best Service Routing : With BSR, the best resource choice (among the local skills and best skills of the remote sites) is based on the lowest adjusted EWT or assigned available agent strategy rule. This rule does *not* consider service level targets that may be assigned to individual skills. However, when an SLM skill is selected as the best resource, the available agent selection *is* based on the specified service level target for the skill. Therefore, service level objectives are maintained within the local or remote skills but not across sites.

Direct agent calls: For agents assigned to SLM skills and eligible to receive direct agent calls, direct agent calls have priority over ACD calls.

Least Occupied Agent: SLM does not use LOA as an agent selection method.

Location Preference Distribution: You can assign reserve agents using SLM. In most cases, the selection of an agent or a call based on Location Preference Distribution takes precedence over SLM. However, SLM takes precedence when a reserve agent is needed because the service level is below the threshold.

Note:
If more than one reserve agent is eligible for the call, Location Preference Distribution is used to choose the agent.

Non-SLM Skills: Agents that have at least one assigned SLM skill will have their administered call handling preference (CHP) ignored and will be treated as if their call handling preference is set to **slm**. The non-SLM skills will be treated as if they are always at service level when it comes to agent and call selection. For more information, see Agent selection rules in mixed skill environments on page 509.

Greatest Need: Greatest Need is not used when SLM is enabled, since call selection is driven by the target call service levels that are administered for each SLM skill.

RONA: Redirected calls are considered in the service level calculations of any SLM skill to which they are sent.

Maximum Agent Occupancy (MAO)

This section includes the following topics:

- Overview on page 515
- When to use MAO on page 516
- MAO administration on page 516
- Determining when an agent is pending availability due to MAO on page 517
- Manual-in mode on page 517
- Auto-in mode on page 517
- Manual override of MAO aux mode on page 518
- Default AUX work reason code for MAO pending state on page 518
- Evaluating MAO using CMS reports on page 518
- Evaluating MAO using CMS reports on page 518

Overview

The Least Occupied Agent (LOA) and Most Idle Agent (MIA) methods attempt to maintain equitable agent occupancy rates based on time spent in call service. In contrast, SLM operations are driven solely by the needs of a skill in terms of meeting a specified target service level, and overall occupancy rates for individual agents are not a factor in the agent selection process. Instead, a Maximum Agent Occupancy (MAO) threshold can be used to achieve equitable agent occupancies and avoid agent burnout issues.

Note:
MAO can be used even when SLM is not active on the system, but MAO *must* be used with an EAS system or an EAS system using Business Advocate.

The MAO threshold is a system-administered option with a system-assigned maximum occupancy percentage value that is applied across all administered agents and is based on the total percentage of agent time in servicing calls. MAO data is derived from the same calculations that are used to derive the Least Occupied Agent (LOA).

When an agent who exceeds the specified MAO threshold attempts to become available, he or she is automatically placed in the AUX work mode for the reason code administered for this purpose. When the occupancy for such *pending* agents drops below the MAO, they are released from AUX work mode and made available.

When to use MAO

MAO is designed to provide short work breaks for agents who have high occupancy rates and is recommended only for call centers that use SLM, Avaya Business Advocate, or otherwise have some agents with high occupancy rates. High occupancy agents tend to be those agents with the highest skill level, or single-skill SLM agents.

MAO is not intended for call centers whose agents are administered in a highly similar manner. The MAO threshold should be set to a value that is sufficiently high to avoid situations where large numbers of agents are simultaneously put into the Auxiliary work mode.

MAO administration

This section lists the administration screens and settings that are required for MAO administration.

MAO administration for setting Maximum Agent Occupancy

Administration command:	`change system-parameters features`	
Page name:	`Hunt Group`	
Required field(s):	Group Type:	EAS skill
	Maximum Agent Occupancy Percentage	___[1]
	Maximum Agent Occupancy Aux Reason Code	___[2]

1. The default value for this field is set to 100%.

2. Aux work reason code 9 is set as the default. A different reason code can be used for this purpose, but Avaya recommends that you do *not* use reason code 0.

Determining when an agent is pending availability due to MAO

When an agent is put in to the Aux Work mode while pending availability because the agent's occupancy is above the system assigned limit, the agent's Aux Work button is lit indicating the agent is in Aux Work. This Aux Work condition is reported to the CMS adjunct reporting system using the system assigned reason code for MAO. If the agent has more than one Aux Work button assigned on their station set because the Aux buttons are associated with specific reason codes, the button that has the MAO reason code assigned will be lit. If none of the buttons have that reason code assigned, the Aux Work button that doesn't have an assigned reason code will be lit.

Manual-in mode

For agents in manual-in mode, when the agent exceeds the maximum-administered occupancy threshold, the agent is first put into the after-call work mode after the current call drops.

The agent is then put into the auxiliary work mode for the administered MAO reason code if the agent attempts to become available again by pressing the Manual-In button or dialing the FAC while occupancy is still above the maximum. When the occupancy for the agent drops below the administered maximum, the agent is put back into manual-in mode.

Auto-in mode

For agents in auto-in mode, when a call drops, the communication server automatically attempts to make the agent available again. If the occupancy for this agent has exceeded the maximum-administered level, the agent is put into auxiliary work mode for the MAO reason code instead of auto-in until the agent occupancy level has dropped below the administered maximum.

Manual override of MAO aux mode

If an agent wants to manually override the AUX code and leave the pending available state to make or receive an ACD call, they must do either of the following:

- Press the auto-in/manual-in button twice
- Enter the FAC code twice

Default AUX work reason code for MAO pending state

On the System-Parameters Feature screen, the default value in the **Maximum Agent Occupancy AUX Reason Code** field is set to 9. If reason code 9 is already being used to track other AUX time activities, agent time spent in the MAO pending state will be combined with time spent in those other activities. For this reason, you should designate an AUX work reason code solely for time spent in the MAO pending state, if it is possible to do so. For information about MAO administration, see MAO administration on page 516.

Important:
Avaya recommends that you do *not* use reason code 0 to track MAO Aux time.

Evaluating MAO using CMS reports

Avaya CMS includes database items that you can use to verify that MAO is functioning properly. In CMS Supervisor R12 or later, the historical Agent Summary report includes two fields that can be used to evaluate MAO performance. Depending on how you treat after call work in agent occupancy calculations, inspect one of the following fields:

- % Agent Occup w/ACW
- % Agent Occup w/o ACW

If the agent occupancy percentage is less then or equal to the specified MAO percentage, then this is one possible indication that MAO is functioning properly. You can also review agent AUX Work time for the reason code that you assigned for time spent in the MAO pending AUX Work state.

It is possible for an agent to have an occupancy that exceeds the MAO threshold if a recent call caused the agent to exceed the MAO threshold and insufficient time has elapsed for the agent occupancy to adjust.

MAO feature interactions

- Skills that are administered as auto-available will ignore the maximum occupancy system parameter. In other words, MAO does not apply.
- If the messaging system and the VRU skill are not assigned as Auto-Available Skills (AAS), the ports enter the pending availability Aux Work mode.
- An agent pending availability because of Maximum Occupancy is put at the bottom of the idle queue when he becomes available, if the hunt group is MIA.
- If the occupancy of an activated reserve agent is above the maximum, his availability is pending until his occupancy drops below the maximum with Avaya Business Advocate.
- When an agent that has been pending availability because of maximum occupancy becomes available, he is auto-reserved if he meets the auto-reserve criteria.
- Maximum Occupancy uses the same options and implementation as Least Occupied Agent (LOA) for determining an agent's occupancy. The occupancy includes the agent's time with ACD calls ringing, calls active, calls on hold at their voice terminal, and logged in After Call Work if the system option or specific agent login ID option considers ACW as agent work time. In other words, the option is set to y.
- (ACW) if the system-wide decision is made to consider ACW as agent work time.
- Agents that have been pending availability because of MAO are placed into the idle queue based on his occupancy, if the hunt group is LOA.
- When an agent becomes available or leaves ACW from a direct agent call, the agent’s availability is pending if his occupancy is above the maximum.
- An agent does not receive new Multiple Call Handling calls if his occupancy is above the system administered maximum. An agent’s availability is pending when he has dropped all active ACD calls.
- If an agent manually enters AUX with the maximum occupancy reason code, he is not treated as an agent with pending availability. The agent is treated as an agent in AUX Work and needs to manually change work mode.
- After the Timed After Call Work timer has expired, an agent’s availability is pending if his occupancy is above the maximum. While availability is pending, the agent is placed in AUX Work instead of Auto-In until the agent occupancy drops below the maximum.
- An agent whose occupancy is above the system maximum is forced to enter Forced Stroke Counts or Forced Call Work Codes before pending availability. After the Stroke Count or CWC has been entered, if the agent attempts to go to an available mode, he will be put into AUX Work mode for MAO if his occupancy has exceeded the maximum.

Call Vectoring commands

This section provides information about the commands used in Call Vectoring and includes the following topics:

- About Communication Manager contact center packages on page 522
- Communication Manager options required to enable vector commands on page 522
- Vector command description on page 526
- # command on page 527
- adjunct routing link command on page 530
- announcement command on page 538
- busy command on page 545
- check command on page 548
- collect digits command on page 556
- consider command on page 561
- converse-on command on page 566
- disconnect command on page 579
- goto step and goto vector commands on page 583
- messaging command on page 600
- queue-to command on page 606
- reply-best on page 614
- return command on page 617
- route-to command on page 619
- set command on page 632
- stop command on page 642
- wait-time command on page 645

About Communication Manager contact center packages

Some Call Vectoring commands require various software to be enabled. The features required to enable vector commands are included in the following Avaya Communication Manager Contact Center packages:

- Avaya Contact Center Introductory for centers with under 50 agents
- Avaya Contact Center Elite with Expert Agent Selection (EAS)
- Avaya Contact Center Elite with Business Advocate

Note:
Avaya Contact Center Deluxe was available for software releases prior to Avaya Communication Manager 2.0. Avaya Contact Center Introductory is identical to Avaya Contact Center Deluxe except it does not include Best Service Routing. Avaya Contact Center Introductory does include BCMS.

Most of the features required to fully enable vector commands are included in the Avaya Contact Center Deluxe or Introductory packages. To use skill options associated with some vector commands, the Avaya Expert Agent Selection (EAS) feature must be enabled. The EAS feature is included in the Avaya Contact Center Elite package. When a vector command requires the EAS feature, the requirement is noted.

In addition, other vector commands require Virtual Routing, which activates Look-Ahead Interflow. Other commands are available with non-contact center right-to-use (RTU) offerings, such as Auto Attendant, which activates Prompting.

Communication Manager options required to enable vector commands

The following table lists the options that are required to enable various vector commands, options, and parameters.

Command	Basic	Prompting	Attendant	Other Options Required
adjunct routing link	x			ASAI
announcement	x	x		
busy	x			
check best	x			ACD; G3V4 Advanced Routing; Best Service Routing

Command	Basic	Prompting	Attendant	Other Options Required
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
collect digits		x		
collect ced/cdpd digits		x		Vectoring (CINFO)
consider location	x			ACD; G3V4 Advanced Routing; Best Service Routing; Look-Ahead Interflow[1]
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	
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			

Command	Basic	Prompting	Attendant	Other Options Required
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[1]
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[2]	x	x		
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

Command	Basic	Prompting	Attendant	Other Options Required
reply-best	x			ACD; G3V4 Advanced Routing; Best Service Routing; Look-Ahead Interflow[1]
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)[2]	x	x		
route-to number if digit with cov y (n)[2]		x		
route-to number if unconditionally	x	x		
route-to number if interflow-qpos	x			ACD, Look-Ahead Interflow[1]
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	

1. Provided with Virtual Routing RTU (right to use).

2. If G3V4 software has not been purchased, these commands require the G3V4 maintenance load.

Vector command description

The following table provides a brief description of the function of each of the Communication Manager Call Vectoring commands. For a complete description of the command, see the listed page number.

Command	Function
# command on page 527	Adds comments to vectors.
adjunct routing link command on page 530	Requests an adjunct to route a call.
announcement command on page 538	Connects a caller to delay recording.
busy command on page 545	Connects a caller to busy tone.
check command on page 548	Connects or queues a call on a conditional basis.
collect digits command on page 556	Prompts a caller for digits.
consider command on page 561	Obtains BSR status data from a local split/skill or a remote location
converse-on command on page 566	Delivers a call to a converse split/skill and activates a Voice Response Unit (VRU).
disconnect command on page 579	Forces the disconnect of a call with an optional announcement.
goto step and goto vector commands on page 583	Causes an unconditional or a conditional branch to another step in the vector.
messaging command on page 600	Allows a caller to leave a message for callback.
queue-to command on page 606	Connects or queues a call to: • The primary split/skill • The attendant, attendant group, or hunt group with Attendant Vectoring • The best resource found by a consider series
reply-best on page 614	Sends BSR status data to the primary vector in a multi-site application.
return command on page 617	Returns vector processing to the step following the `goto` command after a subroutine call has processed.

Command	Function
route-to command on page 619	Connects a call to the destination entered using **collect digits** command, or connects a call to internal or external destination.
set command on page 632	Performs arithmetic and string operations and assigns values to a vector variable or to the Digits buffer during vector processing.
stop command on page 642	Stops further vector processing.
wait-time command on page 645	Initiates feedback to a caller if needed and delays processing of the next step.

command

This section includes the following topics:

- Purpose on page 527
- Syntax and valid entries on page 528
- Considerations on page 528
- Operation on page 529

Purpose

The **#** command adds comment steps to vectors. The step is skipped without being analyzed and vector processing continues at the next step. Administrators can include comments within vectors to make maintaining and troubleshooting vectors easier.

Note:
The **#** command functions differently than the comment out option of the **Esc f 6** vector editing function. See Entering a comment out indication to an existing vector step on page 225 and Removing a comment out indication on page 226 for details.

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 this 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 there are available vector steps.
- There are no limits on the number of # commands in each vector. However, the total number of non-blank steps allowed per system is limited as follows:
  - For S8300/S8400 systems: 1,280 steps in 256 vectors.
  - For S87xx/S8500 systems: 10,000 steps in 2,000 vectors.
- A single # command is counted as one step.
- Two or more consecutive # commands are counted as one step.
- # comment steps are not counted toward the 10,000 executed step limit or by the stepcnt vector variable.
- The comment out capability (adding a # after the step number) is supported by R14 or later CMS.
- The comment command (# followed by a command) is not supported by the CMS. Attempts by the CMS vector administration to access a vector with # will receive the unsupported step type error.

The following sample System Capacities screen shows the "used," "available" and "system limit" values for non-blank # commands.

Sample System Capacities screen

```
display capacity                                          Page   3 of  12

                                SYSTEM CAPACITY

                                                                  System
                                                 Used Available  Limit
                                                ----------------------

                  CALL COVERAGE
                          Coverage Answer Groups:      0    1000     1000
                                  Coverage Paths:      1    9998     9999
                              Call Pickup Groups:      0    5000     5000
                                    Call Records:      -       -    15424

          CALL VECTORING/CALL PROMPTING
                    Total Vector Directory Numbers:     31   19969    20000
                Meet-me Conference VDNs per system:      0    1800     1800
   Maximum Number of Expanded Meet-me Conf. Ports:      0      10       10
                          Total Vectors Per System:     22    1978     2000
            Meet-me Conference vectors per system:      0     999      999
        BSR Application-Location Pairs Per System:      0    2560     2560
                          Background BSR Poll VDNs:      0       5        5
                 Vector Comment Steps (non-blank):     41    9959    10000
```

Operation

You create # command vector steps by typing a **#** character in the command field of a blank vector step. You can enter up to 71 characters as the text parameter to the # command. Any combination of alpha-numeric visible ASCII characters, including blanks, is valid.

adjunct routing link command

This section includes the following topics:

- Purpose on page 530
- Syntax and valid entries on page 530
- Requirements on page 530
- The adjunct routing link process on page 531
- Feature interactions on page 534
- Avaya IQ interactions on page 535
- CMS interactions on page 535
- BCMS interactions on page 537

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[1] [*A-Z, AA-ZZ*] *V1-V9*

1. Link capacity depends on your release and configuration. For more information, see *System Capacities Table for Avaya Communication Manager on Avaya Media Servers*, , 555-245-601.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

The **adjunct routing link** command has the following requirements:

- Adjunct Switch Application Interface (ASAI) software must be installed.
- A MAPD or Application Enablement Services (AES) port is required, and the port must be connected to an ASAI host.

- The link number determined by a variable must be a valid assigned link number. If the value determined during call processing is not a valid, currently-assigned link number, the adjunct route step is skipped and a vector event is logged.

Note:
Do not unassign 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. The switch provides information in an ASAI route request message that the ASAI adjunct can use to first access a data base and then determine a route for the call. In a typical application, the ASAI adjunct might use the dialed number, the calling party number (CPN/BN), or the digits collected using Call Prompting or Caller Information Forwarding (CINFO) to access customer information and thereby determine 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 particular agent. An adjunct can also provide priority ringing, priority queuing, and specify that a route to an agent be done as a direct agent call.

When a call encounters an **adjunct routing link** command, the switch sends to the specified adjunct an ASAI message requesting a call route. The following list identifies the contents of the message, along with a comment or a brief explanation for each item:

- Calling number information. Calling party number or billing number (CPN/BN) provided by ISDN-PRI or R2-MFC signaling facilities. If the call originates from a local switch extension, this extension is the calling number.
- Originating line information (II-digits). Two-digit code provided by ISDN-PRI facilities indicating the type of originating line being used.
- Called number. Originally called extension (if a call is forwarded to a VDN), or the first VDN through which the call was routed (if the call was 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. ASAI identifier that permits the ASAI adjunct to track multiple calls using either Event Notification or Third Party Call Control. For more information on ASAI, see *Avaya Communication Manager CallVisor ASAI Technical Reference*.
- Look-Ahead Interflow (LAI) information (if any). Includes the original VDN display information, the priority level of the call at the originating switch, 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. These could be CINFO digits, but if so it will not be indicated by ASAI. For more information, see Call Prompting on page 237.
- User-to-User Information (if any). ASAI user-provided data associated with the call. If provided by ASAI, this data was provided in a 3rd-Party-Make-Call, Auto-Dial, or Route-Select message. If provided over ISDN, the data was in the SETUP message that delivered the call to this switch.

The **wait-time hearing i-silent** command is used in cases where it is important 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, the switch does not return an ISDN PROGress message to the originating switch. 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 at 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 subsequent treatment (that is, **announcement** and/or **wait-time**) steps (if any) in the vector are usually processed. In some cases, such processing is influenced by the position that the treatment steps occupy in the vector. In other cases, the positioning of these commands along with their relationship to specific **goto** commands come into play. For example, any **announcement** or **wait-time** step that immediately follows an **adjunct routing link** step whose ASAI link is 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, the switch executes the **goto** step and then 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, the switch skips any **announcement** or **wait-time** step that is the first non-**goto** step branched to by the **goto** step.

Note:
The first step to which a **goto** step is usually designed to branch (other than another **goto** step) is a non treatment step. That is, a step containing a command other than a **wait-time** or an **announcement** command).

Alternately, if the **goto** step fails and the branch is not taken, the switch skips any **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 the switch 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 the switch will wait for the ASAI adjunct to reply with a call route. Accordingly, you should always include either a **wait-time** step or an **announcement** step immediately after an **adjunct routing link** step. Moreover, the switch 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**

Note:
Multiple adjunct routing steps can follow each other in sequence. Each step activates a separate adjunct route request. Any intervening vector commands (or blank steps) between two **adjunct routing link** commands cancels any previous route-to requests.

If a valid call route is received by the server using a route-select message before one of the vector commands in the previous list is executed, the server routes the call to the destination specified by the adjunct route. Otherwise, the route request is terminated without affecting vector processing.

The adjunct can also decide to not route a call by rejecting (negatively acknowledging) 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 then continues with the next vector step.

When the server receives a call route (route-select to a destination) from the ASAI adjunct, the server first validates the route as follows:

1. The server verifies that the VDN's COR permits the call to be terminated at the adjunct-supplied destination.
2. The server verifies that the adjunct-supplied information (destination number, ACD split, TAC/AAR/ARS access code, etc.) for the route is valid. This includes checking that 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 option, the destination number (agent) must be logged into the adjunct-specified ACD split.
4. If the destination for the call is external, the server verifies the trunk is available for the call.

If any of these conditions are not met, the route validation fails, and the server does the following:

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 does the following:

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 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 the busy tone. Any other features that may be in effect at the adjunct-supplied destination (such as Send-All-Calls or Call Forwarding) interact with the routed call.

Note:
The operation described above is similar to that for the **route-to with coverage set to yes** commands.

Answer supervision considerations command

The command has no interaction with answer supervision.

If adjunct routing is used with ISDN-PRI, then an **adjunct routing link** command followed by a **wait-time hearing silence** signals the originating server that the receiving server has accepted the call (for Lookahead Interflow), even though answer supervision has not been provided. To prevent this from occurring, use the **wait-time hearing i-silent** option after the **adjunct routing link** step.

Feature interactions

For a call coming in directly to a VDN, the command is treated like a **route-to** command that has the **with cov** or **with coverage** parameter set to **y**.

Note:
If the Display VDN for Route-to DAC option 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 more information, see Displaying VDN names for vector-initiated DACs on page 677.

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 Call Forwarding 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 considered 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 without notification to the adjunct.

If a 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.

Avaya IQ interactions

The ability to report on information associated with use of the **adjunct routing link** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

Adjunct routing attempts are stored in the ADJATTEMPTS database item and reported as Adjunct Routing Attempts in 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 CMS. Accordingly, Adjunct Routing Attempts is not reported for this call.

When a routing response from the adjunct is successfully executed by the server, this action is tracked in the ADJROUTED and ADJROUTTIME database items and shown as Adjunct Routing Completions in standard reports.

Additional tracking of the **adjunct routing link** command varies based on the destination successfully routed to as follows.

Routed to station or to attendant		
Database item	Report Heading	Notes
OUTFLOWCALLS/ OUTFLOWTIME	Vector Flow Out	

Routed to station or to attendant		
INTIME	Avg 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	Avg Time In Vector	

Routed to VDN		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Vector Flow Out VDN Flow Out	
INTIME	Avg Time In Vector	
INFLOWCALLS	Vector Flow In VDN Flow In	new vector new VDN

Routed to split or to hunt group		
Database item	**Report heading**	**Notes**
CALLSOFFERRED		new split
LOWCALLS/MEDCALLS		no priority/priority

Split/skill calls are also shown in the standard reports based on 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.

A call abandoned after the command routes the call to a station or an attendant is tracked in the VDN tables as ABNCALLS/ABNTIME.

BCMS interactions

If the command advances a call to another position (that is, ASAI routing is successful), the call is tracked as outflow in the VDN Report.

announcement command

This section includes the following topics:

- Purpose on page 538
- Syntax and valid entries on page 538
- Requirements on page 538
- Operation on page 539
- Answer supervision considerations on page 543
- Feature interactions on page 543
- Avaya IQ interactions on page 544
- CMS/BCMS interactions on page 544

Purpose

Provides the caller with a recorded announcement.

Syntax and valid entries

announcement	*extension no.* *[A-Z, AA-ZZ]* *V1-V9*

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

Integrated board, aux trunk or analog (T&R or Lineside DS1) announcement equipment must be installed.

Appropriate announcements need to be administered and recorded. For more information, see *Feature Description and Implementation for Avaya Communication Manager*.

Operation

- Basic operation on page 539
- General considerations for announcements on page 539
- Delay announcements on page 540
- Forced announcements on page 541
- Information announcements on page 541
- Announcement recording tips for high traffic volume applications on page 541
- Recording announcements on page 542
- Considerations for DTMF Transfer and Connect applications on page 543

Basic operation

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

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.

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 *Feature Description and Implementation for Avaya Communication Manager*.

General considerations for announcements

You should understand the following considerations before you use the **announcement** command:

- After an announcement is provided, the audible feedback (such as music) should be re-attached.
- Depending on the type of announcement equipment and how the equipment is administered, callers may be required to listen to an entire announcement or they may be able to interrupt an announcement as it is playing. When a call is connected to an announcement, any previous treatment is discontinued.
- When an announcements must start from the beginning, the caller may have to wait in an announcement queue if the announcement is not ready to play. Callers hear the previously established call treatment (if any) 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/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's 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. For more information about the **wait-time** command, see wait-time command on page 645.

Forced announcements

When heavy call traffic is expected due to a major event, such as a widespread service problem that is currently being addressed, a call center may provide a forced announcement. Forced announcements are typically 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 fully addresses their needs 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.]
```

Announcement recording tips for high traffic volume applications

When setting announcement recordings for high traffic volume applications:

- Make sure 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 VAL boards have the highest capacity. These boards consist of 31 play ports and 60 minutes of storage using up to 256 announcement files.
- Make the recordings as short as possible. Long announcements delay further processing and hold up resources for a longer period of time.
- When recording announcements for collecting digits, play the longer portion 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 will 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 details, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .
- Use barge-in 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.
- See Considerations for DTMF Transfer and Connect applications on page 543.

Recording announcements

To make integrated announcement or music recordings that reside in VAL announcement boards or virtual VAL sources in media gateways, use a system telephone or create .wav files using a local PC or recorded at a professional recording studio.

For details on how to record announcements, see *Administrator Guide for Avaya Communication Manager*.

Recording announcements using .wav files: Using .wav files for recording provides the best quality and the most flexibility and reliability.

- Use a PC 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. Avaya recommends Voice Announcement Manager.
- Administer the .wav file name (less the .wav extension) to an announcement extension on the announcement/audio sources screen.

Recording announcements using a telephone: You can also 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 feature access code (FAC).
- 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. Using FTP, move each pre-recorded file to each of the sources defined for the audio group.

For more information, see *Administrator Guide for Avaya Communication Manager*.

- 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.

Considerations for DTMF Transfer and Connect applications

Consider the following when recording DTMF for 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 at least 80ms of digit duration and 80ms of inter-digit silence, and include a pause of at least 350ms between the *8 and the redirection number. The lower and upper bounds are 300ms - 11 seconds.
- Minimize or prevent the recording of any noise or periods of silence.

It is not possible to record DTMF tones using IP phones or to record H.248 Media Gateway VVAL announcement sources using any type of telephone. Instead, you can do any of the following options:

- 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 sources as required.
- Record the DTMF tones using a commercially-available PC 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).

Feature interactions

For LAI, the command may be considered a call acceptance vector command or a neutral vector command.

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

- Announcement is available.

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

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

Avaya IQ interactions

The ability to report on information associated with use of the **announcement** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS/BCMS interactions

The command is not tracked by CMS or BCMS.

busy command

This section includes the following topics:

- Purpose on page 545
- Syntax on page 545
- Requirements on page 545
- Operation on page 546
- Answer supervision considerations on page 546
- Feature interactions on page 546
- Avaya IQ interactions on page 547
- CMS interactions on page 547
- BCMS interactions on page 547

Purpose

The **busy** command gives the caller a busy signal and causes termination of vector processing.

Syntax

```
busy
```

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

No special requirements.

Operation

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

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

An example vector that demonstrates the **busy** command is shown below.

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

Answer supervision considerations

After the 45 second timeout, an unanswered CO trunk call is answered and then dropped. All other unanswered calls after this timeout are dropped without being answered. For 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.

Feature interactions

For LAI or BSR, the command is considered a call denial vector command in all cases.

Avaya IQ interactions

The ability to report on information associated with use of the **busy** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

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.

BCMS interactions

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

check command

This section includes the following topics:

- Purpose on page 548
- Syntax and valid entries on page 549
- For information about unexpected results, see Troubleshooting vectors on page 689. on page 549
- Operation on page 550
- Check split command on page 551
- Answer supervision considerations on page 553
- Feature interactions on page 553
- Avaya IQ interactions on page 554
- CMS interactions on page 554

Purpose

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

Syntax and valid entries

Check	**best**	**if**	**expected wait**		< 1-9999 seconds, > 0-9999 seconds		
			unconditionally				
			wait improved		< 1-9999 seconds, > 0-9999 seconds		
	skill	hunt group[1], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = low **m** = medium **h** = high **t** = top	**if**	**available-agents** > 0-1499[2]	**all-levels**
							pref-level skill level 1[3]
							pref-range skill level 1[3] **to** skill level 2[3]
	skill	hunt group[1], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = low **m** = medium **h** = high **t** = top	**if**	**calls-queued** < 1-999[2] **expected-wait** < 1-9999 seconds **oldest-call-wait** > 1-999 seconds **rolling-asa** < 1-999 seconds **staffed-agents** > 0-1499[2] **wait-improved** > 0-9999 seconds **unconditionally**	
	split	hunt group[1]					
	split	hunt group[1]	**pri**	**l** = low **m** = medium **h** = high **t** = top	**if**	**available-agents** > 0-1499[2]	

1. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
2. 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.
3. Skill levels are 1-16 (1 is best, 16 is lowest). Skill Level 2 must be greater than or equal to Skill Level 1.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

No special requirements.

Operation

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

Each **check** command may be used with one of the following three keywords: **split**, **skill**, or **best**. The **check split** or **check skill** command requires you to specify the split/skill to be checked. The **check best** command checks the status of the best split/skill identified by the immediately preceding series of **consider** steps, then either terminates or queues the call to that split/skill. You don't have to specify the split/skill in **check best** commands since the switch compares two or more skills and identifies the best in the preceding series of **consider** steps.

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

- Number of available agents is greater than the threshold value.
- Number of staffed agents is greater than the threshold value.
- Number of calls queued for a specified priority level or higher is less than the threshold value.
- Oldest call waiting in queue at the specified priority level or higher has been waiting less than the threshold value, which is expressed in seconds.
- Rolling average speed of answer is less than the threshold value, which is expressed in seconds.
- Expected wait time is less than the threshold value, which is expressed in seconds.
- Expected wait time will be improved by more than the threshold value, which is expressed in seconds, by queuing the call to the split/skill specified. EWT in the specified split/skill is compared to the call's current EWT. (A call's EWT will be infinite if the call is not in a queue.)

A call may be queued to up to three splits/skills simultaneously. A call remains queued either 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 any split/skill to which the call is queued, the following actions take place:

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

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

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

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

The check skill command can further have a skill level preference parameter. The skill level preference parameter can either be a skill level or a range of skill levels. This option allows you to request that calls are routed to an available agent with the specified skill level. Skill level preference is applied under the following conditions:

- There must be available agents with the specified skill (agent surplus); otherwise, the step is skipped and vector processing continues at the next step.
- The skill (hunt group) must be administered as EAD-MIA or EAD-LOA.
- If no agents are available with the requested skill level, then 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. For more information about how these commands work, see queue-to split command on page 608.

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 that 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**, **3rd**	**pri**	**l** = low **m** = med **h** = high **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 Level1 field, in which you can enter a skill value between 1 and 16. If you select pref-range, the system displays two fields, Skill Level1 and Skill Level2. Using these two fields, you can enter a range of preference levels, such as 5 (Skill Level1) to 13 (Skill Level2). The values in both these fields need to be between 1 and 16. The number in Skill Level2 field needs to be greater than or equal to the number you enter in the Skill Level1 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 Level1 field in which you can enter a skill level value for the agent from 1 to 16. Preference level of 1 is the best while 16 is the least.
pref-range	The system displays two fields, Skill Level1 and Skill Level2. Using these two fields, you can enter a range of preference levels, such as 5 (Skill Level1) to 13 (Skill Level2). The values in both these fields need to be between 1 and 16. The number in Skill Level2 field needs to be equal to or greater than the number you enter in the Skill Level1 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

No answer supervision is returned.

Feature interactions

The **check** command can access a messaging-system/message center/server split/skill in cases where a VDN is assigned as a coverage point. To enable this function, the split/skill must be assigned as a vector-controlled hunt group.

For BSR and LAI, the command can be considered either a call acceptance vector command or a neutral vector command. For more on BSR interactions, see Best Service Routing (BSR) on page 297.

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

- Call terminates to an agent.
- Call queues to a split/skill.
- BSR interflowed call is accepted at remote interflow vector.

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

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

The **oldest-call-waiting** condition can check only priority level l (low).

Avaya IQ interactions

The ability to report on information associated with use of the **check split/skill** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

Note:
CMS vector administration cannot be used to display or modify vectors that contain a 'check skill' command with the skill level preference option set to 'pref-level' or 'pref-range'. To display or modify these vectors, use Communication Manager vector administration commands instead.

CMS interactions

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 on CMS database items and reports, see *Avaya CMS Database Items and Calculations* and *Avaya CMS Supervisor Reports*.

BCMS interactions

The total number of calls to the VDN that are queued with the command and then answered by an agent within a specified time period is tracked as ACD Calls in the VDN Report. The average time that calls spend in a vector before being connected with the command as an ACD call 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 these calls as outflow in the VDN Report.

collect digits command

This section includes the following topics:

- Purpose on page 556
- Syntax and valid entries on page 556
- Requirements on page 556
- Operation on page 557
- Answer supervision considerations on page 561
- Feature interactions on page 561
- Avaya IQ interactions on page 561
- CMS/BCMS interactions on page 561

Purpose

The **collect digits** command allows the user to enter up to 16 digits from a touch-tone phone or an internal rotary phone, or allows the vector to retrieve Caller Information Forwarding (CINFO) digits from the network.

Syntax and valid entries

collect	**ced** **cdpd**	**for**	**none** *A-Z, AA-ZZ*		
	1-16	**digits after announcement**	*extension no.* **none** *A-Z, AA-ZZ* *V1-V9*	**for**	**none** *A-Z, AA-ZZ*

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

The Avaya Call Center Deluxe package or Avaya Call Center Elite package must be installed. This command is also available with the Automated Attendant RTU.

At least one TN744 Call Classifier circuit pack or TN2182 Tone Clock circuit pack must be in the system unless the command is used only to collect digits returned by a VRU or sent by the network and never to collect digits from a caller.

The Vectoring (CINFO) feature used to collect ced or cdpd digits from the network ISDN and the AT&T Network Intelligent Call Processing (ICP) service or equivalent.

Operation

The collect command has two modes of operation:

- Collecting digits on the switch
- Collecting CINFO digits

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

Note:
You can set the **Reverse Star/Pound Digit For Collect Step?** field on the Parameters page of the Feature-Related System Parameters screen to y in order to reverse the normal handling of the asterisk (*) and pound (#) digits by the **collect** vector command. With the Reverse Star/Pound Digit for Collect Step set to y, the asterisk (*) digit is interpreted as a caller end-of-dialing indicator and the pound (#) digit is interpreted to clear all digits that were previously entered for the current collect vector step.

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

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

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

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

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

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

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

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

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

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

The inter-digit timer used for the initial digit timeout (to detect rotary dial callers), between digit timeout and for short entry (with variable length digit collection) is set to 10 seconds by default. However, the timer can be set to a value between 4 and 10 seconds using the Prompting Timeout field on the System-Parameters Customer-Options screen.

If, during the digit collection phase, an asterisk symbol (*) is encountered within a stream of dialed or dial-ahead digits, all digits that are collected for the current **collect digits** step are discarded. If additional dial-ahead digits occur after the asterisk, these digits continue to be processed. If there are no such digits, and if no 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 touch-tone phone), the interdigit timer is restarted to detect a timeout for releasing the TTR. Vector processing then continues at the next vector step. However, the switch continues to collect any subsequent dialed digits (including the pound sign (#) and asterisk (*) digits) to allow for the dial-ahead capability. These additional dialed ahead digits are saved for use by subsequent **collect digits** commands, and they provide the caller with a means to bypass subsequent unwanted announcement prompts. A single # digit can be collected and tested by subsequent **route-to...if digits** or **goto...if digits** commands. Alternately, any collected digits (whether collected from callers or CINFO) can be passed to a host with ASAI or forwarded to another site with Information Forwarding. Collection of dial-ahead digits continues until one of the following occurs:

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

  Note:
  Any asterisk (*) or pound sign (#) digits count towards the 24-digit limit, as do any dial-ahead digits entered after the asterisk or pound sign digit.
- The TTR required by the touch-tone phone user to collect digits is disconnected. This occurs under the following conditions:
  - Successful or unsuccessful **route-to number** step is encountered during vector processing except where the number routed to is a VDN extension.
  - Successful or unsuccessful **route-to digits** step is encountered during vector processing except where the number routed to is a VDN extension.
  - Successful or unsuccessful **adjunct routing link** step is encountered during vector processing.
  - Successful or unsuccessful **converse-on** step is encountered during vector processing.
  - 10 second timeout occurs, during which time the caller does not dial any digits, asterisks (*) or pound signs (#).
  - A collect ced/cdpd digits step is processed.

Note:
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. This means that, following a failed **route-to**, **converse-on** or **adjunct routing link** step, a subsequent **collect digits** step always requires the caller to enter digits.

Note:
Dial-ahead digits are available for use only by subsequent **collect digits** commands. The digits are never used by other vector commands that operate on digits (for example, **route-to digits**, **goto...if digits**, etc.). In addition, these digits are not displayed as part of the CALLR-INFO button operation until they are collected with a **collect digits** command.

Collecting CINFO digits: The collect digits step allows you to collect CINFO Digits from the network. When a **collect ced digits** or **collect cdpd digits** step is processed, the system retrieves the first sixteen ced or cdpd digits from the ISDN User Entered CODE (UEC) Information Element that is associated with the call. It places the digits in the collected digits buffer. Any digits that were in the collected digits buffer when the ced or cdpd digits are collected, are erased. If a TTR was connected to the call from a previous **collect digits** step, it is disconnected.

If the ced or cdpd digits contain invalid digits (not 0-9, *, #) the digits are not placed in the collected digits buffer. However, the collected digits buffer is still cleared and if a TTR is attached it is disconnected.

If no ced or cdpd digits were 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 it is disconnected.

A * 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. If a single # is sent, it is placed in the collected digits buffer.

The number of ced or cdpd digits to collect cannot be specified in the **collect digits** step. If there are 16 or fewer digits, all the digits are collected. If there are more than 16 digits, 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.

Once ced or cdpd digits are collected, they can be displayed on a two-line display, or using the callr-info button.

Answer supervision considerations

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

Feature interactions

For BSR and LAI, the command is considered a call acceptance vector command except for collect ced/cdpd digits which is neutral.

Avaya IQ interactions

The ability to report on information associated with use of the **collect digits** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS/BCMS interactions

Collected digits are passed to the CMS when the **collect** step is processed. Digits are not passed to the BCMS.

consider command

This section includes the following topics:

- Purpose on page 562
- Syntax and valid entries on page 562
- For information about unexpected results, see Troubleshooting vectors on page 689. on page 562
- Operation on page 562
- Recommendations on page 564
- Answer supervision considerations on page 565

- Feature interactions on page 565
- Avaya IQ interactions on page 565
- CMS/BCMS interactions on page 565

Purpose

The **consider** command defines the resource (split, skill, or location) that is checked as part of a BSR consider series and obtains the data BSR uses to compare resources. After the consider series has been executed, a **queue-to best** or **check best** command can queue the call to the best resource identified.

If the **consider** commands are in a status poll vector, a **reply-best** step returns the data for the best resource found to the primary vector on the origin switch.

Syntax and valid entries

<table>
<tr><td rowspan="3">consider</td><td>location[1]
(multi-site BSR only)</td><td colspan="3">1-255
A-Z, AA-ZZ
V1-V9</td><td rowspan="3">adjust by</td><td rowspan="3">0-100 percent
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>skill</td><td>hunt group[2],
skills for VDN: 1st, 2nd, 3rd</td><td rowspan="2">pri</td><td rowspan="2">priorities:
l = low
m = medium
h = high
t = top</td></tr>
<tr><td>split</td><td>hunt group[2]</td></tr>
</table>

1. This item is available only with the Virtual Routing feature.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

For switch requirements, see Server requirements on page 301.

Operation

In order to deliver a call to the resource that can provide the best service, **consider** commands collect and compare information. Whether you use single-site BSR, multi-site BSR, or both, consider steps work very much the same.

Each **consider** command collects status data from one split/skill. Splits or skills on the same switch are identified by number. Remote locations must be identified by a location number assigned on the BSR Application screen. For more information, see Multi-site BSR applications on page 323.

Consider commands are typically written in a series of two or more steps called a *consider series*. The first step in a consider series collects status data from the resource (a split, skill, or location specified by the user in the command) and saves this data to a buffer. The next **consider** step collects status data on its assigned split/skill and compares the data to that already in the buffer. If the existing data in the buffer indicates the first split/skill can provide better service to the call, the data for the first split/skill remains in the buffer as the best data. If the second split/skill can provide better service to the call, its status data replaces the data already in the buffer. Each subsequent step works similarly, collecting data from one resource, comparing it 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** step delivers or queues the call, or when a **reply-best** step returns the data for the best resource to a primary vector on the origin switch.

The first consider step in a series shortens the call vectoring 15-step timeout from 1.0 to 0.2 seconds. The timeout is shortened for BSR vectors only (that is, vectors that use **consider** series) in order to reduce real-time delays for call processing and reduce the incidence of race conditions in multi-site BSR applications.

User adjustments

You may have preferences as to which skills should answer certain types of calls. In both single- and multi-site BSR, the **adjust-by** portion of the **consider** command allows you to program these preferences into your vectors.

If a resource does not have an available agent when its **consider** step tests it, the **consider** step collects the Expected Wait Time (EWT) were the call to be queued to that resource. You can adjust this EWT value, for purposes of calculation only, by assigning a value of 0-100 in the user adjustment. The units of this value are supplied by the switch depending on the conditions whenever that **consider** step executes.

For example, in the command **consider split 1 pri h adjust-by 20,** the switch interprets **adjust-by 20** to mean *add 20% to the EWT, but add at least 20 seconds*. For Expected Wait Times of 1-100 seconds, an adjustment of 20 will therefore add 20 seconds. Above 100 seconds, the same adjustment will add 20% to the EWT for the split/skill specified in the **consider** step.

Important:
If the user adjustment are defined as a number of seconds, BSR would 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 multi-site 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 EWT. For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) 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)

Recommendations

It is recommended that you follow the guidelines below when using **`consider`** commands:.

- Don’t put a consider series in vector loops.

- Don’t put any commands between the steps of a consider sequence that would cause a delay. The **announcement** and **wait** commands, for example, should not be used within a consider sequence. The **goto** commands are OK.
- Arrange your **consider** steps in order of preference.

  The **consider** step that tests the main, or preferred, resource should be the first in the series. The second **consider** step should test the resource that is your second preference for handling the given call type, and so on. To avoid unnecessary interflows, put **consider** steps for local resources before steps that consider remote resources. Arranging **consider** steps in order of preference is recommended for all BSR vectors. It’s especially important when the active VDN for the call is using the 1st-found agent strategy: since the switch will deliver the call to the first available agent found, arranging **consider** steps in order of preference will ensure that calls are delivered to the best of the available resources and that unnecessary interflows are avoided.

Answer supervision considerations

All forms of the **consider** command are ISDN neutral and do not return answer supervision.

Feature interactions

Splits used in **consider** commands must be vector-controlled.

Avaya IQ interactions

The ability to report on information associated with use of **consider** commands is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS/BCMS interactions

BCMS does not log LAI attempts. Therefore, it will not log BSR status polls since they are LAI attempts.

converse-on command

This section includes the following topics:

- Syntax and valid entries for the converse-on command on page 566
- For information about unexpected results, see Troubleshooting vectors on page 689. on page 566
- Operation on page 567
- converse-on split command on page 570
- Answer supervision considerations on page 572
- Feature interactions on page 573
- Avaya IQ interactions on page 577
- CMS interactions on page 577
- BCMS interactions on page 577

Syntax and valid entries for the converse-on command

converse-on	**skill**	*hunt group*[1], skills for VDN: **1st, 2nd, 3rd**	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op	**passing**	*6-digit string* ***** **#** **ani** **vdn** **digits qpos** **wait** *A-Z, AA-ZZ V1-V9*	**and**	*6-digit string* ***** **#** **ani** **vdn** **none digits qpos** **wait** *A-Z, AA-ZZ V1-V9*
	split	*hunt group*[1]				**none**	**and**	**none**

1. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements for the converse-on command

A converse split must be a vector-controlled ACD or non-ACD hunt group.

Note:
The converse-on command with a non-ACD hunt group is not supported if EAS is enabled.

Operation

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

For details regarding call flows, data passing, collection, and return specifications involving the **converse-on** command, see Appendix M: Call flow and specifications for converse - VRI calls on page 807.

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

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

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

Digits returned from the VRU can be:

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

The communication server 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 may serve two major purposes: the digits may notify the VRU of the application to be executed, and they may share call related data, such as ANI (BN) or caller digits collected by the communication server. In many applications, both application selection and data sharing are required. The touch tone outpulsing rate is adjustable. For details, see Appendix M: Call flow and specifications for converse - VRI calls on page 807.

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

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

Any data passed from the switch to a VRU is outpulsed in-band. The user can administer two time delays on the System Parameter Features screen: converse first data delay and converse second data delay fields. These delays may range 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 delays are needed to give the VRU time to invoke an application and to allocate a touch-tone receiver to receive the passed digits.

Note:
No time delays are invoked when the keyword none is administered.

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

If <data_2> is not none, the converse second data delay timer starts when the end-of-string character (#) from the first digit string is outpulsed. When the timer expires, the <data_2> digits are outpulsed in-band to the VRU. The end-of-string character (#) for the second digit string is then outpulsed.

Data 1 and Data 2 values administered within the converse-on command

The following values may be administered for <data_1> and <data_2> within the **converse-on** command:

- Administered digit string: This string can contain up to six characters consisting of one or more digits (0 through 9) or asterisks (*). The pound sign (#) may not be included in a digit string because it is reserved as the end-of-string character. However, a single # may be administered.
- ani: 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: This data type can be used only if Call Prompting is optioned. To pass CINFO digits, Vectoring (CINFO) must also be enabled. The digits data type causes the most recent set of digits collected in vector processing, either from the caller or from the network, to be outpulsed. If no digits are available, the end-of-string pound sign (#) is the only character outpulsed.

- none: This data type causes no characters to be outpulsed. Also, no end-of-string pound character (#) is outpulsed, and no time delays are invoked.
- qpos: This data type causes the value of the queue position of a call in a 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. This data may be used by the VRU to inform callers of their position in queue or to decide whether to execute a long or short version of a voice response script.

  Note:
  The use of this keyword is not recommended with multiple split/skill queuing. Any queue position value that is sent may not be meaningful. If the call is queued to multiple non converse splits/skills, the value of the caller's queue position in the first non converse split/skill is sent. Priority queuing (priority assigned to the queue vector step) and Dynamic Queue Position, which is available with Avaya Business Advocate, can put subsequent calls into the queue ahead of the waiting call.
- 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.
- wait: This data type can be used only if the Vectoring (G3V4 Advanced Routing) customer option is enabled. It causes the expected wait time of the call in seconds to be outpulsed. For a detailed description of expected wait time, see Expected Wait Time (EWT) on page 259. If the call is not queued or if it is queued only to splits that are unstaffed or splits where all agents are in AUX work mode, the end-of-string character # is the only character outpulsed. The value outpulsed is a variable number not padded with zeroes. It is a maximum of four digits always followed by #. The range is 0# to 9999# or a single #.
- A to Z, 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 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.
- #: This is the only character outpulsed. Outpulsing this character causes the corresponding **prompt** and **collect** command in the voice response script to be skipped.

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

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

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.

This section includes the following topics:

- VRI capabilities on page 570
- VRI benefits on page 570
- Other VRI considerations on page 571
- Using converse-on to outpulse caller information to VRUs on page 571

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 server.

VRI benefits

Use of VRUs with vector processing provides the following advantages:

- Access to local and host databases
- Validation of caller information
- Text to speech capabilities
- Speech recognition
- Increased recorded announcement capacity
- Audiotex applications

- Interactive Voice Response (IVR) applications
- Transaction processing applications

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

When Advanced Vector Routing is enabled, the expected wait time for a call can be passed to the VRU, and conveyed to the caller. For more information, see Expected Wait Time (EWT) on page 259.

Other VRI considerations

You should also understand the following considerations when you implement VRI:

- If callers need to hear an entire voice response script before speaking to an agent, the call should not be queued until after a **converse-on** step is executed.
- Audible feedback should be provided prior to a **converse-on step** whenever a large number of digits need 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 Call Prompting is enabled)
- Expected Wait Time (if Advanced Vector Routing is enabled)
- 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. For more information, see Variables in Vectors on page 107.

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 shown above, a vector can be used to respond to calls that originate from VDNs that serve customer needs (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 the switch becomes available to service the call, the line to the VRU is immediately dropped, and the calling party is connected to the available agent.

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, you may wish to include a vector step to test whether sufficient time is available for a voice response script to be executed. In the example shown above, step 4 includes a **calls-queued** condition that is used for this purpose.

It is also important to provide a feedback step prior to the converse-on step in case there is a delay in reaching an available converse split port. In the example shown above, step 5 provides music for this purpose.

For more information about the call flow for converse-VRI calls, see Appendix M: Call flow and specifications for converse - VRI calls on page 807.

Answer supervision considerations

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

Feature interactions

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

Adjunct Switch Applications Interface (ASAI**):** Since vector-controlled splits/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/skills.

If a **converse-on** step is executed while an adjunct routing request is outstanding, the route request is canceled.

Auto-Available Splits/Skills: A **converse-on** step may place a call to an auto-available split/skill. Except in cases where the converse split/skill is ASAI-controlled, auto-available converse splits/skills are recommended for Voice Response Integration (VRI).

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

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

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

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

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

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

Call Vectoring—Basic: The **converse-on** step is an enhancement to the Basic Call Vectoring customer option. This option must be enabled in order to invoke the VRI feature.

Class of Restriction (COR**):** As is the case for the **queue-to split/skill** and **check split/skill** vector steps, no COR checking is carried out when a **converse-on** step places a call to a split/skill.

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

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

Direct Department Calling (DDC**):** A converse split may be administered as a direct department calling split.

Distributed Communications System (DCS**):** If an incoming DCS call is placed to a vector with a **converse-on split/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 may be queued at one of four priority levels: low, medium, high or top.

Hunt Groups: The **converse-on** step may deliver a call to a vector-controlled hunt group, ACD split/skill, message center or a messaging-system hunt group.

Integrated Services Digital Network (ISDN**):** The **converse-on** step may be administered to outpulse to the VRU with the ANI (calling party number/billing number CPN/BN) of the calling party. The outpulse uses an ANI keyword.

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

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

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

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

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

Look-Ahead Interflow (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/skill on the receiving switch 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 may 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 may retrieve his or her messages in the usual manner.

If a call is forwarded to or covers to a VDN and is then delivered to 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 may leave a message for the principal.

Multiple Split/Skill Queuing: A call can be queued to three different splits/skills and then to a converse split/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/Skills: A **converse-on** step may not place a call to a non vector-controlled split/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 they are delivered to busy hunt groups. Call Vectoring audible feedback is not disconnected while a converse call is in queue.

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

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

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

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

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

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

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

System Access Terminal (SAT**):** **converse-on** steps may be administered 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 auto-available splits (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 the switch.

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

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

If an illegal attempt to transfer a converse call is made, a vector event is logged, the line to the 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 may 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 then 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**):** A converse split/skill may be administered as a Uniform Call Distribution split/skill.

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

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

VDN **Reports:** For call tracking in the CMS and BCMS VDN reports, a **converse-on** step is treated like an **announcement** step. A call is considered answered when it is answered by a non converse split/skill but never when it is answered by a converse split/skill.

Vector-controlled Splits/Skills: A **converse-on** step may place a call to a split/skill only if that split/skill is administered as a vector-controlled split/skill.

Avaya IQ interactions

The ability to report on information associated with use of the **converse-on** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

The CMS tracks calls placed by a **converse-on** step to a CMS-measured split/skill. Since a **converse-on** step allows a call to be answered in more than one split/skill, trunk totals no longer match split/skill totals. However, VDN totals and trunk totals will match.

For call tracking in the CMS VDN reports, a **converse-on** step is treated like an **announcement** step. A call is considered answered when it is answered by a non converse split/skill but never when it is answered by a converse split/skill.

BCMS interactions

BCMS tracks calls placed by a **converse-on** step to a BCMS-measured split/skill. Since a **converse-on** step allows a call to be answered in more than one split/skill, trunk totals no longer match split/skill totals. However, VDN totals and trunk totals will match.

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

disconnect command

This section includes the following topics:

- Purpose on page 579
- Syntax and valid entries on page 579
- For information about unexpected results, see Troubleshooting vectors on page 689. on page 579
- Operation on page 580
- Answer supervision considerations on page 580
- Feature interactions on page 581
- Avaya IQ interactions on page 581
- CMS interactions on page 581
- BCMS interactions on page 582

Purpose

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

Important:
You should always warn the caller prior to disconnecting the call.

Syntax and valid entries

disconnect	**after announcement**	*extension no.* **none** *A-Z, AA-ZZ* *V1-V9*

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

The relevant announcements must be administered and recorded.

Operation

The **disconnect** command forcibly disconnects a call with an optional announcement. Any previously established call treatment ends when the **disconnect** command is executed, and the call is removed from vector processing and from the switch.

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, the switch sends answer supervision (if it was not already sent) just before the announcement plays.

When the **disconnect** command does not include an announcement, the switch sends answer supervision before it disconnects a call.

Note:
Answer supervision is not sent for ISDN trunks.

An example of the **disconnect** command is shown below.

Call disconnect example

```
 disconnect after announcement 2918 [Today has been declared a snow    day.  Please
report for work tomorrow at 8 P.M.]
```

In this example, the caller is provided with sufficient information to meet their needs, so that no further interaction is required.

Answer supervision considerations

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

Feature interactions

For LAI, the command can be considered either a call acceptance vector command or a call denial vector command.

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

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

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

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

Avaya IQ interactions

The ability to report on information associated with use of the **`disconnect`** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

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

BCMS interactions

A call that is disconnected using the command is tracked as OTHER in the VDN Report.

goto step and goto vector commands

This section includes the following topics:

- Purpose on page 583
- Syntax and valid entries on page 584
- For information about unexpected results, see Troubleshooting vectors on page 689. on page 588
- Operation on page 589
- Time adjustments using goto conditionals on page 593
- Comparing none, # and numeric digits on page 593
- Media gateway, port network, and server vector conditionals on page 595
- Feature interactions on page 598
- Avaya IQ interactions on page 598
- CMS/BCMS interactions on page 598

Purpose

The **goto step** command allows conditional or unconditional movement (branching) to a preceding or subsequent step in the vector.

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

All parameters, options and value limits are identical for the **goto step** and **goto vector** commands.

Syntax and valid entries

goto step and goto vector				
goto step ***1-99*** **if** **or** **goto vector** ***1-2000***[1] **@step** ***1-99*** **if**				
A-Z, AA-ZZ	**>,<,=,<>,>=,<=**	threshold value or string of digits: *1-16, wildcards (?, +)*[2]*, [A-Z, AA-ZZ], V1-V9*		
	=,<>	**none**[3]**, #**[4]		
	in table	*1-100*[1]*, [A-Z, AA-ZZ], V1-V9*		
	not-in table			
ani	**>,>=,<>,=,<,<=**	*1-16, wildcards (?, +)*[3]*, [A-Z, AA-ZZ], V1-V9*		
	=,<>	**none**[3]**, #**[4]		
	in table	*1-100*[1]*, [A-Z, AA-ZZ], V1-V9*		
	not-in table			
available-agents	**in skill**	*hunt group*[5]*,* skills for VDN: **1st, 2nd, 3rd**	**>,>=,<>,=,<,<=**	*0-1499*[1] *1-1500*[1] *A-Z, AA-ZZ* *V1-V9*
	in split	*hunt group*[5]		

1. 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.
2. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (+) matches any or no characters at the specified position.
3. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
4. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
5. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

<table>
<tr><th colspan="7">goto step and goto vector (cont.)</th></tr>
<tr><td colspan="7">goto step 1-99 if
or
goto vector 1-2000[1] @step 1-99 if</td></tr>
<tr><td rowspan="2">calls-queued</td><td>in skill</td><td>hunt group[2], skills for VDN: 1st, 2nd, 3rd</td><td rowspan="2">pri</td><td rowspan="2">priorities:
l = low
m = medium
h = high
t = top</td><td rowspan="2">>
>=
<>
=
<
<=</td><td rowspan="2">0-098[1]
1-999[1]
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>in split</td><td>hunt group[2]</td></tr>
<tr><td>counted-calls</td><td>to vdn</td><td>vdn extension, latest, active[3]</td><td colspan="2">>,>=,<>,=,<,<=</td><td colspan="2">0-998[1]
1-999[1]
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td rowspan="5">digits</td><td>>,>=,<>,=,<,<=</td><td colspan="5">threshold value or string:
1-16, wildcards (?, +)[4], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td><>,=</td><td colspan="5">none[5]</td></tr>
<tr><td>=</td><td colspan="5">meet-me-access[6]</td></tr>
<tr><td>in table</td><td colspan="5" rowspan="2">1-100[1], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>not-in table</td></tr>
<tr><td rowspan="4">expected-wait</td><td>for best</td><td rowspan="2">>,>=,<>,=,<,<=</td><td colspan="5" rowspan="2">0-9999 seconds, [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>for call</td></tr>
<tr><td>for split</td><td>hunt group[2]</td><td rowspan="2">pri</td><td rowspan="2">priorities:
l = low
m = medium
h = high
t = top</td><td rowspan="2">>
>=
<>
=
<
<=</td><td rowspan="2">0-9998 sec
1-9999 sec
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>for skill</td><td>hunt group[2], skills for VDN: 1st, 2nd, 3rd</td></tr>
</table>

1. 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.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
3. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.
4. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (+) matches any or no characters at the specified position.
5. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
6. This item is available only with meet-me conference vectors.

goto step and goto vector (*cont.*)			
goto step *1-99* **if** **or** **goto vector** *1-2000*[1] **@step** *1-99* **if**			
holiday	**in table**	*1-99, [A-Z, AA-ZZ], V1-V9*	
	not-in table		
ii-digits	**>,>=,<>,=,<,<=**	*2-digit string*, wildcards (?, +)[2], *[A-Z, AA-ZZ], V1-V9*	
	<>,=	**none**[3]	
	in table	*1-100*[1], *[A-Z, AA-ZZ], V1-V9*	
	not-in table		
interflow-gpos	**>,>=,<>,=,<,<=**	*1-9, [A-Z, AA-ZZ], V1-V9*	
media-gateway	H.248 gateway ID[4] *1-999*	**=, <>**	**registered**
	all		
	any		
meet-me-full[5] (**goto step** only)			
meet-me-idle[5] (**goto step** only)			
no match[6]			

1. 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.
2. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (+) matches any or no characters at the specified position.
3. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
4. The maximum number of port networks and media-gateways supported 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.
5. This item is available only with meet-me conference vectors.
6. This item is available only with the Dial by Name feature.

goto step and goto vector (*cont.*)						
goto step ***1-99*** **if** **or** **goto vector** ***1-2000***[1] **@step** ***1-99*** **if**						
oldest-call-wait	**in skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**>,>=,<>, =<,<=**	*0-998* sec *1-999* sec *A-Z, AA-ZZ* *V1-V9*
	in split	*hunt group*[2]				
port-network	Port network ID[3] *1-999*		**=, <>**	**registered**		
	all					
	any					
queue-fail[4]						
rolling-asa	**for skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**		**>,>=,<>,=, <,<=**	*0-998 sec'* *1-999 sec* *A-Z, AA-ZZ* *V1-V9*	
	for split	*hunt group*[2]				
	for vdn	*vdn extension*, **latest**, **active**[5]				
server	**=, <>**	**main, ess, lsp**				
service-hours	**in table**		*1-99, [A-Z, AA-ZZ], V1-V9*			
	not-in table					
staffed-agents	**in skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**		**>,>=,<>,=, <,<=**	*0-1499*[1], *1-1500*[1] *A-Z, AA-ZZ* *V1-V9*	
	in split	*hunt group*[2]				

1. 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.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
3. The maximum number of port networks and media-gateways supported 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.
4. This item is available only with the Attendant Vectoring feature.

5. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

<table>
<tr><td colspan="9">goto step and goto vector (cont.)</td></tr>
<tr><td colspan="9">goto step 1-99 if
or
goto vector 1-2000[1] @step 1-99 if</td></tr>
<tr><td>time-of-day</td><td>is</td><td>mon, tue, wed, thu, fri, sat, sun, all</td><td>hour:
00-23</td><td>minute:
00-59</td><td>to</td><td>mon, tue, wed, thu, fri, sat, sun, all</td><td>hour:
00-23</td><td>minute:
00-59</td></tr>
<tr><td rowspan="4">V1-V9</td><td>>, <, =,<>, >=, <, <=</td><td colspan="7">threshold value or string of digits: 1-16, wildcards (?, +), [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>=,<></td><td colspan="7">none[2], #[3]</td></tr>
<tr><td>in table</td><td colspan="7" rowspan="2">1-100[1], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>not-in table</td></tr>
<tr><td rowspan="2">wait-improved for</td><td>best</td><td colspan="4">>, >=, <>, =, <, <=</td><td colspan="2" rowspan="2">0-9998 sec
1-9999 sec
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>skill</td><td>hunt group[4], skills for VDN:
1st, 2nd, 3rd</td><td>pri</td><td>priorities:
l = low
m = medium
h = high
t = top</td><td>>,>=, <>,=<, <=</td></tr>
<tr><td>split</td><td>hunt group[5]</td><td colspan="4"></td></tr>
<tr><td colspan="9">unconditionally</td></tr>
</table>

1. 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.
2. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
3. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
4. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

For more information about options required to enable the **goto** commands, see Communication Manager options required to enable vector commands on page 522.

Operation

This section includes the following topics:

- Basic operation on page 589
- General considerations on page 590
- Unconditional branching on page 591
- Conditional branching on page 591

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 [1]	Basic Call Vectoring	Advanced Vector Routing[2]	ANI and II-Digits Routing[3]
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		

1. For information about the comparators that can be used with these condition statements, see goto step and goto vector commands on page 583. 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.
2. For more information about this feature, see Advanced Vector Routing - EWT and ASA on page 257.
3. For more information about this feature, see ANI /II-digits routing and Caller Information Forwarding (CINFO) on page 169.

The following vector example includes several **goto** steps 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 example shown above, 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 help you 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. For more information, see Time adjustments on the Service Hours Table screen on page 368.
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 is considered to be in the table from the first second of the start time (for example, 08:00:00). Also, it is still considered to be in the table until the last second of the end time (for example, 17:00:59).

Comparing none, # and numeric digits

This section includes the following topics:

- How comparisons worked before vector variables on page 594
- How comparisons work now on page 594
- Comparisons still not allowed on page 595

How comparisons worked before vector variables

Prior to the introduction of Communication Manager 3.0, goto comparison tests using the keywords **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 could not 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

This section includes the following topics:

- Description of conditionals on page 595
- Reason to use on page 596
- Syntax of gateway conditionals on page 596
- Syntax of server conditionals on page 597

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

These conditionals allow you to monitor the communication server 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 following table describes the syntax of the gateway conditionals.

```
goto step [1-99] if media-gateway [1-x, all, any] [=, <>] registered
goto step [1-99] if port-network [1-x, all, any] [=, <>] registered
goto vector [1-99] @step [1-99] if media-gateway [1-x, all, any] [=, <>] registered
goto 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 CM 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 following table describes the syntax of the server conditionals.

```
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 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. ...
```

Feature interactions

For BSR and LAI, the command is considered a neutral vector command in all cases. When a call experiences Look Ahead interflow, the ANI value is sent along with the call only for ISDN PRI calls. ANI is not sent for internal or DCS calls.

Avaya IQ interactions

The ability to report on information associated with use of the **goto** commands is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS/BCMS interactions

The **goto step** command is not tracked on the CMS or on the BCMS.

The ANI and/or II-digits are passed to the CMS when the call first starts vector processing if the following is 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
- II-digits is available from the network
- The CMS is R3 (R3V5 for II-digits) or a newer version

ANI and II-digits are not passed to BCMS.

The **goto vector** command is tracked on CMS. The following database items are created.

goto Vector command		
Database Item	**Report Heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME GOTOCALLS/ GOTOTIME	Vector Flow Out	
INTIME	Avg Time In Vector	
INFLOWCALLS	Vector Flow In	new vector

CMS interaction notes for goto vector: The ANI and/or II-digits is passed to the CMS when the call first starts vector processing if the following is 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
- II-digits is available from the network

ANI and II-digits are not passed to BCMS.

messaging command

This section includes the following topics:

- Purpose on page 600
- Syntax and valid entries on page 600
- Requirements on page 601
- Operation on page 601
- Answer supervision considerations on page 603
- Feature interactions on page 603
- Avaya IQ interactions on page 604
- CMS interactions on page 604
- BCMS interactions on page 605

Purpose

The **messaging split/skill** command allows the caller to leave a message for the specified extension or the active or latest VDN extension (default).

Syntax and valid entries

messaging	**skill**	*hunt group*[1] **1st** (VDN skill) **2nd** (VDN skill) **3rd** (VDN skill)	**for extension**	*extension no.* **latest**[2] **active**[2] *A-Z, AA-ZZ* *V1-V9*
	split	*hunt group*[1]		

1. 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.
2. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

The split/skill involved must be a messaging system split/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/skill so that the caller may leave a message for the specified extension (call answering service or mail).

If the split/skill number specified in the command is a valid message service split/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/skill for call answering service.

If the call is queued to the message service split/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/skill queue is full.
- The messaging-system link is down.
- All messaging-system voice ports are out of service.
- The message service split/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 and/or supervisor stations may 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/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 he or she 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 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    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

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

Feature interactions

Messaging-system hunt group: The command can use a messaging-system hunt group in its operation.

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

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/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 they subsequently appear in the display to the answering agent.

Mixed-length numbering plans: The messaging system does not support mixed-length numbering plans.

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 considered as either a call acceptance vector command or a neutral vector command.

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

- Call terminates to an agent or to a messaging-system port.
- Call queues to a messaging split/skill.

Neutral vector command: The command is considered 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 will not work prior to Communication Manager 2.0 load 205.

For an example, see Using a messaging step in a vector on page 602

Avaya IQ interactions

The ability to report on information associated with use of the **messaging** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

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. These 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). These 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.

BCMS interactions

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

queue-to command

This section includes the following topics:

- Purpose on page 606
- Syntax and valid entries on page 606
- Requirements on page 607
- Operation on page 607
- queue-to split command on page 608
- Answer supervision considerations on page 611
- Feature interactions on page 612
- Avaya IQ interactions on page 612
- CMS interactions on page 612
- BCMS interactions on page 613

Purpose

The **queue-to** command unconditionally queues a call to a split/skill, attendant group, attendant, or hunt group, and assigns a queuing priority level to the call in case all agents or attendants are busy.

Syntax and valid entries

queue-to	**attd-group**[1]			
	attendant[1]	*extension no.*		
	best			
	hunt-group[1]	*group number*[2]	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op
	skill	*hunt group*[3], VDN skills (**1st, 2nd, 3rd**)		
	split	*hunt group*[3]		

1. This item is available with only the Attendant Vectoring feature.
2. A valid group number is a vector-controlled hunt group of any type (ACD, UCD, and so on).
3. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

The split/skill involved must be vector-controlled.

Operation

A call sent with this command either connects to an available agent or attendant in the specified resource or enter the resource's queue. When it enters the queue, feedback is not given to the caller by this command.

Note:
In Attendant Vectoring, a wait-time 0 secs hearing ringback step should be used to give immediate feedback to the caller. The queue-to command does not provide ringback until the call is actually ringing the attendant. The wait-time step should be implemented as the first vector step or as the step immediately before the queue-to step.

If single-site BSR is enabled, **`queue-to best`** queues or delivers a call to the best local split/skill found by a consider series. If multi-site BSR is enabled, the best resource may be at a remote location; in this case, **`queue-to best`** interflows the call to the interflow VDN defined for that location on the BSR Application screen.

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

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

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

- Desired split/skill's queue is full.
- Desired split/skill's is not vector-controlled.
- Desired split/skill's has no queue and also no available agents.
- Call has been previously queued to three different split/skills.

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

A **queue-to best** command will have the same operation and interactions as the **queue-to split/skill** command when the best resource is a local split/skill. When the best resource is at a remote location, the **queue-to best** command will function as an unconditional **route-to** command (with cov=n) performing LAI.

When a **queue-to best** command executes, it initializes the data for the best resource (the best data) the consider series found for this call. If no best data has been defined by the consider series, a vector event is logged and processing continues at the next vector step. A consider series might not produce best data for any of the following reasons:

- All resources considered are unstaffed
- No resource considered has an open queue slot
- Best data has been initialized before execution of the **reply-best** step (because there are no consider steps in the status poll vector or because the vector contains a prior step that initializes best data).

For a list of events and vector commands that initialize best data produced by consider series, see Events that clear best data on page 564.

If a queue attempt to a local resource fails, a vector event is logged and processing continues at the next vector step. The best data is initialized.

If an interflow attempt to a remote resource fails, a vector event is logged and processing continues at the next vector step. If a local split/skill was identified as best at some point in the consider series before the interflow attempt, the call is queued to the local resource. Whether or not the call can be queued locally in this case, the best data is initialized and processing continues at the next vector step.

queue-to split command

This command queues a call unconditionally. The command sends a call to a split and assigns a queuing priority level to the call in case all agents are busy. The following topics also apply to the **check split** command.

This section includes the following topics:

- General considerations on page 609
- Multiple split queuing on page 609
- Option with the VDN as the coverage point on page 610

General considerations

You should understand the following considerations when you use the **queue-to split** or **check split** commands:

- Make split queues large enough to allow all incoming calls to be queued. If a queue is too small, a **queue-to split** or a **check split** command might fail to queue a call due to a lack of available queue slots and the call will be dropped.
- Include a vector step that tests a split queue before queuing occurs and an alternate step that provides fallback treatment if the queue is full.
- When calls are and/or to backup splits, they also remain in queue for any previous splits to which they may have been 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 those splits that are vector-controlled. A split is considered to be vector-controlled if **yes** is entered in the Vector field of the Hunt Group screen.
- When the EAS feature is enabled, *Multiple Split Queuing* is referred to as *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 shown above, step 1 test whether the main split queue (which has 10 queue slots) is full, and branches to one of the following. A low priority is specified in so that calls in queue at all priority levels are counted.

Note:
To avoid completing vector processing without queuing the call to a split, it is always good practice to check a split's queue before queuing to that split. If the queue is full, alternate treatment such as queuing to an alternate split should be provided.

If the main split queue is full, a **goto step** 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 a 12-second delay. If the call is not answered within this time interval, 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 into the split's 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 the VDN as the coverage point

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

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 at this
       time.  Please call back tomorrow.]
```

In steps 3 and 8 of the vector example shown above, the caller is given the option of leaving a recorded message, but the **queue-to split** command instead of the **messaging split** command is used in each case. Thus, the call is actually queued to the AUDIX split.

However, a **messaging split** command does not queue the call to the split. Instead, if it is successful, it connects the caller to the split so the caller can leave a message for the specified extension. However, termination to the split may turn out to be unsuccessful due to a factor that cannot be checked by vector processing. For example, the AUDIX link might not be functioning, or all AUDIX ports might be out of service.

As a result of the queuing process, a wait-announcement loop can be included after each **queue-to split** step, and the appropriate loop can 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

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

Feature interactions

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

For BSR and LAI, the command can be considered either a call acceptance vector command or a neutral vector command.

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

- Call terminates to an agent.
- Call queues to a split/skill.
- BSR interflowed call is accepted at remote interflow vector.

The command is considered 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/skill.

Avaya IQ interactions

The ability to report on information associated with use of the **queue-to split/skill** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

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).

For more information on the database items and reports, see Avaya *Avaya CMS Database Items and Calculations*, and *Avaya CMS Supervisor Reports*.

BCMS interactions

The total number of calls to the VDN that are queued using the command and then answered by an agent within a specified time period is tracked as ACD Calls in the VDN Report. The average time that calls spend in a vector before being connected using the command as an ACD call 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 these calls as outflow in the VDN Report.

reply-best

This section includes the following topics:

- Purpose on page 614
- Syntax on page 614
- Requirements on page 614
- Operation on page 615
- Avaya IQ interactions on page 615
- CMS/BCMS interactions on page 616

Purpose

The **reply-best** command is used only in status poll vectors in multi-site BSR applications, where it returns best data for its location to the primary vector on the origin switch.

Syntax

```
reply-best
```

Note:
This multi-site BSR command is available only when the Virtual Routing feature is enabled.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

The EAS feature must be enabled 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 multi-site 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
- Initializes, or clears, the best data
- Terminates processing in the status poll vector

If the incoming call is not a trunk call, the **reply-best** command will drop the call and log a vector event. No status data will be returned to the origin switch.

If the consider series yields no best data, the **reply-best** command will drop the incoming call without returning answer supervision, terminate vector processing, and return an infinite value for EWT in the DISCONNECT message. A consider series might not produce best data for any of the following reasons:

- All resources considered are unstaffed
- No resource considered has an open queue slot
- The best data has been initialized before execution of the **reply-best** step (because there are no consider steps in the status poll vector or because the vector contains a prior step that initializes best data.

For a list of events and vector commands that initialize best data produced by consider series, see Events that clear best data on page 564.

Answer supervision considerations command

The **reply-best** step does not return answer supervision.

Avaya IQ interactions

The ability to report on information associated with use of the **reply-best** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS/BCMS interactions

Operation of the **reply-best** command is not reported or tracked by the CMS or by the BCMS.

return command

This section includes the following topics:

- Purpose on page 617
- Syntax on page 617
- Operation on page 617

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
```

For information about unexpected results, see Troubleshooting vectors on page 689.

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 this failed return step. Consider the possibility of a failed return step when programming vector subroutines.

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

This section includes the following topics:

- Purpose on page 619
- Syntax and valid entries on page 620
- Requirements on page 620
- Operation on page 620
- Route-to number command on page 624
- Answer supervision considerations on page 626
- Feature interactions on page 626
- Avaya IQ interactions on page 628
- CMS interactions on page 629
- BCMS interactions on page 631

Purpose

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

Syntax and valid entries

route-to[1]	**digits**	**with coverage**	**y, n**					
	meetme[2]							
	number	*0-9* ***** **#** **~p** **~m** **~s** **~w** **~W** *A-Z, AA-ZZ* *V1-V9* **~r** *<up to 14 digits>*[3] **~r**[*V1-V9*][3] **~r**[*A-Z, AA-ZZ*][3]	**with cov**	**y, n**	**if**	**digit**	**>** **>=** **<>** **=<** **<=**	*0-9* **#**[4]
						interflow-qpos	**<** **=** **<=**	*1-9*
						unconditionally		
	name1[5]	**with coverage**	**y, n**					
	name2[5]							
	name3[54]							

1. The route-to digits and route-to number commands support the Service Observing FACs, remote logout of agent FAC, remote access extension, attendant access number, and other dialable destination numbers.
2. This item is available only with meet-me conference vectors.
3. When the specified number is preceeded by ~r, Network Call Redirection is attempted.
4. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
5. This item is available only with the Dial by Name feature.

For information about unexpected results, see Troubleshooting vectors on page 689.

Requirements

The Dial by Name feature must be enabled to use the **route-to name** command.

Operation

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

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/hunt group, station, and so on.)
- VDN extension
- Attendant or Attendant Queue
- 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, 10-digit DDD, and so on.)
- Remote Access Extension.
- Service Observing FAC
- Another Avaya switch

  For more information, see Using the route-to command for NCR on page 623.
- Remote Logout of Agent FAC

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

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 cov=n for a covered call regardless of the cov setting on the **route-to** command.

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

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

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

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 UDP extension: Standard AAR/ARS processing is performed to select the trunk group and outpulse the digits. If a trunk is seized, vector processing terminates, and the calling party hears feedback provided by the far end. Otherwise, the call cannot complete successfully (because no trunks are available, the FRL/COR is restricted, etc.), and vector processing continues at the next vector command.

If the number is a TAC **plus digits, and a trunk is seized:** Vector processing terminates, and the calling party hears feedback provided by the far end. Otherwise, the call cannot complete successfully (because no trunks are available, the COR is restricted, 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

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

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

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

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

You can use the following variables in the number field:

- A - Z vector variable
- V1 - V9 VDN variable assigned to the active VDN for the call

Using the route-to command for NCR

You can use variables with the **~r** vector step in the **route-to** command to activate Network Call Redirection (NCR). 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 variable format can contain up to 14 digits of redirection because **~r** takes up 2 digit spaces in the 16-digit field.
- **~r***V1-V9* - This option allows you to enter a VDN variable. For example ~rV1. This variable can contain up to 16 digits.
- **~r***A-Z* - This option allows you to enter a vector variable. For example, ~rA. This variable can contain up to 16 digits.

For examples, see Using route-to number ~r vector step to activate NCR on page 393.

Coverage parameter

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

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

Note:
For detailed information about the operation of the route-to command with or without coverage for the different destinations see the table shown in Switch route-to command operation on page 767.

Route-to number command

The **route-to number** command is used to route calls to a vector-programmed number.

This section includes the following topics:

- About interflow routing on page 624
- General considerations on page 624
- Service Observing routing on page 625

About interflow routing

Calls can be routed to a programmed number using a process that is known as *interflow*.

Interflow allows calls directed to a split can be redirected to an internal or an external destination. For Basic Call Vectoring, this destination is represented by a number programmed in the vector. The number 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

General considerations

You should understand the following considerations before you use interflow routing:

- Calls should not interflow back and forth between vectors on remote servers and local servers. This process could cause a single call to use up all available trunks.
- When the **route-to number** command is used to chain multiple vectors together to enhance 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 it was previously assigned.
3. Any previously assigned wait treatment is disabled.

4. Processing then 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 [We're sorry, all our agents are busy. Please leave a
         message. Thank you.]
      2. messaging split 18 for extension 3000
      3. disconnect after announcement 2505 [We cannot take a message at this
         time. Please call back tomorrow.]
```

In the example shown above, Vector 55 provides a series of initial vector steps that test the queue status for split 1. Depending on the outcome of those 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 that is assigned to vector 100. When the **route-to number** command is executed, vector processing in Vector 55 is terminated, 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 these 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:
It is good practice to provide an announcement to explain to the caller that the messaging connection could not be made.

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 feature access code (FAC).

Important:
The following example does not provide security checks and should 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 shown above, the caller is connected to a listen-only Service Observing FAC. Once connected, the person who is service observing must dial the extension number that is to be observed. To observe in a listen or talk mode, the observer would dial a different VDN.

Related topics

- For more information about the Service Observing feature, see:
  - *Feature Description and Implementation for Avaya Communication Manager*
  - *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference*
- For more information about the route-to number A to Z and AA to ZZ variables, see Variables in Vectors on page 107.

Answer supervision considerations

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

Feature interactions

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

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

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

If the call covers to a VDN, the command supports a remote messaging-system interface to a local hunt group extension that is assigned as a remote messaging-system hunt group. The remote messaging-system hunt group (which has no members and cannot be vector-controlled) forwards the call to the remote messaging-system destination in the same manner as when the hunt group is assigned as a point in the coverage path. A DCS link down condition for a call that covers to a VDN is treated as a direct call to the messaging system.

If the command is directed to a station with bridged appearances, the bridged appearance button lamps are updated.

The following destinations always result in a failure, and vector processing continues at the next step:

- Controlled trunk group
- Code calling FAC
- Facility test call
- TAAS access code
- Priority access code
- Loudspeaker paging access code
- Station Message Detail Recording (SMDR) account code
- Voice message retrieval access code.

If the command is executed and Direct Outward Dialing (DOD) is in effect, the COR of the latest VDN is compared with the COR of the called facility to determine if the call is permitted. If access is not permitted, the command fails and vector processing continues. In the case where a COR requiring the entry of account codes is assigned to a VDN, and the command is executed by the associated vector, the command is unsuccessful, and vector processing continues at the next step.

The individual extension number assigned to an attendant console can be used as the command's argument.

A call processed by the command can wait in the individual attendant queue and is subsequently removed from vector processing.

The command can access both public and private networks.

If the command dials the attendant, and if the system is in night service, the call routes to the DID Listed Directory Number (LDN) night destination.

The command can place AAR/ARS calls that implement subnet trunking, which is the routing of calls over trunk groups that terminate in switches with different dial plans.

Authorization codes are disabled with respect to routing using VDNs. In other words, if authorization codes are enabled, and a **route-to** command in a prompting vector accesses AAR or ARS, and the VDN's FRL does not have the permission to utilize the chosen routing preference, no authorization code is prompted for, and the **route-to** command fails.

If the command routes the call without coverage to a display station, the station displays the following: a = Originator Name to VDN Name.

If the command calls a station that is a member of a pickup group, the call can be picked up by another pickup group member.

Anytime a **route-to with cov n** command initiates a call over ISDN-PRI facilities and LAI is optioned, the call will be treated on a Look-Ahead basis. However, if the command is used with the **coverage yes** option in effect, unconditional interflow results.

For LAI, the **route-to** command can be considered either a call acceptance vector command or a neutral vector command. The command is considered a call acceptance vector command whenever one of the following is true:

- Command terminates to a valid local destination.
- Command successfully seizes a non-PRI trunk.
- Command execution results in a LAI call attempt, and the call is accepted by the far end switch.

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

- Termination is unsuccessful.
- Trunk is not seized.
- LAI call attempt is denied by the far end switch.

For a call that covers or forwards to a VDN, the **route-to with coverage y** command functions the same way as the **route-to with coverage n** command. For a covered or forwarded call, the coverage option for the command is disabled since such a call should not be further redirected.

A **route-to with cov y** to a station that has call forwarding activated is forwarded.

Service Observing can be initiated with Call Vectoring using the route-to command. For detailed instructions, see Service Observing routing on page 625.

Note:
Appendix I: Operation details for the route-to command on page 767 gives a detailed description of the feature interactions for the **route-to** number with and without coverage command.

Avaya IQ interactions

The ability to report on information associated with use of the **route-to** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

Tracking of the **route-to digits** command varies according to the destination successfully routed to, as follows.

Routed to station or to attendant		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Flow Out Vector Flow Out	1st split
DEQUECALLS/ DEQUETIME	Dequeued Calls Dequeued Avg Queue Time	2nd/3rd splits
INTIME	Avg Time In Vector	
CONNECTCALLS/ CONNECTTIME	Other Calls Connect	answered calls on G3

Routed to trunk		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Flow Out Vector Flow Out VDN Flow Out	1st split
DEQUECALLS/ DEQUETIME	Dequeued Calls Dequeued Avg Queue Time	2nd/3rd splits

Routed to VDN		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Flow Out Vector Flow Out VDN Flow Out	1st split

Routed to VDN		
DEQUECALLS/ DEQUETIME	Dequeued Calls Dequeued Avg Queue Time	2nd/3rd splits
INTIME	Avg Time In Vector	
INFLOWCALLS	Vector Flow In VDN Flow In	new vector new VDN
INTERFLOWCALLS/ INTERFLOWTIME	VDN Flow-Interflow	

Routed to Split or Hunt Group		
Database item	**Report heading**	**Notes**
OUTFLOWCALLS/ OUTFLOWTIME	Flow Out	1st split
DEQUECALLS/ DEQUETIME	Dequeued Calls Dequeued Avg Queue Time	2nd/3rd splits
INTIME	Avg Time In Vector	
CALLSOFFERRED		new split
MEDCALLS/ HIGHCALLS		no priority/priority

Note:
For calls that route to a split or a hunt group and later intraflow to a station or to an attendant, OTHERCALLS/OTHERTIME are tracked in the vector and in the VDN tables.

Split calls are also shown in the standard reports according to the final disposition of the call.

Calls that route over a trunk are LAI calls. When a call attempts to route to a trunk (Look-Ahead Interflow), the LOOKATTEMPTS database item is tracked and reported as Look-Ahead Interflow Attempts. If the call successfully routes, LOOKFLOWCALLS/LOOKFLOWTIME are tracked and reported as Look-Ahead Interflow Completions. Interflow always occurs whenever the **`with coverage yes`** option is in effect.

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, 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). DEQUECALLS/DEQUETIME is tracked in the second and third splits if these splits are not the answering split, 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, and the call is reported as Flow In.

If the command directs a call to a destination, 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.

A call abandoned after the command routes the call to a station or an attendant is tracked in the VDN tables as ABNCALLS/ABNTIME.

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 on the CMS database items and reports, see *Avaya CMS Database Items and Calculations*, and *Avaya CMS Supervisor Reports*.

BCMS interactions

A call advanced to another position using the 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

This section includes the following topics:

- Description of the set command on page 632
- Reason to use the set command on page 632
- Syntax and valid entries on page 633
- Variable, digits buffer, and asaiuui on page 633
- Operand1 on page 638
- Operand2 on page 639
- Operators on page 639
- Set command considerations on page 640

Description of the set command

The **set** vector command can do 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 the set command

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***]

Command	Variables or digits		Operand1	Operator	Operand2
set	*user-assigned*[1] type *A-Z or AA-ZZ* vector variable	**=**	*user-assigned*[1] type *A-Z or AA-ZZ* vector variable	**ADD** **SUB** **MUL** **DIV** **CATL** **CATR** **MOD10** **SEL**	*user-assigned*[1] *A-Z or AA-ZZ* vector variable
	asaiuui *A-Z or AA-ZZ* vector variable				
	digits[2]		system-assigned[3] *A-Z or AA-ZZ* vector variable		system-assigned[3] *A-Z or AA-ZZ* vector variable
			V1-V9 VDN variable		directly-entered numeric string[4]
			digits		*V1-V9 VDN variable*
			none		**digits**
					none

1. Only global or local collect type vector variables can be assigned using the set command.
2. The collected digits buffer holds up to 16 digits.
3. For example, ani, asaiuui, doy, and so on.
4. Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.

For information about unexpected results, see Troubleshooting vectors on page 689.

Variable, digits buffer, and asaiuui

You can enter a variable, use a collected digits buffer, or assign asaiuui user data in this field:

- Variable on page 634
- digits on page 634
- Assign to asaiuui variable type on page 634

Variable

You can enter user-assigned A to Z and AA to ZZ collect vector or the asaiuui variable types in this field. The collect vector variable can be either local or global.

For more information, see User-assigned vector variable types on page 130.

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

See also:

For more information about the digits buffer and dial-ahead digits, see Dial-ahead digits and the digits buffer on page 640.

Assign to asaiuui variable type

The set command can be used to assign a value to a defined vector variable and to replace or append that 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.

This 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 active.

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 on page 635 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.
- Will be passed by a subsequent event report to an ASAI adjunct in an ASAI IE.
- Will be 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 information obtained from vectoring to make ASAI routing decisions or to provide adjunct display information to the agent.

For example, you can choose to:

- Give higher priority status to calls that have been waiting at both remote and local call centers beyond a certain amount of time.
- 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 those 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 157 85670 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 CM. If the Call Vectoring set command initially creates the ASAI UUI data, the protocol discriminator is set to 0x04 by CM.
- 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) only containts 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 preceeded 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 varible 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 on page 638.

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 CM.
- The protocol discriminator for the shared UUI format is usually set to 0x00 (user specified), however it can be set to 0x04 which would indicate IA5 (ASCII) coding of the characters. In this case it 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 CM, 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 can not 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) only contains 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 on page 635.
- 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 #.

See also:

For application examples, see Assigning ASAI UUI values on page 803.

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.

  For more information, see User-assigned vector variable types on page 130.

- The system-assigned A to Z and AA to ZZ vector variables, such as: ani, asaiuui, doy, and so on.

  For more information, see System-assigned vector variable types on page 121.
- V1 to V9 VDN variables

  For more information, see VDN variables on page 151.
- 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.

  For more information, see User-assigned vector variable types on page 130.
- The system-assigned A to Z and AA to ZZ vector variables, such as: ani, asaiuui, doy, and so on.

  For more information, see System-assigned vector variable types on page 121.
- V1 to V9 VDN variables

  For more information, see VDN variables on page 151.
- **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.

See also:

For details and examples, see Advanced set command rules and applications on page 773.

Set command considerations

This section includes the following topics:

- Dial-ahead digits and the digits buffer on page 640
- DIGITS message on page 641
- Allowed assignments on page 641
- Assigning a new value to a collect variable on page 641
- Determining the number of digits on page 641
- Clearing the digits buffer on page 641

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.

If the digits buffer is	And the dial-ahead digits are	Then set digits = digits ADD 1111
1234	5678	Sets the digits buffer to 2345 and the dial-ahead digits remain as 5678

If the digits buffer is	And the dial-ahead digits are	Then 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 the Call Management System (CMS) when the **set** command changes the digits content. Only the last digits sent are saved for the call. See *Avaya CMS Reports*.

Allowed assignments

Assignment is only allowed to a collect type vector 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 (except the collect 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 seeing the contents of 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
```

stop command

This section includes the following topics:

- Purpose on page 642
- Syntax on page 642
- Requirements on page 642
- Operation on page 642
- Answer supervision considerations on page 643
- Feature interactions on page 643
- Avaya IQ interactions on page 643
- CMS interactions on page 644
- BCMS interactions on page 644

Purpose

The **stop** command halts the processing of any subsequent vector steps.

Syntax

```
stop
```

For information about unexpected results, see Troubleshooting vectors on page 689.

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 example shown above, if the **stop** command is reached, the caller remains in queue at split 21 and continues to hear ringback. Further vector processing is stopped and vector processing does not continue to step 6. Therefore, callers connected to split 21 do no hear a busy signal.

Answer supervision considerations

The command has no effect on answer supervision.

Feature interactions

For LAI, the command is considered a neutral vector command in all cases except when a call is dropped, then it is considered a denial.

Avaya IQ interactions

The ability to report on information associated with use of the **stop** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS interactions

When the 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.

BCMS interactions

None.

wait-time command

This section includes the following topics:

- Purpose on page 645
- Syntax and valid entries on page 645
- Requirements on page 646
- Operation on page 646
- Considerations on page 649
- Feature interactions on page 651
- Avaya IQ interactions on page 652
- CMS/BCMS interactions on page 652

Purpose

The **wait-time** command enables you to create a vector that delays the call with audible feedback. In presenting an example of a delay announcement earlier in this section, we mentioned that this type of announcement is usually coupled with a delay step. A delay step is provided by the **wait-time** command, which allows the caller to remain on hold for at least the amount of time that is indicated in the command.

Syntax and valid entries

wait-time	*0-999*	**secs**	**hearing**	**music, ringback, silence, i-silent**		
	0-480[1]	**mins**		*audio source ext.*[2] *A-Z, AA-ZZ* *V1-V9*	**then**	**music** **ringback** **silence** **continue**[3]
	0-8[1]	**hrs**				

1. This option is not available for vector administration done through Avaya Call Management System or Visual Vectors.
2. This consists of a valid announcement or music source extension that is defined on the announcement audio sources form.
3. The continue treatment is valid only with Multiple Audio/Music Sources. It indicates that the caller continues to hear the alternate audio or music source using an announcement until another vector command takes effect.

For information about unexpected results, see Troubleshooting vectors on page 689.

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 that the Vectoring (G3V4 Enhanced) customer option be enabled.

Operation

This section includes the following topics:

- Basic operation on page 646
- Call delay with audible feedback on page 647
- Multiple audio or music sources on delay on page 647
- Call delay with continuous audible feedback on page 648
- Multiple music sources on hold on page 648

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's terminal).
- Switch receives a destination from the ASAI adjunct.
- Vector disconnect timer expires.

Wait times up to 8 hours are allowed for customers who want to use the ASAI Phantom Call feature to track e-mail 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 of our agents are busy. Please hold.]
     wait-time 20 seconds hearing music
```

In the example shown above, the caller waits at least 20 seconds for the call to be answered by an agent. During this wait period, the caller is provided with system music, which is one type of feedback 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 either 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 and/or a Touch-Tone Receiver (TTR). For more information about TTRs, which are used with the Call Prompting feature, see Touch-tone collection requirements on page 238.

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. For instructions for entering an audio or music source on this screen, see *Administrator Guide for Avaya Communication Manager*.

With the Multiple Audio/Music Sources feature, 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. Or, additional advertising messages can be heard by the callers as they wait for an available agent.

An example of an announcement that includes an alternative audio or music source in the **wait-time** step is shown below.

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 shown above, if the call is not answered in 20 seconds, the caller hears system music until a subsequent **announcement**, **busy**, **collect**, **converse-on**, **disconnect** or **wait-time** step is encountered.

You can specify **music** (system music), **ringback**, **silence**, or **continue** for the **then** option. When continue is specified, the caller continues to hear the alternative audio or music source until it 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.

Note:
For more information, about Tenant Partitioning, see *Feature Description and Implementation for Avaya Communication Manager*.

You should understand the following considerations about how TN works with multiple music sources on hold:

- 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 (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 these 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, the switch 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.

Considerations

When music is indicated as a treatment, it refers to the system music, not 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/music sources

The expanded **wait-time _ secs hearing <extension> then <treatment2>** command provides what is known as Multiple Audio/Music Sources wait treatment. The **<extension>** option defines an audio or music source that is assigned on the Announcements/Audio Source administration 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 announcement/audio source types listed above 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 recommended for call center applications)
- Barge-in operation (**Queue** field set to **b**)

In addition, integrated board announcements can be set to play once (integrated) or to repeat after each playing continuously (integ-rep). For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* , and the Announcements/Audio Sources screen reference in *Administrator Guide for Avaya Communication Manager*.

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 can not be connected to the source. Failure to connect to the source can result from conditions such as:

- source not available - extension/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 will occur:

- If **<treatment2>** is set to **continue**, the caller returns to what they were hearing 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, regardless 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 it 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/music source specified by the **<extension>** stops (disconnects) before the wait-time period expires or the caller cannot be connected to that source (source not available), the caller will hear the source specified by the **then <treatment2>** segment of the vector. In this case, if **<treatment2>** is specified as **continue**, then the caller hears silence.

Answer supervision

If the **music** or **audio source** treatment is included in the command, answer supervision is triggered. If the command is encountered and answer supervision was sent previously, the caller hears the treatment specified in the current command. If, for a CO trunk user, the command with **silence**, **ringback**, or **i-silent** treatment is encountered prior to answer supervision, the caller continues to hear ringback from the CO.

Feature interactions

Music-on-Hold: When the command is implemented with music as the treatment, the system-wide music-on-hold feature must be administered. Otherwise, the caller hears silence. 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.

Look-Ahead Interflow (LAI): For LAI, the wait-time command is considered a call acceptance vector command in all cases, except i-silent, which is considered a neutral vector command.

Avaya IQ interactions

The ability to report on information associated with use of the **wait-time** command is not currently supported. Support for reporting on this command is planned for a future Avaya IQ release.

CMS/BCMS interactions

The command is not tracked on the CMS or on the BCMS. Vectors with **wait-time** steps are only accessible to CMS if the time unit is administered in **secs**.

Appendix A: Job aids

This section includes the following topics:

- Vector commands job aid on page 653
- Vector variables job aid on page 664

Vector commands job aid

The vector command job aid shown in this section lists the Call Vectoring commands, together with the various conditions, and parameter options and values that are available for use with each command.

Most vector commands require one or more input values for the command, as well as for various parameters, such as an announcement extension number, a time interval, a maximum queue size, and so forth. 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 Avaya Communication Manager on Avaya Media Servers*. You can find the latest capacity tables from the Avaya support Website at:

http://www.avayadocs.com

For detailed information about these commands, see Call Vectoring commands on page 521.

#	[***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 this command.***]

adjunct routing link	*1-64* - CTI Link ID[1] *[A-Z, AA-ZZ]* *V1-V9*

1. Link capacity depends on your release and configuration. For more information, see *System Capacities Table for Avaya Communication Manager on Avaya Media Servers*.

announcement	*extension no.* *[A-Z, AA-ZZ]* *V1-V9*

busy

Check	**best**	**if**	**expected wait**		< 1-9999 seconds, > 0-9999 seconds		
			unconditionally				
			wait improved		< 1-9999 seconds, > 0-9999 seconds		
	skill	hunt group[1], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = low **m** = medium **h** = high **t** = top	**if**	**available-agents** > 0-1499[2]	**all-levels**
							pref-level skill level 1[3]
							pref-range skill level 1[3] **to** skill level 2[3]
	skill	hunt group[1], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = low **m** = medium **h** = high **t** = top	**if**	**calls-queued** < 1-999[2] **expected-wait** < 1-9999 seconds **oldest-call-wait** > 1-999 seconds **rolling-asa** < 1-999 seconds **staffed-agents** > 0-1499[2] **wait-improved** > 0-9999 seconds **unconditionally**	
	split	hunt group[1]					
	split	hunt group[1]	**pri**	**l** = low **m** = medium **h** = high **t** = top	**if**	**available-agents** > 0-1499[2]	

1. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
2. 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.
3. Skill levels are 1-16 (1 is best, 16 is lowest). Skill Level 2 must be greater than or equal to Skill Level 1.

collect	**ced**	**for**	**none** *A-Z, AA-ZZ*		
	cdpd				
	1-16	**digits after announcement**	*extension no.* **none** *A-Z, AA-ZZ* *V1-V9*	**for**	**none** *A-Z, AA-ZZ*

consider	**location**[1] (multi-site BSR only)	*1-255* *A-Z, AA-ZZ* *V1-V9*			**adjust by**	*0-100* percent *A-Z, AA-ZZ* *V1-V9*
	skill	*hunt group*[2], skills for VDN: **1st, 2nd, 3rd**	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op		
	split	*hunt group*[2]				

1. This item is available only with the Virtual Routing feature.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

converse-on	**skill**	*hunt group*[1], skills for VDN: **1st, 2nd, 3rd**	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op	**passing**	*6-digit string* ***** **#** **ani** **vdn** **digits qpos** **wait** *A-Z, AA-ZZ* *V1-V9*	**and**	*6-digit string* ***** **#** **ani** **vdn** **none digits qpos** **wait** *A-Z, AA-ZZ* *V1-V9*
	split	*hunt group*[1]				**none**	**and**	**none**

1. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

disconnect	**after announcement**	*extension no.* **none** *A-Z, AA-ZZ* *V1-V9*

<table>
<tr><th colspan="5">goto step and goto vector</th></tr>
<tr><td colspan="5">goto step 1-99 if
or
goto vector 1-2000[1] @step 1-99 if</td></tr>
<tr><td rowspan="3">A-Z, AA-ZZ</td><td>>,<,=,<>,>=,<=</td><td colspan="3">threshold value or string of digits:
1-16, wildcards (?, +)[2], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>=,<></td><td colspan="3">none[3], #[4]</td></tr>
<tr><td>in table
not-in table</td><td colspan="3">1-100[1], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td rowspan="3">ani</td><td>>,>=,<>,=,<,<=</td><td colspan="3">1-16, wildcards (?, +)[3], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>=,<></td><td colspan="3">none[3], #[4]</td></tr>
<tr><td>in table
not-in table</td><td colspan="3">1-100[1], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td rowspan="2">available-agents</td><td>in skill</td><td>hunt group[5]
skills for VDN:
1st, 2nd, 3rd</td><td rowspan="2">>,>=,<>,=,<,<=</td><td rowspan="2">0-1499[1]
1-1500[1]
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>in split</td><td>hunt group[5]</td></tr>
</table>

1. 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.
2. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (+) matches any or no characters at the specified position.
3. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
4. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
5. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

<table>
<tr><th colspan="8">goto step and goto vector (cont.)</th></tr>
<tr><td colspan="8">goto step 1-99 if
or
goto vector 1-2000[1] @step 1-99 if</td></tr>
<tr><td rowspan="2">calls-queued</td><td>in skill</td><td>hunt group[2], skills for VDN: 1st, 2nd, 3rd</td><td rowspan="2">pri</td><td rowspan="2">priorities:
l = low
m = medium
h = high
t = top</td><td rowspan="2">>
>=
<>
=
<
<=</td><td rowspan="2">0-098[1]
1-999[1]
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>in split</td><td>hunt group[2]</td></tr>
<tr><td>counted-calls</td><td>to vdn</td><td>vdn extension, latest, active[3]</td><td colspan="2">>,>=,<>,=,<,<=</td><td colspan="2">0-998[1]
1-999[1]
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td rowspan="5">digits</td><td>>,>=,<>,=,<,<=</td><td colspan="6">threshold value or string:
1-16, wildcards (?, +)[4], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td><>,=</td><td colspan="6">none[5]</td></tr>
<tr><td>=</td><td colspan="6">meet-me-access[6]</td></tr>
<tr><td>in table</td><td colspan="6" rowspan="2">1-100[1], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>not-in table</td></tr>
<tr><td rowspan="4">expected-wait</td><td>for best</td><td rowspan="2">>,>=,<>,=,<,<=</td><td colspan="5" rowspan="2">0-9999 seconds, [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>for call</td></tr>
<tr><td>for split</td><td>hunt group[2]</td><td rowspan="2">pri</td><td rowspan="2">priorities:
l = low
m = medium
h = high
t = top</td><td rowspan="2">>
>=
<>
=
<
<=</td><td rowspan="2">0-9998 sec
1-9999 sec
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>for skill</td><td>hunt group[2], skills for VDN: 1st, 2nd, 3rd</td></tr>
</table>

1. 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.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
3. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.
4. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (+) matches any or no characters at the specified position.
5. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
6. This item is available only with meet-me conference vectors.

goto step and goto vector (*cont.*)			
goto step ***1-99*** **if** **or** **goto vector** ***1-2000***[1] **@step** ***1-99*** **if**			
holiday	**in table**	*1-99, [A-Z, AA-ZZ], V1-V9*	
	not-in table		
ii-digits	**>,>=,<>,=,<,<=**	*2-digit string*, wildcards (*?, +*)[2], *[A-Z, AA-ZZ], V1-V9*	
	<>,=	**none**[3]	
	in table	*1-100*[1], *[A-Z, AA-ZZ], V1-V9*	
	not-in table		
interflow-gpos	**>,>=,<>,=,<,<=**	*1-9, [A-Z, AA-ZZ], V1-V9*	
media-gateway	H.248 gateway ID[4] *1-999*	**=, <>**	**registered**
	all		
	any		
meet-me-full[5] (**goto step** only)			
meet-me-idle[5] (**goto step** only)			
no match[6]			

1. 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.
2. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (+) matches any or no characters at the specified position.
3. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
4. The maximum number of port networks and media-gateways supported 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.
5. This item is available only with meet-me conference vectors.
6. This item is available only with the Dial by Name feature.

goto step and goto vector (*cont.*)						
goto step ***1-99*** **if** **or** **goto vector** ***1-2000***[1] **@step** ***1-99*** **if**						
oldest-call-wait	**in skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**>,>=,<>, =<,<=**	*0-998* sec *1-999* sec *A-Z, AA-ZZ* *V1-V9*
	in split	*hunt group*[2]				
port-network	Port network ID[3] *1-999*		**=, <>**	**registered**		
	all					
	any					
queue-fail[4]						
rolling-asa	**for skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**		**>,>=,<>,=, <,<=**	*0-998 sec'* *1-999 sec* *A-Z, AA-ZZ* *V1-V9*	
	for split	*hunt group*[2]				
	for vdn	*vdn extension*, **latest**, **active**[5]				
server	**=, <>**	**main, ess, lsp**				
service-hours	**in table**		*1-99, [A-Z, AA-ZZ], V1-V9*			
	not-in table					
staffed-agents	**in skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**		**>,>=,<>,=, <,<=**	*0-1499*[1], *1-1500*[1] *A-Z, AA-ZZ* *V1-V9*	
	in split	*hunt group*[2]				

1. 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.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
3. The maximum number of port networks and media-gateways supported 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.
4. This item is available only with the Attendant Vectoring feature.

5. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

<table>
<tr><td colspan="9">goto step and goto vector (cont.)</td></tr>
<tr><td colspan="9">goto step 1-99 if
or
goto vector 1-2000[1] @step 1-99 if</td></tr>
<tr><td>time-of-day</td><td>is</td><td>mon, tue, wed, thu, fri, sat, sun, all</td><td>hour:
00-23</td><td>minute:
00-59</td><td>to</td><td>mon, tue, wed, thu, fri, sat, sun, all</td><td>hour:
00-23</td><td>minute:
00-59</td></tr>
<tr><td rowspan="4">V1-V9</td><td>>, <, =,<>, >=, <, <=</td><td colspan="7">threshold value or string of digits: 1-16, wildcards (?, +), [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>=,<></td><td colspan="7">none[2], #[3]</td></tr>
<tr><td>in table</td><td colspan="7" rowspan="2">1-100[1], [A-Z, AA-ZZ], V1-V9</td></tr>
<tr><td>not-in table</td></tr>
<tr><td rowspan="2">wait-improved for</td><td>best</td><td colspan="4">>, >=, <>, =, <, <=</td><td colspan="2" rowspan="2">0-9998 sec
1-9999 sec
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td>skill</td><td>hunt group[4], skills for VDN:
1st, 2nd, 3rd</td><td>pri</td><td>priorities:
l = low
m = medium
h = high
t = top</td><td>>,>=, <>,=<, <=</td></tr>
<tr><td>split</td><td>hunt group[5]</td><td colspan="4"></td></tr>
<tr><td colspan="9">unconditionally</td></tr>
</table>

1. 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.
2. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
3. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
4. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

<table>
<tr><td>messaging</td><td>skill</td><td>hunt group[1]
1st (VDN skill)
2nd (VDN skill)
3rd (VDN skill)</td><td>for extension</td><td>extension no.
latest[2]
active[2]
A-Z, AA-ZZ
V1-V9</td></tr>
<tr><td></td><td>split</td><td>hunt group[1]</td><td></td><td></td></tr>
</table>

1. 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.
2. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

queue-to	**attd-group**[1]			
	attendant[1]	*extension no.*		
	best			
	hunt-group[1]	*group number*[2]	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op
	skill	*hunt group*[3], VDN skills (**1st, 2nd, 3rd**)		
	split	*hunt group*[3]		

1. This item is available with only the Attendant Vectoring feature.
2. A valid group number is a vector-controlled hunt group of any type (ACD, UCD, and so on).
3. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

reply-best

return

route-to[1]	**digits**	**with coverage**	**y, n**					
	meetme[2]							
	number	*0-9* ***** **#** **~p** **~m** **~s** **~w** **~W** *A-Z, AA-ZZ* *V1-V9* **~r** *<up to 14 digits>*[3] **~r**[*V1-V9*][3] **~r**[*A-Z, AA-ZZ*][3]	**with cov**	**y, n**	**if**	**digit**	**>** **>=** **<>** **=<** **<=**	*0-9* **#**[4]
						interflow-qpos	**<** **=** **<=**	*1-9*
						unconditionally		
	name1[5]	**with coverage**	**y, n**					
	name2[5]							
	name3[54]							

1. The route-to digits and route-to number commands support the Service Observing FACs, remote logout of agent FAC, remote access extension, attendant access number, and other dialable destination numbers.
2. This item is available only with meet-me conference vectors.
3. When the specified number is preceeded by ~r, Network Call Redirection is attempted.
4. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
5. This item is available only with the Dial by Name feature.

set [***vector variable*, Digits] =** [***operand1***] [***operator***] [***operand2***]

Command	Variables or digits		Operand1	Operator	Operand2
set	*user-assigned*[1] type *A-Z or AA-ZZ* vector variable	**=**	*user-assigned*[1] type *A-Z or AA-ZZ* vector variable	**ADD** **SUB** **MUL** **DIV** **CATL** **CATR** **MOD10** **SEL**	*user-assigned*[1] *A-Z or AA-ZZ* vector variable
	asaiuui *A-Z or AA-ZZ* vector variable				
	digits[2]		system-assigned[3] *A-Z or AA-ZZ* vector variable		system-assigned[3] *A-Z or AA-ZZ* vector variable
			V1-V9 VDN variable		directly-entered numeric string[4]
			digits		*V1-V9 VDN variable*
			none		**digits**
					none

1. Only global or local collect type vector variables can be assigned using the set command.
2. The collected digits buffer holds up to 16 digits.
3. For example, ani, asaiuui, doy, and so on.
4. Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.

stop

wait-time	*0-999*	**secs**	**hearing**	**music, ringback, silence, i-silent**		
	0-480[1]	**mins**		*audio source ext.*[2] *A-Z, AA-ZZ* *V1-V9*	**then**	**music** **ringback** **silence** **continue**[3]
	0-8[1]	**hrs**				

1. This option is not available for vector administration done through Avaya Call Management System or Visual Vectors.

2. This consists of a valid announcement or music source extension that is defined on the announcement audio sources form.
3. The continue treatment is valid only with Multiple Audio/Music Sources. It indicates that the caller continues to hear the alternate audio or music source using an announcement until another vector command takes effect.

Vector variables job aid

For detailed information about variable types, see Variables in Vectors on page 107.

Items in bold are default values that cannot be changed.

Variable type	Description	Scope	Specification	Max digit length	Assigns
ani	Tests the caller's phone number	**L**	Start digit position and Length	16	Incoming call data
asaiuui	Processes call-specific user data associated with the call	**L**	Start digit position and Length	16 out of a total of 96	Incoming call or ASAI application data
collect	Processes collected digits for user-defined control, routing, or treatment	L or G	Start digit position and Length	16	The for parameter of the collected digits command or assignment in the variables table
tod	Holds the current time of day in 24-hour time for processing	**G**	None	Always 4	The main server system clock - for example, 0219 = 2:19 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	Counts the number of vector steps executed for the call, including the current step	**L**	None	4	The vector processing step counter

Variable type	Description	Scope	Specification	Max 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 VAC FAC procedure or assignment in the variables table
vdn	Holds the VDN extension number of the call for processing	**L**	Active or Latest	7	Routing for a call
vdntime	Provides the time in seconds that a call has been in vector processing by the call center	**L**	None	Always 4	Time in vector processing including prior processing for a call routed by BSR/LAI

Appendix B: Vector management and monitoring

To manage your vectors, there are several considerations and tasks of which you need to be aware.

This section includes the following topics:

- Implementation requirements for the Call Vectoring features on page 667
- Enabling the Vector Disconnect Timer on page 673
- Upgrading to a Call Vectoring environment on page 673
- Changing and testing a vector on page 674
- Identifying links to a vector on page 675
- Finding all occurrences of a digit string on page 676

Implementation requirements for the Call Vectoring features

The following tables indicate the screens and the hardware required for the following Call Vectoring features:

- Basic Call Vectoring Requirements on page 668
- Call Prompting Requirements on page 668
- G3V4 Enhanced Vectoring Requirements on page 669
- Advanced Vector routing requirements on page 669
- Vectoring (Best Service Routing) requirements on page 669
- ANI/II-Digits requirements on page 670
- CINFO requirements on page 670
- Look-Ahead Interflow requirements on page 671
- Adjunct Routing requirements on page 671
- Holiday Vectoring requirements on page 672
- Network Call Redirection requirements on page 672

- Variables in Vectors requirements on page 672
- VDN variables requirements on page 672
- 3.0 Enhanced Vectoring requirements on page 673

Basic Call Vectoring Requirements

Screens	Hardware
• Vector Directory Number • Hunt Group • Call Vector • Feature Related System Parameters	Announcement capabilities require either: • TN750 Integrated Announcement circuit pack(s), or • External announcement facility (analog announcements). Also, each analog announcement requires a port on an analog line circuit pack or on an auxiliary trunk circuit pack. For a list of available analog circuit packs, see the *Hardware Description and Reference for Avaya Communication Manager*.

Note:
The TN750 Integrated Announcement circuit pack provides 16 ports for listening to announcements. The system provides for the installation of multiple TN750C Integrated Announcement circuit packs.

Call Prompting Requirements

Screens	Hardware
• Vector Directory Number • Hunt Group • Call Vector	Announcement capabilities require either: • TN750 Integrated Announcement circuit pack(s), or • External announcement facility (analog announcements). Also, each analog announcement requires a port on an analog line circuit pack. For a list of available analog circuit packs, see *Hardware Description and Reference for Avaya Communication Manager*.

G3V4 Enhanced Vectoring Requirements

Screen(s)	Hardware
• Vector Directory Number screen • Hunt Group screen • Call Vector screen	Requires no hardware in addition to that required for Basic Call Vectoring.

Advanced Vector routing requirements

Screen(s)	Hardware
• Vector Directory Number screen • Hunt Group screen • Call Vector screen	Requires no hardware in addition to that required for Basic Call Vectoring.

Vectoring (Best Service Routing) requirements

Screen(s)	Hardware
Single-site BSR	
• Vector Directory Number screen • Call Vector screen	No special hardware required for single-site BSR.
Multi-site BSR	
• Best Service Routing Application Plan screen • Vector Directory Number screen • Call Vector screen • Trunk screens	Multi-site BSR requires no special hardware other than ISDN BRI/PRI or SIP connectivity between switches.

ANI/II-Digits requirements

Screens	Hardware
• Vector Directory Number screen • Hunt Group screen • Call Vector screen • Trunk Group screens • Vector Routing Tables screens	Requires no hardware in addition to that required for Basic Call Vectoring.

CINFO requirements

Screens	Hardware
• Vector Directory Number screen • Hunt Group screen • Call Vector screen • Trunk Group screens • Vector Routing Tables screens	Requires no hardware in addition to that required for Basic Call Vectoring.

Look-Ahead Interflow requirements

Screens	Hardware
• Trunk Group screen • CPN Prefix Table screen	Existing hardware can be used for LAI connectivity to the receiving switch. 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 may interflow successfully, but the ability to do so on an intelligent basis will be lost as will the Look-ahead DNIS information. Look-Ahead Interflow calls can connect ISDN-PRI switch-to-switch using private, public, or SDN facilities.

Adjunct Routing requirements

Screens	Hardware
• Hunt Groups • 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 latter provides packet bus control. Also, an adjunct/ host processor must be in place to receive the request and select the route. A TN2198 two-wire BRI port circuit pack can be used in place of the TN556. In this case, an NT1 is also required. **MAPD Connection** MAPD hardware is a sandwich of two boards, the TN801 and a Pentium processor, which allows the switch to be connected to Ethernet and TCP/IP networks. The MAPD requires three contiguous slots on the switch: two slots are occupied by the MAPD unit, and the third is reserved for future use. **Packet Bus** The Packet Bus option (G3r only) must be enabled on the Maintenance-Related System Parameters screen before associated ISDN-BRI screens and fields can be administered.

Holiday Vectoring requirements

Screens	Hardware
• Holiday Table • Call Vector	No special hardware required for Holiday Vectoring.

Network Call Redirection requirements

Screens	Hardware
• BSR • Trunk Group • Signaling • DS1	No special hardware required for Network Call Redirection.

Variables in Vectors requirements

Screens	Hardware
VDN	No special hardware required for Variables in Vectors.

VDN variables requirements

Screens	Hardware
VDN	No special hardware required for VDN variables.

3.0 Enhanced Vectoring requirements

Screens	Hardware
System Parameters Customer Options	No special hardware required for 3.0 Enhanced Vectoring.

Enabling the Vector Disconnect Timer

Call Vectoring provides a Vector Disconnect Timer, which can be set for any amount of time between 1 and 240 minutes inclusive. The timer is enabled by selecting the timer field in the Feature-Related System-Parameters screen. The timer is started when vector processing is started. Once the timer runs out, the call is dropped. The timer is canceled when vector processing terminates.

Enabling the timer allows queued calls that have not been answered within a determined amount of time to be dropped. For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

Upgrading to a Call Vectoring environment

If you are already equipped with ACD and want to use Call Vectoring, the ACD environment must be upgraded to a Call Vectoring environment. This involves installing VDNs, vectors and hunt groups for the desired Call Vectoring feature(s).

The set of guidelines that follows is intended to serve as a general procedure for upgrading to a Call Vectoring environment.

1. Verify the vector options on the Customer Option screen.
2. Add the VDNs.
3. Evaluate the number of queue slots assigned to each split. Usually, you want to assign enough queue slots to allow all calls processed by Call Vectoring to be queued. For more details, see the considerations for Basic Call Vectoring in Appendix C: Considerations for the vectoring features on page 677.
4. Change hunt-groups to be vector-controlled.
5. Administer the vectors and at least one test hunt group.

6. Test all of the vectors to be installed.
7. Change the trunk groups, night destinations, etc., to use the VDNs.

Changing and testing a vector

Vectors currently being used to process calls should not be changed because changes would have an immediate and uncertain effect on the treatment that the calls are receiving. Instead, a new vector should always be written.

In testing the vector, you should not consider the entire vector at once. Rather, you should first figuratively divide the vector into portions, then test each of these portions until the entire vector is tested.

After the new vector is thoroughly tested, the vector should be brought into service by changing the VDN to point to the new vector.

The set of following guidelines is intended to serve as a general procedure for changing and testing vectors.

1. Check that a current version of the translation data is available.
2. Create a new VDN that points to the new vector. This VDN, which is temporary, is necessary to test the new vector.
3. Administer the new vector. Vector commands should be added and tested, one command at a time, starting with the first command. Be sure that each line is correct before proceeding to the next one.
4. Test the new vector with the new VDN. This ensures the new vector will function correctly when the vector is installed.
5. Install the new vector by changing the old VDN’s vector assignment so that the VDNs now point to the new vector. Calls that are already being processed by the old vector will continue to be handled by that vector until the vector terminates vector processing.
6. Once all the calls are handled, remove the old vector and the VDN that was used for testing.

Identifying links to a vector

One or more VDNs always point to a vector. In addition, some vectors are linked to other vectors by **goto vector** commands or by **route-to** commands that point to a VDN. Before you delete or change a vector, you should identify all the VDNs and vectors that will be affected.

The **list usage vector nnn** command finds all the VDNs and vectors that send calls to vector ***nnn***, where ***nnn*** is the assigned vector number.

For example, let's say you want to delete vector 3. To determine what other elements of your system send calls to vector 3, enter **list usage vector 3** and press **Enter**.

The List Usage Report screen is displayed.

```
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, you'll need to change this vector and VDN so they point to a different vector or delete them too.

Finding all occurrences of a digit string

A single extension or an external phone number can be used in several elements in a complex vectoring system. When you modify VDNs or vectors, or when you change the phone numbers used in system elements such as **route-to** commands or Best Service Routing Plans, the switch allows you to find a specific digit string.

1. The **list usage digit-string *(1-16 digits)*** command finds the specified digit string in vectors, vector routing tables, and Best Service Routing Plans. The digit string can contain the numerals 0-9 and the characters *, #, ~, 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 Best Service Routing Plans and steps in four different vectors route calls to this VDN. If you delete this VDN or assign a different extension, you'll need to update the extension used by these system elements.

Appendix C: Considerations for the vectoring features

This section provides various considerations you should bear in mind when using the Call Vectoring features. These considerations are intended to help you get the highest degree of productivity from Call Vectoring. For Look-Ahead Interflow considerations, see Look-Ahead Interflow (LAI) on page 273.

Note:
If EAS is optioned, skill replaces split.

This section includes the following topics:

- Displaying VDN names for vector-initiated DACs on page 677
- Transferring calls to VDNs on page 683
- VDN Return Destination on page 683

Displaying VDN names for vector-initiated DACs

The Display VDN for Route-to DAC feature improves the efficiency of call center agents who answer vector-initiated direct agent calls that originate from multiple Vector Directory Numbers (VDNs).

The type of information displayed at the agent station display with a vector-initiated direct agent call can be summarized as follows:

- When the Display VDN for Route-to DAC feature is *not* enabled, only the EAS LoginID name for the agent who receives the call is shown.
- When the Display VDN for Route-to DAC feature *is* enabled for such calls, the active VDN name associated with the call is shown.

Providing agents with the ability to see the VDN name associated with an incoming call improves agent efficiency and customer satisfaction. For example, if an agent receives incoming trunk calls for different products from three different VDNs, the VDN name displayed by the Display VDN for Route-to DAC feature allows the agent to answer the call as a sales representative of that product. This feature is especially useful when vector-initiated direct agent calls route incoming trunk callers to personalized agent providing services for new customers, special product offers, or premier levels of service.

This section includes the following topics:

- Operations on page 678
- Prerequisites on page 679
- Administering the Display VDN for Route-To DAC feature on page 679
- Creating vectors that use the Display VDN for Route-to DAC feature on page 680
- Interactions with other Communication Manager features on page 682

Operations

The Display VDN for Route-to DAC feature is designed for call scenarios where a VDN-initiated call is routed to a vector where direct agent calls are originated by one of the following methods:

- **A route-to number** vector step with **cov** parameter set to **y**, where the **number** field is administerd with a valid EAS loginID extension.
- **A route-to digits** vector step with **coverage** parameter set to **y**, where a **collect digits** vector step preceding this step is used to allow the caller to enter the digits for an EAS LoginID extension.
- An **adjunct routing link** vector step, where a direct agent call is originated by the Route Select digit information returned from a CTI application.

The Display VDN for Route-to DAC feature is activated for an incoming trunk call when the call is routed through a VDN that has the **Display VDN for DAC Calls?** field administered to **y**. When one of the above-listed vector steps routes such an incoming call as a direct agent call to an EAS loginID extension, the active VDN name is shown on the called agent station display instead of the called EAS agent's LoginID name. If this call is routed to another EAS agent in the initially-called EAS agent coverage path, the active VDN name will again be shown on the covered-to agent station display, instead of the initially-called EAS agent LoginID name.

Station display formats

If the Display VDN for Route-to DAC feature is activated for an incoming trunk call routed through a VDN to a vector that initiates a direct agent call to an EAS agent, the format of the called agent station display appears as one of the following:

```
<Incoming Trunk Name> to <VDN Name>
```

```
<Incoming caller ANI> to <VDN Name>
```

If the Display VDN for Route-To DAC feature is not activated for an incoming trunk call, the called agent station display appears as one of the following:

```
<Incoming Trunk Name> to <EAS loginID extension>
```

```
<Incoming caller ANI> to <EAS loginID extension>
```

Note:
If the EAS agent to which the call is routed by vector-initiated Direct Agent Calling (DAC) is not available, and the called EAS agent has a coverage path to other EAS agents, the Display VDN for Route-to DAC feature preserves the active VDN name and sends it to the agent station display for a covered-to EAS agent. If the call covers to a normal station extension in the called EAS agent coverage path, the Display VDN for Route-to DAC feature does not apply to the covered-to station display, and the EAS LoginID of the called EAS agent is displayed instead.

Prerequisites

To use the Display VDN for Route-to DAC feature for incoming trunk calls routed through a Vector Directory Number to an EAS agent using Direct Agent Calling (DAC), the following administration settings are required:

- The Expert Agent Selection (EAS) feature must be enabled using the System-Parameters Customer-Options screen and the Features-Related System Parameters screen.
- The VDN used to route an incoming trunk call to a vector that initiates a direct agent call must have the **Display VDN for DAC Call?** field set on page 2 of the Vector Directory Number screen. Also, the Class of Restriction (COR) administered for this VDN must have the **Direct Agent Calling** field set to **y** on page 1 of the **Class of Restriction** screen.
- The EAS LoginID to which a vector-initiated direct agent call is routed must have an administered COR that has the **Direct Agent Calling** field set to **y** on page 1 of the **Class of Restriction** screen.

For detailed feature administration instructions, see Administering the Display VDN for Route-To DAC feature on page 679.

Administering the Display VDN for Route-To DAC feature

To activate the Display VDN for Route-to DAC feature, the VDN used to route an incoming trunk call must be administered with the **Display VDN for DAC Calls?** field set to **y** The active VDN name station display treatment provided the Display VDN for Route-to DAC feature applies to the initial EAS agent who receives the vector-initiated direct agent call, as well as any EAS agents who may be in the coverage path of the EAS agent the call is initially routed to.

To enable the Display VDN for Route-to DAC feature:

1. Log in to the switch administration system.
2. Enter:

   display system-parameters customer-options
3. Go to page 5 of the screen.
4. If the **Expert Agent Selection?** field is set to **n**, change it to **y**.

   Important:
   This screen can only be changed by installing a new license file to the communication server. Contact your Avaya or indirect channel account executive for assistance.
5. Enter:

   change system-parameters features
6. Go to page 10 of the screen.
7. If the **Expert Agent Selection (EAS) Enabled?** field is set to **n**, set the field to **y**.
8. Enter:

   change vdn *XXXXX*

   Where ***XXXXX*** is the VDN number for which the Display VDN for Route-to DAC feature is to be enabled.
9. Go to page 2 of the screen.
10. Set the **Display VDN for Route-To DAC?** field to **y**.

Creating vectors that use the Display VDN for Route-to DAC feature

You can administer a vector in several different ways to utilize the Display VDN for Route-to DAC feature.

Note:
For any of the vector examples shown below, if an incoming trunk call is routed through a **VDN with the Display VDN for Route-to DAC?** field set to **y**, the direct agent call is activated with the VDN Display for Route-to DAC feature.

Using collect digits and route-to digits commands

The following vector example shows how to:

- Use a **collect digits** vector step to prompt a caller to enter digits for a valid EAS agent loginID extension
- Use a **route-to digits** vector step to route the call to an agent as a direct agent call:

```
wait-time 0 secs hearing ringback
collect 5 digits after announcement 3001
go to step 5 if digits < > 1????
route-to digits with coverage y
announcement 3002
goto step 2
```

Using route-to number commands

The following simple vector uses the **route-to number** vector step to originate a direct agent call to an EAS LoginID extension:

```
wait-time 0 secs hearing ringback
route-to number 85103 with cov y
```

Using adjunct routing link commands

You can also originate a direct agent call with a vector that includes an **adjunct route vector** step. When an incoming trunk call is routed through a VDN to a vector that includes an adjunct route vector step, vector processing treats this step like a **route-to number with cov** set to **y** vector step.

The following vector uses the adjunct route vector step to originate a direct agent call. In this example, the CTI application would be designed to route the call as a direct agent call in a Route Select ASAI message.

```
1. wait 0 secs hearing ringback
2. adjunct route link 3
3. wait 30 secs hearing ringback
4. announcement 3501
5. disconnect
```

Interactions with other Communication Manager features

Interactions of the Display VDN for Route-to DAC feature with other Communication Manager features include the following:

Call Coverage: When the Display VDN for Route-to DAC feature is activated for a call, and a vector-initiated direct agent call is made to an EAS agent having a coverage path that has other agents as coverage points, the active VDN name associated with the call is displayed on a covered-to agent's station display instead of the originally-called EAS agent's LoginID extension.

Call Forwarding: Display VDN for Route-to DAC has no impact on the Call Forwarding feature.

Station Conference/Transfer: When an EAS agent transfers or conferences a vector-initiated direct agent call that has the Display VDN for Route-to DAC feature activated to another agent or station user, the station display of the answering agent or station does not show the active VDN name that was previously displayed for the call. This is consistent with the existing station display treatment for transferred or conferenced calls that have a VDN name shown as the *to* party for a call.

VDN Override : Active VDN name station display rules for the VDN Override feature are applied to the Display VDN for Route-to DAC feature. For example, if an incoming trunk call is routed through a VDN where the VDN Override feature is enabled, and the call is routed to a second VDN by a route-to number vector step where the **Display VDN for Route-To DAC?** option is set to **y**, the station display for an EAS agent that receives a subsequent vector-initiated direct agent call shows the second VDN's name for the call instead of the called EAS agent's LoginID extension.

Redirect on No Answer (RONA): The Display VDN for Route-to DAC feature is activated only for vector-initiated direct agent call to an EAS LoginID extension. When the RONA timer expires after the call is not answered, one of the following results occurs:

- If subsequent vector processing again routes the call to an EAS LoginID extension by means of the Direct Agent Calling (DAC) feature, and the Display VDN for Route-to DAC feature is enabled, the active VDN name is shown on the covered-to agent station display.
- If subsequent vector processing again routes the call to an EAS LoginID extension by means of the DAC feature, and the Display VDN for Route-to DAC feature is not enabled, then the EAS LoginID for the covered-to agent is shown on their station display.

Messaging systems for EAS Agents: The Display VDN for Route-To DAC feature has no interaction with messaging systems for a vector-initiated direct agent call that is routed to an EAS agent and subsequently covers to the agent's messaging-system mailbox.

Adjunct Routing: If a call is routed through a VDN having the **Display VDN for Route-to DAC?** feature set to **y**, and an **`adjunct route`** vector step is executed that results in a direct agent call to an EAS agent, the active VDN name is displayed on the routed-to agent's station display instead of the called EAS agent's LoginID.

Transferring calls to VDNs

Care needs to be taken when writing a vector to which callers will be transferred. This is especially true if the vector manipulates or tests data that is delivered with the incoming call, such as ANI, II-digits, or CINFO digits.

To understand why care is needed, it is necessary to understand how a transferred call is treated. There are three main steps in a call transfer.

1. The transferring party hits the transfer button. The caller is put on hold. A second call is created with the transferring party as the originator.
2. The transferring party dials the VDN extension. Vector processing starts. The transferring party, not the caller, hears the initial vector provided feedback, if any.
3. The transferring party hits the transfer button for the second time. The two calls merge. The transferring party is dropped from the call. The caller becomes the originator of the new call. The caller now begins to receive vector provided feedback.

Between transfer steps 2 and 3 there is always a small but finite amount of time during which it is the transferring party who is connected to the vector. Any testing of ANI, II-digits, or CINFO digits during this time window applies to the transferring party and not to the caller. For this reason, it is recommended that vectors not start with an ANI, II-digit, or collect cdpd/ced step. Insert a delay of sufficient length to allow the transferring party to complete the transfer.

A delay is not required before a **`collect x digits after announcement`** step because a collect announcement is restarted for the caller when the transfer is complete.

VDN Return Destination

This section includes the following topics:

- About VDN Return Destination on page 683
- User scenario — remote access with host provided security on page 685
- User scenario — saving in trunk facilities between call centers on page 686

About VDN Return Destination

The VDN Return Destination feature allows an incoming trunk call to be placed back in vector processing after all parties, except the originator, drop. This feature is activated through switch administration of the VDN screen. This feature is included in the Avaya Contact Center Deluxe package and the Avaya Contact Center Elite package.

A field on the VDN screen allows the user to enter a VDN extension as a Return Destination. In this section, the VDN which has the Return Destination field administered will be called the **VDN with this feature active**. The Return Destination VDN (the one specified in the new field) will be referred to as the **Return Destination**.

Every incoming trunk call that is processed through a VDN with this feature active will be placed back in vector processing when all parties on the call, except the originator, drop. For this feature, the originator is the incoming party that originated the call at the time the call entered the VDN with this feature active.

Note:
Incoming calls on DCS ties do not go to VDN Return Destination.

The VDN that the call will be placed in (when the originator is the only remaining party) is determined by the return destination. This VDN may be the same or different than the original VDN.

This feature is used to keep the call active and give the caller the opportunity to signal the need for sequence dialing (by entering a #). There are two ways this can happen:

1. When the destination drops on its own (after having answered), the call will go to the Return Destination which will have a **`collect digits`** vector step. This step will try to collect the # sign entered by the caller.
2. When the call is not answered, the caller enters the # to request sequence calling (this # will be collected by the ASAI-Requested Digit Collection feature). This # is reported to the adjunct. The adjunct requests the third_party_drop (or third_party_end_call) for the destination, and at that point the call goes to the Return Destination.

The VDN Return Destination and ASAI-Requested Digit Collection features may be used independently, with the following rules:

1. If there is no ASAI request to collect digits, but a Return Destination is provided: when all parties, except the originator, drop, the switch will route the call with only one party active (the caller) to the Return Destination. At this point, the call enters vector processing for the VDN specified by the Return Destination.

   The caller will keep returning to this same return destination indefinitely until either the caller hangs up or a busy or disconnect vector step is executed. Once a call leaves vector processing for the first time, the return destination will never be changed.
2. If a request is made to collect digits but there is no Return Destination provided: the switch will collect the digits and pass them on to the ASAI adjunct. It will be up to the adjunct to take action. However, if the action taken by the adjunct is to drop one party on the call, the switch will drop the other party as well and clear the call (it cannot retain a call with only one party, if there is no Return Destination for further processing).

User scenario — remote access with host provided security

A customer may use the VDN Return Destination feature to provide a more flexible remote access feature together with host-based call security. The remote user/caller does not have to call back into the switch when multiple destinations need to be reached nor does the caller have to enter his/her identification every time a new destination is desired.

This system consists of three VDN/vector pairs. The first VDN uses the vector shown in The following example.

Sample vector for remote access

```
1. collect 6 digits after announcement 1001 ("Please enter
     your identification number and password followed by # sign")
2. adjunct routing link 12
3. wait-time 6 seconds hearing silence
4. disconnect after announcement 1003 ("We are sorry, but we are
     experiencing technical difficulties at this time, please try
     again later")
```

In this scenario, a remote caller calls into the switch by dialing the first VDN. The vector shown above prompts the caller to enter an identification number and a password that will be passed, using the **adjunct routing link** vector command, to the host for validation. The host can keep track of invalid attempts or decide to de-activate or activate certain identification numbers based on customer set criteria. If the host is not available, the call will be disconnected after an announcement (vector step 4 above).

Sample return destination vector with disconnect

```
1. collect 16 digits after announcement 1002 ("Please enter
     the telephone number of your destination, followed by # sign")
2. adjunct routing link 12
3. wait-time 6 seconds hearing silence
4. disconnect after announcement 1003 ("We are sorry, but we are
     experiencing technical difficulties at this time, please try
     again later")
```

If the ID and password are valid, the adjunct specifies a route to the second VDN, which uses the vector shown above. The switch collects digits for the destination that the caller wants to reach (vector step 1 above). The host receives the number entered by the caller (vector step 2 above) and validates the entered number to check if the caller is allowed to reach the specified destination. If so, the host routes the call to the destination. After the called destination disconnects from a call, the caller can remain on the line to be connected to the Return Destination, which points to the same vector.

Note:

If the ID or password entered at the first VDN is invalid, then the call can be routed to a third VDN. The vector for this VDN (not shown) consists simply of a **disconnect after announcement** step with an appropriate announcement. The invalid call attempt is logged.

The caller, once connected to the Return Destination, can enter a second destination/phone number to connect to. The host performs the same validation on the destination number as in the first destination and routes the call as appropriate (destination entered by caller or alternate destination). Note that the host can also provide reports on all the destinations and times reached by each remote user.

In the Return Destination vector, it is recommended that the first vector command give the caller the opportunity to disconnect from the call rather than immediately routing the call to some destination. If the call was immediately routed and then the caller decided to hang-up, the destination that the call was routed to would ring, alerting the called party, but then no one would be on the line at the other end (this could be confusing to customers, and could be misinterpreted as a problem with the feature). Vector commands such as **wait-time**, **collect after announcement**, and **announcement** can provide the caller with the opportunity to disconnect before the call is routed. As an example, an **announcement** command with the recording, *Please hang-up to end your call, or remain on the line if you wish to place another call*, instructs the caller to disconnect before the call is routed.

User scenario — saving in trunk facilities between call centers

You can also use VDN Return Destination to return a call to a local agent after the call is transferred to a remote destination (call). This eliminates the need for the remote agent to transfer the caller back to a local agent and will save in switch trunk facilities, since each time the call is transferred back to a local agent an additional trunk is being used by the call.

For example, calls can be received at the local call through a VDN that has the return destination administered. These calls are delivered to an agent on the local switch. If the local agent transfers the call to a remote destination (because the caller needed to talk to an agent on the remote switch), the call returns to the Return Destination after the remote switch drops the call. The remote switch agent must inform the caller to remain on the line after they are finished and the remote agent just needs to disconnect from the call (hang up).

The Return Destination for this scenario should include an **announcement** vector command at the beginning to inform the caller to disconnect from the call, if they do not want to be reconnected to an agent on the local switch. A sample Return Destination vector is shown in the following example.

Sample return destination vector with announcement

```
1. announcement 1004 ("Please remain on the line, if you want
     to talk a to another representative")
2. queue-to split 101 pri m
3. announcement 1005 ("All our representatives are busy,
     please wait")
4. wait-time 60 secs hearing silence
5. goto step 3 if unconditionally
```

Appendix D: Troubleshooting vectors

This section serves as a troubleshooting guide for Call Vectoring and includes the following topics:

- Criteria for success/failure of call vectoring commands on page 689
- Unexpected feature operations on page 695
- Unexpected command operations on page 696
- Converse command debugging on page 704
- Tracking unexpected events on page 707
- Vector events on page 709
- Clearing events on page 726
- Global variables can change during processing on page 726

Criteria for success/failure of call vectoring commands

The following table summarizes the success and failure criteria for various vector commands. Before you write or evaluate vectors, it is important to understand the information in this table.

Note:
If EAS is enabled, skill replaces split.

Call vectoring command success/failure criteria	
adjunct routing link	
Fails if any of the following are true: • VDN's COR does not permit routing to the adjunct-supplied destination. • TAC/ARS/AAR code is invalid. • Specified agent is not logged into the specified split for a direct agent call. • Local extension is not in the dialplan. • Invalid number was dialed.	Stop **wait-time** or **announcement** step (if present). Then continue vector processing with the next sequential step.
Otherwise, succeeds.	Route the call and provide feedback.
announcement	
Fails if specified announcement is not administered, not recorded, or busied out.	Continue vector processing with the next sequential step.
Otherwise, succeeds.	Play the announcement, then continue at the next sequential step.
busy	
Always succeeds. Central Office (CO) without answer supervision trunk callers will not hear the busy tone.	Exit vector processing, then play the busy tone for 45 seconds before dropping the call. (Unanswered CO trunk calls receive 45 seconds of ringback.)

Call vectoring command success/failure criteria (continued)	
check split	
Fails if any of the following are true: • Vector conditional is false. • Split's queue is full. • Split is not vector-controlled. • Call is already queued at the specified priority to the specified split. • Call is already queued to three different splits. Otherwise:	Continue vector processing with the next sequential step.
Succeeds, and the call is terminated to an agent.	Exit vector processing, and pass control to call processing.
Succeeds, and the call is queued or requeued in the specified split at the specified priority.	Continue vector processing with the next sequential step.
collect-digits	
Fails if any of the following are true:	
Call originates from an outside caller who is not using a touch-tone telephone.	Call Prompting timer takes effect, command times out, and vector processing continues at the next vector step.
No TTR is in the system, or the TTR queue is full.	Continue vector processing at the next step.
Caller enters fewer digits than the maximum specified.	Call Prompting timer takes effect, command is terminated, and any digits collected prior to the timeout are available for subsequent processing.
Otherwise, succeeds.	Continue vector processing at the next step.

Call vectoring command success/failure criteria (continued)	
consider locations	
Fails if any of the following are true: • No BSR application administered in active VDN. • Location not administered in BSR application. • Status Poll VDN number not administered in BSR application. • Status Poll VDN number is invalid. • Status Poll fails because all trunks are busy.	Continue vector processing with the next sequential step.
Otherwise:	
Succeeds, but takes no action if polling of specified location is suppressed.	Continue vector processing with the next sequential step.
Succeeds, and place status poll call to the status poll VDN.	Suspend vector processing until status poll response received.
consider split	
Fails if any of the following are true: • VDN skill (1st, 2nd, 3rd) is used in consider step but not administered for active VDN.	Continue vector processing with the next sequential step.
Otherwise: Succeeds, and the status of the local split is evaluated.	
converse-on split	
Fails if any of the following are true: • Converse split queue is full. • Converse split is not vector-controlled. • Auto-available split is in effect, and all agents are logged out by Redirection on No Answer (RONA).	Continue vector processing with the next sequential step.
Otherwise: Succeeds, call is delivered to the converse split, and (if administered) digits are outpulsed to the VRU. The caller is connected to the VRU, the voice response script is executed, and (if necessary) digits are outpulsed to the switch.	Continue vector processing with the next sequential step.

Call vectoring command success/failure criteria (continued)	
disconnect	
Always succeeds.	Play the announcement (if specified). Then drop the call.
goto step and goto vector	
Fails if the step condition is not met. Succeeds if the step condition is met.	Continue vector processing with the next sequential step. **goto step** - continue vector processing with the destination step **goto vector** – continue vector processing with the first nonblank step of the destination vector.
messaging split	
Fails if any of the following are true: • Specified split is not a messaging-system split. • Specified extension is invalid. • Messaging split queue is full. • Messaging split is not vector controlled and has no working agents (none logged in or all in AUX work mode). • Communications link with the messaging-system adjunct is inaccessible. Otherwise, succeeds.	Continue vector processing with the next sequential step. Terminate vector processing.

Call vectoring command success/failure criteria (continued)	
queue-to split	
Fails if any of the following are true: • Split's queue is full. • Split is not vector-controlled. • Call is already queued at the specified priority to the specified split. • Call is already queued to three different splits. Otherwise:	Continue vector processing with the next sequential step.
Succeeds, and the call is terminated to an agent.	Exit vector processing, and pass control to call processing.
Succeeds, and the call is queued or requeued in the specified split at the specified priority.	Continue vector processing with the next sequential step.
reply-best	
Fails if any of the following are true: • Incoming call is not ISDN or SIP • Incoming trunk group is not administered for shared UUI or for QSIG Supplementary Service b.	Drop the call.
Otherwise: Succeeds and returns status data of best resource found in consider series.	Drop the call.
return	
Fails if there is no return destination data stored for the call.	Continues vector processing on the subsequent vector step. If this is the last step, the step is treated as a stop step.
Succeeds when there is return destination data.	Returns to the calling vector.
set	
Always succeeds. If there is an invalid assignment, a vector event is generated.	Continues to the next step with an invalid assignment or not.

Call vectoring command success/failure criteria (continued)	
stop	
Always succeeds.	Exit vector processing. Control is passed to normal call processing. Any queuing or treatment in effect remains in effect. Call is dropped if not queued.
wait-time	
Always succeeds.	Connect the specified treatment and pass control to the delay timer. Any feedback is continued until other feedback is provided.

Unexpected feature operations

The following table indicates and explains unexpected operations within Call Vectoring that you may encounter.

Unexpected feature operations	
Customer observations	**Causes**
General Vector Processing	
Vector stuck	10,000 steps executed. No default treatment in the vector.
Audible feedback lasts longer than the delay interval.	Last vector step. Queuing for an announcement. Queuing for a touch-tone receiver for a **collect digits** step.
Look-Ahead Interflow	
Agent receiving phantom call.	Agents on both switches become available simultaneously. Avoid by including at the beginning of the receiving switch vector a short **wait-time** or **announcement** step. Also, use the interflow-qpos conditional (see How enhanced LAI works on page 281).
Remote agent receiving phantom calls when vectoring uses qpos conditional.	Interflow-qpos threshold may be set too low.

Unexpected feature operations (continued)	
Customer observations	**Causes**
No Look-Ahead Interflow attempts accepted.	No trunks. Network failure. Insufficient FRL.
All Look-Ahead Interflow attempts accepted.	Look-Ahead Interflow attempts are interworking off of one of the following: Interworking off of the network Receiving vector not designed for conditional acceptance **`route-to with coverage yes`** command was used to interflow Look-Ahead Interflow not optioned at the receiving switch.
Look-Ahead DNIS name not displayed or no collected digits received	LAI IE or VDN Name (Shared UUI) not forwarding with call. Trunk group settings are not administered to support this data. For more information, see Information Forwarding on page 189.

Unexpected command operations

The following table indicates and explains the unexpected operations the customer may encounter when using the Call Vectoring commands.

Unexpected command operations	
Customer observation	**Cause**
`adjunct routing link`	
Step skipped (that is, default treatment).	Invalid link extension. No trunks available. COR/FRL restricted. Timeout. (Application did not respond within the time specified in the **`wait-time`** command and/or within the time length of the recorded announcement.) Digit string inconsistent with networking translation. ASAI link down. Invalid route destination returned from adjunct.
Busy tone.	Busy local destination has no available coverage points.

Unexpected command operations (continued)	
Customer observation	**Cause**
Network reorder or intercept	The digit string supplied by the adjunct is inconsistent with public network translation.
	The digit string is inconsistent with the networking translation.
Intercept or reorder tone is heard	Vector processing succeeded routing off the switch, but a problem has occurred before routing to its final destination.
All trunks are busy on a quiet system	Two switches are treating each other as a backup switch.
Step skipped	The Port Network (PN) link is down.
	A variable represents an invalid number, such as out of range or null. The variable is assigned the # character and an event is generated.
announcement	
Announcement not heard	The announcement board is not present.
	The announcement is not administered.
	The announcement is not recorded.
	The announcement is being rerecorded.
	All ports are busied out.
	The announcement restore is in progress.
	The link to the announcement circuit pack is down.
Extra delay before hearing announcement	The announcement queue is full.
	All of the integrated announcement ports are busy.
	The analog announcement is busy.
Vector processing stops	The analog announcement does not answer.
Listening to silence after announcement	The announcement is the last step.
Incomplete announcement	The agent becomes available.
	The previous **adjunct routing link** step succeeds.

Unexpected command operations (continued)	
Customer observation	**Cause**
busy	
Ringback heard instead of busy tone.	Unanswered CO trunk.
check	
Call does not enter queue or terminate to agent.	Step condition not met.
check and **queue-to**	
Call does not enter queue or terminate to agent.	Queue length specified on the hunt group screen has been exceeded. Invalid split. Split not vector-controlled. Already queued to three different splits. No queue. Queue or check status indicates space when queue is full due to direct agent calls. Best keyword is used but consider series is not defining best data.
Call apparently answered in wrong order.	Call being requeued at different priority. Call superseded by higher priority call, including direct agent call.
Call is not routed to remote best location.	No trunk available.
collect digits	
Announcement not heard while waiting for digits, but network billing indicates that the call was answered.	Announcement board not present. Announcement not administered. Announcement not recorded. Announcement being rerecorded. All ports busied out.

Unexpected command operations (continued)	
Customer observation	**Cause**
	Announcement restore in progress.
	Dial ahead digit exists.
`collect digits` (continued)	
Collect step and announcement skipped.	TTR not in system.
	Link to PN that has TTR is down.
	TTR queue full.
Delay before hearing announcement.	All TTR ports busy, but space in queue.
	Announcement queue full.
	All integrated announcement ports busy.
	Analog announcement busy.
Vector stuck.	Analog announcement does not answer.
Dial-ahead digits not recognized.	Dial-ahead digits entered prior to first collection step.
	Call has been transferred.
	LAI attempt has been made.
	TTR has been released.
	24 digits have already been provided.
	Call Prompting timeout since the last digit was entered.
Vector processing halted at collect step; announcement heard again upon return.	Call put on hold, transferred, or conferenced.
Insufficient digits collected; call routed to intercept.	Caller dialed # too soon.
	Caller dialed * without reentering correct digits.
	Call Prompting interdigit time-out.
Caller information button denied.	No digits were collected.

Unexpected command operations (continued)	
Customer observation	**Cause**
	Display not in Normal mode.
Collect announcement not heard and first collected digit incorrect.	System does not contain all TN748C Vintage 5 (or later) circuit packs.
Incomplete announcement.	Agent becomes available. First digit dialed.
consider	
Local split/skill best (in Primary vector or Status Poll vector)	If split/skill number is correct, split or skill has no agents logged in, no queue slots available, or all agents are in AUX work.
Remote location is never best	No BSR application plan assigned to Primary VDN. Location number not assigned in application plan. Missing routing number for Status Poll VDN. No vector assigned to Status Poll VDN. Step in Status Poll vector is initializing best data before reply-best step.
A step is skipped	A variable represents an invalid number, such as out of range or null and an event is generated.
converse-on split[1]	
VRU script not executed	Queue full. No queue. Invalid split. Split not vector-controlled. VRU down.
Ani digits not passed	ANI not available.
Qpos digits not passed	Call not queued to a nonconverse split.
No data returned from VRU	No TTRs available.
VRU script terminated prematurely	Agent becomes available. VRU script attempted to transfer the call.
Wait digits not passed	Call not queued or no working agents in splits where call is queued.
disconnect	
Announcement not heard.	Announcement board not present. Announcement not administered. Announcement not recorded. Announcement being rerecorded.

Unexpected command operations (continued)	
Customer observation	**Cause**
	All ports busied out. Announcement restore in progress.
`disconnect` (continued)	
Extra delay.	Announcement queue is full. All integrated announcement ports busy. All analog announcements busy.
Vector stuck.	Analog announcement does not answer.
`goto step`	
Branch is not made to the specified step.	Step condition not met. System time not set.
`goto vector`	
Branch is not made to the specified vector.	Step condition not met.
Vector stuck.	Goto vector with no steps or with all failed steps.
messaging	
Vector stuck (with ringback).	A variable represents an invalid number, such as out of range or null. The variable is assigned the # character and an event is generated.
`messaging split`	
Vector stuck (with ringback).	Extension unknown to the messaging system.
Step skipped, no message left.	Messaging-system link is down. DCS link to the remote messaging system is down. All DCS trunks busy. Queue for messaging-system voice ports is full.
Vector stuck (with busy).	Remote messaging-system link down.
Messages not found.	Message extension is none (message is left for VDN that accessed the vector).

Unexpected command operations (continued)	
Customer observation	**Cause**
Delay before messaging-system answers.	All messaging-system ports are busy, but there is space in the queue.
Busy tone. Step skipped.	Queue for the messaging-system voice ports is full. Split not a messaging-system split anymore.
reply-best	
Status poll VDN/vector not processing any calls	Incoming call not ISDN or SIP. No application plan defined for BSR application. Status Poll VDN routing number missing from or wrong in application plan.
route-to[2]	
Step skipped (that is, default treatment)	Invalid local extension. No trunks available. COR/FRL restricted. Digit string inconsistent with networking translation. Busy local destination (route to digits without coverage and route to number). No digits collected. Step condition not met.
Network reorder.	Digit string inconsistent with public network translation.
Intercept or reorder tone heard	Vector processing succeeded routing off switch, but a problem has occurred before routing to its final destination.
All trunks busy on a quiet system	Two switches treating each other as a backup switch.
set	
A variable or digits buffer is assigned the # character	In an arithmetic operation, the # character signifies an invalid value, an overflow value, or an underflow value. For more information, see Invalid results for arithmetic operations on page 775.
stop	
Call dropped	Call not queued when vector processing stops.

Unexpected command operations (continued)	
Customer observation	**Cause**
wait-time	
Audible feedback longer than delay interval.	Queuing for an announcement or for a TTR. **Stop** command executed.
Audible feedback shorter than delay interval.	Agent becomes available. Previous **adjunct routing link** step succeeds.
Music not heard.	No music port administered. Music source disconnected or turned off.
Alternate audio/music source not heard	Announcement board not present. Audio/Music source not administered. Audio/Music source not recorded. Audio/Music source being rerecorded. All ports busied out. Announcement restore in progress.

1. Refer to the Converse command debugging section later in this section for more details on converse-on command debugging
2. Complete operation details for the route to commands are presented in Appendix I: Operation details for the route-to command on page 767.

Converse command debugging

The following table is intended to help your troubleshooting efforts with the **converse-on** command.

Note:
Refer to Appendix M: Call flow and specifications for converse - VRI calls on page 807 for details on the call flow for converse-VRI calls.

Converse command debugging

Symptom	Cause	Analysis
Placing a call:		
Converse step skipped.	VRU down (RONA).	Vector event.
	Split queue full	Vector event.
Call stuck in converse.	VRU port doesn't answer, RONA not used.	Check split administration.
	VRU down, RONA leaves call in queue.	Check split status.
Data passing:		
First set of digits not collected.	Converse first delay too short.	Check administration.
	No ANI available.	Vector event.
	No digits collected.	Vector event.
	Call not queued (qpos).	Vector event.
	Expected wait time not available	Vector event.
	VRU timed out awaiting first digit.	VRU error log/trace.
	VRU first digit timeout too short.	Check VRU script. Check converse first data delay.
	Faulty hardware.	Diagnostics

Converse command debugging (continued)

Symptom	Cause	Analysis
Second set of digits not collected.	VRU digit count on first prompt in VRU script does not include #.	Check VRU script.
	Converse second delay too short.	Check administration.
	No ANI available.	Vector event.
	No digits collected.	Vector event.
	Call not queued (qpos).	Vector event.
	Expected wait time not available because call is not queued or the splits/skills that the call is queued to are not staffed	Vector Event
	VRU timed out awaiting first digit.	
	VRU error log/trace.	
	VRU first digit timeout too short.	Check VRU script. Check converse second data delay.
	Inter-digit timeout too short on first prompt and collect.	Check VRU script.
	Faulty hardware.	Diagnostics.
Digits incomplete.	Converse data delay too short.	Check administration.
	Faulty hardware.	Diagnostics.
Second set of digits is the same as the first digits passed.	VRUs first prompt timed out.	Check administration.
	Faulty hardware.	Diagnostics.

Converse command debugging (continued)

Symptom	Cause	Analysis
Data return:		
No digits returned to the switch.	Flash not recognized by switch.	VRU error log/trace.
		Check flash timing on VRU.
	Converse data return FAC not administered.	Check administration.
	VRU does not return FAC.	VRU script. Transfer attempt vector event.
	VRU returns incorrect FAC.	VRU script. Transfer attempt vector event.
	Digit timeout during FAC.	Transfer attempt event.
	Converse data return FAC overlaps with other entries in the dial plan	Check dial plan.
	Faulty hardware.	Diagnostics.
Not all digits returned to the switch.	Digit timeout after FAC.	None unless VRU logs being dropped by the switch.
	Overflow of Call Prompting buffer	Vector Event.
	Faulty hardware.	Diagnostics.
Collect announcement not heard.	Too many digits returned by VRU.	Check VRU script.
	Faulty hardware.	Diagnostics.

Tracking unexpected events

You can display unexpected events related to Call Vectoring and Meet-me Conference. When you have corrected each problem, then you can clear events from the error log. An event is an error that results from resource exhaustion, from faulty vector programming, or from incorrect user operation rather than from a switch software error. For example, failures involving the **route-to** command are usually due to an invalid extension entered by the user.

By displaying events, you can diagnose and correct each problem, as indicated by its corresponding event number, and eliminate the need for a technician to make on-site visits to do the same.

The following sections explain how you can troubleshoot by tracking unexpected events and include the following topics:

- Display events criteria on page 707
- Display events report on page 708

Display events criteria

Use the **display events** command to access the EVENT REPORT screen. Use the fields on this screen to specify the event report criteria.

```
display events                                          Page   1 of   1   SPE B
                                EVENT REPORT

      The following options control which events will be displayed.

         EVENT CATEGORY

            Category: meetme

         REPORT PERIOD

            Interval: a      From:   /  /  :   To:   /  /  :

         SEARCH OPTIONS

                           Vector Number:
                              Event Type:
                               Extension: 36090
```

The following table describes the fields used with the **display events** command.

Field	Description
Category	Enter **denial**, **meetme**, **vector**, or **all** to specify the type of event you want to display.
Interval	Select the time period for which you want to display events. Enter **h** (hour), **d** (day), **w** (week), **m** (month), or **a** (all).
From/To	Enter the date and time of day when you want to start and end the search.
Vector Number	Enter a specific vector number to report on. When the Category field is set to **meetme**, this field is ignored.
Event Type	Enter a specific event type to report on. If this field is blank, events for all types are reported.
Extension	Enter a specific extension or VDN to report on. If this field is blank, events for all extensions are reported.

Display events report

After you have entered your report criteria, submit the command by pressing **Enter**. The following screen shows examples of events.

```
display events

                                EVENTS REPORT

Event Event                       Event     Event      First        Last      Evnt
Type  Description                 Data 1    Data 2     Occur        Occur      Cnt

90    Wait step music failed       3/1      2A2      02/12/15:42 02/13/09:40  255
112   Converse no prompt digits    3/2      2A2      02/12/15:42 02/13/09:40  255
56    Call not in queue            8/1      28B      02/12/15:43 02/13/09:40  255
220   EWT call not queued          8/2      28B      02/12/15:43 02/13/09:40  255
150   Invalid hunt group           8/3      28B      02/12/15:43 02/13/09:40  255
56    Call not in queue            8/5      28B      02/12/15:43 02/13/09:40  255
```

The following table describes the information displayed in the event report.

Column	Description
Event Type	Displays a unique number that identifies the type of event that occurred. These are explained in more detail in Vector events on page 709.
Event Description	Displays a brief explanation of the event.

Column	Description
Event Data 1	Displays the following data: ● *<number1>/<number2>* (for example, 12/5), where *<number1>* is the vector number associated with the vector event, and where *<number2>* is the step number associated with the vector event. ● *Split<number>* (for example, Split 89), where *<number>* is the split number associated with the vector event. ● For Meet-me Conference events, this is the port ID of the user associated with the event.
Event Data 2	Displays the following data: ● Additional data encoded as a hex number (for example, 4C). This number serves as a call identifier. If two or more events with an identical identifier occur at about the same time, it can be concluded that the events were caused by the same call. ● For Meet-me Conference events, this is the VDN of the Meet-me Conference used during the event.
First Occur/Last Occur	Displays the date and time the event first occurred and the date and time the event last occurred.
Evnt Cnt	Displays, up to 255, the total number of vector events of this type that have occurred.

Vector events

The following table provides a list of events, the brief description that displays on the screen, and a full explanation of the event.

Event type	Event description	Explanation
1	Call dropped; call not queued at stop step.	Vector processing ended without the call being queued to a split and, as a result, the call cannot be answered. This implies that some default condition was not programmed or that the vector was designed to not always answer the call. Also, call was subsequently dropped.
2	Vector with no steps	The call encountered a vector with no steps administered.

Event type	Event description	Explanation
3	10,000 step executed	This can occur due to the following: Incorrect vector programming (for example, including a series of **goto** steps that point to one another) Excessive repetition of a programmed loop during a single call (for example, recurring announcement-wait loop)
4	Administration change	The administration of this step occurred while the step was being executed. The call flow for this call is unpredictable. Vectors should not be changed while calls are active.
5	Call dropped by vector disconnect timer	The call was still in vector processing when the vector disconnect timer expired. The call dropped.
7	Attd Vec Mismatch-VDN/Vec	There is a mismatch between Attendant Vectoring and Call Vectoring between the VDN and the vector.
9	Attd Vec Mismatch-CR/ Vec	There is a mismatch between Attendant Vectoring and Call Vectoring between the incoming call and the vector.
10	Retrying announcement	During an **announcement** step, a **collect digits** step that contains an announcement, or a **disconnect** step, the announcement was not available, and the announcement queue (if specified) was full. The step is retried at regular intervals.
11	No announcement available	During an **announcement** step, a **collect digits** step that contains an announcement, or a **disconnect** step, the announcement was not available for one of the following reasons: • Announcement was not recorded • Analog announcement was busied out • Integrated announcement board was not installed • Integrated announcement ports were busied out • Integrated announcement was being recorded or restored
20	Call cannot be queued	A **queue-to split**, **messaging split**, or **check split** command failed to queue the call. NOTE: Event types 520, 521, 522 and 541 may be observed for the same call at the same time.

Event type	Event description	Explanation
21	Queued to three splits	The call attempted to queue to four splits. Multiple split queuing allows the call to queue to a maximum of three splits simultaneously. If the call queued to one or more splits, and if it should now be dequeued from those splits and then queued elsewhere, one solution is to route the call to a station (which may be administered without hardware). Once this happens, the call is forwarded to the VDN that controls the next stage of the call.
22	Attd Vec: Cannot requeue	Applies to Attendant Vectoring and indicates that the call is in the attendant queue and another attempt is made to queue the call to an attendant or hunt group, or the call is in the hunt group queue and an attempt is made to queue it to an attendant or too many attempts are made at queueing to the hunt group.
30	No TTR available	A **collect digits** command failed because: • TN744 port was not available • All queue slots were occupied
31	Dial-ahead discarded	Previously entered dial-ahead digits have been discarded using access of an **adjunct routing link**, **converse-on**, **route-to number**, or **messaging split** step.
32	Prompting buffer overflow	The prompting digit buffer already contained the maximum of 24 digits when additional dial-ahead digits were entered by the caller. These additional digits are not stored.
33	ced digits left behind	A **collect ced digits** step collected digits from a UEC IE, and more than 16 digits were sent from the network.
34	cdpd digits left behind	A **collect cdpd digits** step collected digits from a UEC IE, and more than 16 digits were sent from the network
35	ced digits not available	A **collect ced digits** step collected digits from a UEC IE, and no digits were sent from the network, or no digits were present in the UEC IE.
36	cdpd digits not available	A **collect cdpd digits** step collected digits from a UEC IE, and no digits were sent from the network, or no digits were present in the UEC IE.

Event type	Event description	Explanation
37	collect digits for variable error	• Failed to put the local variable value in the local linked list of collect variables for the call. This implies that the system variable limit was reached. • Failed to put the global variable value in the Variables for Vectors table due to messaging issue with the switch. • Unknown or invalid variable type defined in the collect vector step.
38	Variable not defined	• The variable conditional that is tested is not defined in the Variables for Vectors administration table. • A command, or the messaging extension contains a variable with an invalid value of **none** or **#**.
39	Invalid table number	A variable used as a table entry had an invalid assignment.
40	Messaging step failed	A messaging step failed because the Messaging Adjunct was not available. NOTE: Event types 540 and 541 may be observed for the same call at the same time.
41	Messaging ext invalid	The messaging extension contains a variable with an invalid value of **none** or **#**. Vector event 38 is also generated.
50	Route -to step failed	A **route-to** step failed to reach the intended destination. NOTE: Event types 51 and 52 may provide more specific information regarding the reason for the failure. For more information, see Appendix I: Operation details for the route-to command on page 767.
51	No digits to route-to	The **route-to digits** step was unable to route the call because the previous **collect digits** step failed to collect any digits. This could result from an error in vector programming (for example, a **route-to digits** step appears without a preceding **collect digits** step). More often, however, this results because the caller was unable to enter the required digits (that is, the caller was using a rotary telephone), or because the caller was not provided with enough information to do so (as can be the case for auto-attendant applications).
52	No available trunks	A **route-to** command was unable to reach the specified off-switch destination due to a lack of available trunks.
53	Route-to step failed	The step was unable to seize a trunk because of a hardware problem or glare.

Event type	Event description	Explanation
54	LAI retry	Look Ahead Interflow **route-to** step failed because of glare. The route will be retried once.
55	Double coverage attempt	Coverage option on route-to step was ignored because double coverage is not allowed. This may happen when the call has covered to a VDN.
57	Deny vector-initiated MSO	The vector cannot add an observer because SRVOBS_MAX=2 has been reached.
58	Deny observing observer	The vector cannot cannot observe this agent because this vector is already an observer.
59	Variable Invalid Value	The adjunct route link ID for GAZ (1-8) or MIPSLX (1-64) is invalid.
60	Adjunct route failed	An adjunct route failed for one of reasons indicated in event types 61 through 66.
61	Invalid destination	The **adjunct routing link** command returned digits that did not represent a valid destination.
62	Adjunct route cancelled	The **adjunct routing link** step was cancelled because another routing step (such as a **queue-to split** step) was encountered in the vector.
63	Queue before route	The **adjunct routing link** command was skipped because the call had already been queued using a **queue-to split** or a **check split** command.
64	Adjunct link error	The **adjunct routing link** command was cancelled for one of the following reasons: • Link to the adjunct was down • ASAI protocol violation prevented the call from completing • Software resources to complete the call were unavailable
65	Agent not logged in	A direct agent call was made to an agent who was not logged into the relevant split. Used for adjunct routing request only.
66	Agent not member of split	A direct agent call was made to an agent who is not a member of the relevant split. Used for adjunct routing request only.
67	Invalid direct agent	A direct agent call was made to an agent extension that is not valid. Used for adjunct routing request only.

Event type	Event description	Explanation
68	Adj route via NCR failed	NCR routing failed and a tandem trunk-to-trunk routing could not be done.
69	Adj rte fail-link not adm	The adjunct route link ID is within range but not administered.
70	Busy step for CO trunk	A CO trunk call reached a **busy** step in a vector without having previously received answer supervision. As a result, the caller continues to hear ringback rather than the busy tone.
80	Time not set	A **goto** step with a **time-of-day** conditional was processed, but the switch time was not set.
81	No digits collected	No digits were collected and a comparison was requested against a digit string or in-table. The comparison test was considered false and the next step in the vector was executed.
83	Service-hours table empty	The service-hours table is empty. The table must be administered.
90	Wait step music failed	A **wait-time** step with music was accessed, but the music was not connected. Music may not be administered correctly.
91	Wait step ringback failed	A **wait-time** step with ringback was accessed, but the ringback was not connected.
100	Redirect unanswered call	The call was sent to an agent using a vector, but, due to the Redirection on No Answer (RONA) feature, the call was redirected from the ringing agent.
101	Redirect of call failed	The call was sent to an agent using a vector, but, due to the Redirection on No Answer (RONA) feature, the call was redirected from the ringing agent. The call could not be redirected.
110	Converse no ANI digits	On a **converse-on** step with passing type **ani**, no information was available to populate the field.
111	Converse no qpos digits	On a **converse-on** step with passing type **qpos**, no information was available to populate the field.
112	Converse no prompt digits	On a **converse-on** step with passing type **digits**, no information was available to populate the field.
113	Converse drop during data	On a **converse-on** step, the converse agent hung up while data was being passed. This may indicate a port failure.

Event type	Event description	Explanation
115	ASAI transfer converse	ASAI attempted a transfer of a call that was active at a **converse** step. The transfer failed, and vector processing continued at the next vector step.
116	Converse transfer denied	A transfer of a call that was active at a **converse-on** step was attempted. The transfer either failed or was denied, and vector processing continued at the next vector step.
117	Agent drops converse	While active on a **converse-on** step, an agent became available in a split associated with a **queue-to split** or **check split** step. The call was delivered to the nonconverse agent, and the converse agent was dropped.
125	Data return no digits	On a **converse-on** step, the converse agent activated data return but did not return any digits.
126	Data return timeout	On a **converse-on** step, the converse agent activated data return but timed out while waiting to return digits. Vector processing continued at the next vector step.
140	Coverage conference denied	Coverage to a VDN in a coverage path was denied because more than one party was active on the call.
150	Invalid EAS hunt group used in the vector step	Either the skill hunt group was removed or the skill hunt group became a non-ACD hunt group.
151	Skill indirection used improperly	Either no VDN skills are administered or the vector command has skill indirection and EAS is not enabled.
160	No vector steps, ANI sent	ANI was sent to the CMS for a call that reached a VDN that accessed a vector with no steps defined.
161	uui sent to CMS, but there were no steps in the vector	A call was directed to a VDN associated with a vector that has no steps.
170	ASA - invalid VDN	A **check** or **goto** test requested a comparison of ASA for a VDN that had been removed since the vector was programmed. The comparison test was considered false and the next step in the vector was executed.
200	ANI not avail - digits	A **goto** test requested a comparison of ANI against a digit string and ANI was not available for the call. The comparison test was considered false and the next step in the vector was executed.

Event type	Event description	Explanation
210	Routing table not assigned	A **goto** test requested a comparison with a vector routing table that is not assigned or had been removed since the vector was programmed. The comparison test was considered false and the next step in the vector was executed.
211	No entries in routing table	A **goto** test requested a comparison with a vector routing table that has no entries. This is considered as a non-match.
212	ANI not avail - table	A **goto** test requested a comparison of ANI against in-table and ANI was not available for the call. The comparison test was considered false and the next step in the vector was executed.
213	No digits in variable	In-table is administered, but the variable does not contain any digits on which to search.
220	EWT call not queued	A **goto** test for a call or converse data passing requested EWT for a call not in queue. In this case, the wait time was assumed to be infinite and the comparison was based on EWT > largest possible threshold.
221	EWT not sent to VRU	The EWT wait time for the call was not sent to the VRU for a **converse-on passing wait** vector step because the call was not queued or the splits/skills that the call was queued to were unstaffed.
222	System clock change	The system clock was changed, therefore any calculations involving time (i.e., ASA and EWT) will be inaccurate.
230	II-digits not avail - digits	A **goto** test requested a comparison of II-digits against a digit string and II-digits were not available for the call. The comparison test was considered false and the next step in the vector was executed.
231	II-digits not avail - table	A **goto** test requested a comparison if II-digits against in-table and II-digits were not available for the call. The comparison test was considered false and the next step in the vector was executed.
240	No agent strategy found in VDN	The active VDN for the call, as determined by VDN override, did not have a BSR Available Agent Strategy.
251	Call is not incoming ISDN	Occurs when a **reply-best** command in a status poll vector receives and tries to process a non-ISDN call. Processing in the status poll vector terminated is without a reply being sent.

Event type	Event description	Explanation
261	No best location found	A **queue-to best**, **check-best**, or **reply-best** command failed because the call vector was unable to calculate a best value or because no local best existed. Vector processing continues at the next step. Vectors in multi-site BSR applications won't attempt to interflow calls in this situation.
262	Look-Ahead Interflow attempt failed	Interflow of the call failed: no trunk was available, LAI denial, or some other problem. Vector processing continues at the next step. In BSR applications, polling of this resource is temporarily suppressed.
271	No BSR app num in VDN	A **queue-to best**, **check-best**, or **consider location** command failed because the active VDN for the call as determined by VDN override has no BSR application number assigned. Processing continues with the next vector step. Only occurs in multi-site BSR applications.
272	No BSR application plan administered	A **queue-to best**, **check best**, or **consider location** command failed because the application number assigned to the active VDN does not have an application plan assigned. Processing continues at the next step.
273	Location not on BSR screen	A **consider** command failed because it refers to a location number that is not in the BSR Application screen assigned to the active VDN. Vector processing continues at the next step.
274	Status Poll VDN field is blank	A **consider** command failed because the entry for this location on the BSR Application screen does not contain a routing number for the status poll VDN.
275	Interflow VDN field is blank	A **queue-to best** or **check-best** command failed because the entry on the BSR Application screen for the relevant location does not contain a routing number for the interflow VDN.
276	Agent Status Info Invalid	A **consider location** command failed because the status poll returned invalid data for an available agent (AIT, skill level, or occupancy is missing or out of range). Vector processing continues at the next step. Polling of this location is temporarily suppressed.
277	BSR Status Info Invalid	A **consider location** command failed because the status poll returned invalid EWT data. Vector processing continues at the next step. Polling of this location is temporarily suppressed.

Event type	Event description	Explanation
278	No BSR Data in Response	A **consider location** command failed because the status poll did not return data in the DISCONNECT message. Vector processing continues at the next step. Polling of this location is temporarily suppressed.
279	No response from status poll	A **consider location** command failed because the status poll did not respond within the time allowed or because the status poll could not be performed. Vector processing continues at the next step. Polling of this location is temporarily suppressed.
280	Bad resp from status poll	A **consider location** command failed because it received an invalid response from the status poll such as an LAI acceptance message (such as ALERT or CONNECT). Vector processing continues at the next step. Polling of this location is temporarily suppressed.
281	BSR EWT is infinite	A **consider** command failed because the EWT for the referenced split or skill is infinite. This may be because all agents are logged out or in AUX work, or because no queue slots are available. Vector processing continues at the next step. Polling of this location is temporarily suppressed.
282	BSR status poll attempt failed	A **consider location** command failed because the status poll attempt failed. See other events for the specific reason. Vector processing continues at the next step. Polling of this location is temporarily suppressed.
283	BSR poll no trunks	A **consider location** command failed because there were no available trunks. Vector processing continues at the next step. Polling of this location is temporarily suppressed.
284	BSR poll seize fail	A **consider location** command failed because the status poll was unable to connect to a trunk due to a hardware problem. Vector processing continues at the next step. Polling of this location is temporarily suppressed.
285	BSR poll glare retry	The first status poll attempt for a **consider location** command was unable to connect to a trunk due to a race condition (the same trunk being seized for the outgoing call had an incoming call from the remote end). This status poll will be attempted once more. A second attempt failure will result in event 282.
287	Invalid status polling destination	An attempt was made to perform BSR polling over ISDN without B-Channel over a tandem trunk configuration that combines QSIG TSCs and AT&T TSCs (this type of interworking is not supported by Avaya's ISDN protocol).

Event type	Event description	Explanation
288	BSR Poll: TSC not administered	The trunk group screen does not contain a trunk member administered for purposes of TSC.
289	BSR: Adjust-by invalid	The **consider location adjust by** command contains a variable with an invalid value of **none** or **#**. Vector event 38 is also generated.
291	BSR: Location invalid	The **consider location** command contains a variable with an invalid value of **none** or **#**.
291	No AITCI storage left	No longer used.
292	Data dropped by other app	The network does not support the transport of all user data, so some user data was not sent. You can prioritize the user data using the Shared UUI Feature Priorities page of the Trunk screen. For more information, see Information Forwarding on page 189.
293	No room for reply-best information	The network or shared trunk setting does not support the transport of all data for the best resource. This is unlikely under normal circumstances since only 12 bytes of user information are required. Also see event 298.
294	No room for in-VDN time	The network does not support the transport of all user data. You can prioritize the user data using the Shared UUI Feature Priorities page of the Trunk screen. For more information, see Information Forwarding on page 189.
295	No room for collected dgt	
296	No room for VDN Name	
297	No room for Other LAI	
298	Reply-best got bumped	The network or shared trunk setting does not support does not support the transport of all data about the best resource. (No other applications share user data included in a DISCONNECT message.)
299	In-VDN time got bumped	The network does not support the transport of all user data. You can prioritize the user data using the Shared UUI Feature Priorities page of the Trunk screen. For more information, see Information Forwarding on page 189.
300	Collected dgts got bumped	
301	VDN Name got bumped	
302	Other LAI got bumped	
303	Block: send reply-best	The transport of the best data for a **reply-best** command was denied because the trunk group is neither Supplementary Service b or Shared UUI.

Event type	Event description	Explanation
304	No enhanced info is sent	During the execution of a **queue-to best** or **check best** step, information forwarding transport over this trunk was denied because the trunk group is neither Supplementary Service b nor Shared UUI. This event is not logged for LAI (for example, in execution of a **route-to** step) in order to permit backward compatibility. For more information, see Unexpected feature operations on page 695 as well as Information Forwarding on page 189 and Appendix E: Advanced multi-site routing on page 729.
305	A BSR local treatment vector pulled a remotely queued call back to the local switch to route it elsewhere	If a **queue-to best** step is followed by steps that use any commands other than **announcement**, **wait**, or **go-to**, the trunk to the remote queue is dropped. This functionality can be exploited to allow the local server to take back calls that are interflowed to a remote location after a specified time limit is exceeded. To implement this strategy, a **wait** step with a specified time interval is included in the interflow vector on the local server, followed by one or more **route-to** steps that redirect the call to an alternate call center locations.
310	NCR: Invoke trunk not ISDN	Check that only ISDN trunks are executing the vector steps where NCR is being invoked.
311	Bad NCR trunk admin	Check that all Trunk screen and Signaling Group screen fields related to the NCR feature are correct.
312	NCR: No NCT service	Check that the service provider has activated the NCT feature for the trunk being used for NCT call redirections.
313	NCR: No NCT outgoing trk	Check that the trunk group is administered as a two-way trunk group and that the Usage Allocation settings for the trunk have been set up correctly.
314	NCR: NCT outgo trk drop	Shows that the second leg of the NCT call has been dropped due to a trunk hardware problem, or that a vector step has been executed that returned and ISDN DISCONNECT message (such as a **busy** vector step).
315	NCR: PSTN NCT invoke err	The PSTN switch has not accepted the NCT invocation attempt. Check that the PSTN network switch complies with the NCT standards.
316	NCR: NCT netwrk err	The switch has accepted the NCT invocation attempt, but has rejected it due to some error condition within the network switch. Check that the **Network Call Redir** field on the Trunk screen is administered correctly. Make a request to the service provider for troubleshooting assistance.

Event type	Event description	Explanation
317	NCR: Used NCT trk-to-trk	NCT has not been successfully invoked, but the incoming call is still active as a switch trunk-to-trunk connection (this is only an informational message).
318	NCR: No NCD service	Check that the service provider has activated the NCD feature for the trunk being used for NCD call redirections.
319	NCR: invalid nmbr	The switch has detected that the number used for the NCR invocation that was administered in the *~r* **route to number** vector step or in the BSR Application Table's **VDN Interflow Number** field is an invalid number (the correct number used through switch administration).
320	NCR: NCD call connect err	The vector step has been executed before the vector step invoking NCD that sends an ISDN CONNECT message to the PSTN.
321	NCR: PSTN NCD invoke err	The PSTN has not accepted the NCD invocation attempt. Check that the PSTN network switch complies with the NCD standards. Make a request to the PSTN service provider for troubleshooting assistance.
322	NCR: netwrk err	The switch has accepted the NCD invocation attempt, but has rejected it due to some error condition within the network switch. Make a request to the service provider for troubleshooting assistance.
323	NCR: PSTN NCD max redirs	The PSTN has detected that the call has been redirected by NCD more that the public network maximum number of call deflections limit will allow. Modify vector processing to reduce the number of NCD attempts.
324	NCR: PSTN NCD no disc	The PSTN switch has not disconnected the trunk after performing the NCD or NCT call redirection. Make a request to the PSTN service provider for troubleshooting assistance.
325	NCR: Internal system err	The switch problem with call processing for the NCR invocation attempt. Alternately, for NCT, the first vector step at the redirected-to endpoint is possibly not programmed with a call treatment vector step such as **wait hearing ringback**, **wait hearing music**, or **announcement**. Avoid the use of a vector step such as **wait hearing silence** or **wait hearing i-silence** for the first vector step at the redirected switch endpoint.

Event type	Event description	Explanation
326	No ETSI ECT linkID	The PSTN switch has returned a FACILITY message to the local communication server that includes the following reject component: LinkIDNotAssignedByNetwork. In this case, the local communication server leaves the calls in a trunk-to-trunk transfer state.
327	NCR: Caller not SIP trk	Check that only SIP trunks are executing the vector steps where NCR is being invoked.
350	No return destination	The **return** command failed and continues to the next step because no return destination data exists for the call.
351	Results Truncated	A **set** command executed with operator CATL or CATR. The result was truncated because it was higher than 16 digits.
352	Negative Result	A **set** command attempted to execute. A negative result was converted to # (underflow) during the processing.
353	Divide by Zero	A **set** command attempted to execute with operator DIV. The operation specified by operand 2 divided by zero and resulted in a # (underflow) assignment.
354	Assignment not allowed	A **set** command attempted to execute. The assignment field contains an invalid system-assigned variable. The variable is invalid because it is not a user-assigned variable or a digits buffer.
355	Can't set, no lcl var	A **set** command attempted to assign a value to a user-assigned variable when the 8000 system limit was reached.
356	Return destination stack error	A **goto vector** command was executed with a full return destination stack for the call. The return destination could not be saved.
357	Operand Overflow Underflow	A **set** command attempted to execute with operator ADD, SUB, MUL, or DIV. One of the operands has a # value or a value greater than 4294967295.
358	Overflow Error	A **set** command executed and obtained one of the following results: • A value greater than 4294967295 with a ADD or MUL operator • A number assigned to a variable from an arithmetic operation has exceeded the length definition For example: **set A = none ADD 1000** If variable **A** is defined as having a length of 3, **A** is set to # and this vector event is generated.

Event type	Event description	Explanation
520	Split queue is full	A **queue-to split**, **check split**, or **messaging split** command was executed, but the call did not queue to the split because the queue (if administered) was full. To prevent this condition, use a **goto step...if calls queued in split...>...** before each **queue-to split** or **check split** step so that an alternative treatment may be provided for these cases.
521	Not vector-controlled	The split accessed by a **queue-to split** or **check split** command is not vector-controlled. As a result, the step is skipped.
522	AAS split cannot queue	A **queue-to split**, **check split**, or **messaging split** command was executed on an auto-available split (AAS), but the call did not queue to the split because all the agents were logged out by Redirection on No Answer (RONA).
540	AUDIX link down	Messaging system could not be accessed using a **messaging split** command because the messaging-system link was down. As a result, the step is skipped.
541	Not a messaging split	The split administered for the **messaging split** command is not a messaging split (that is, it does not have a messaging type administered). As a result, the step is skipped.
542	Can't connect idle agent	The call at the head of the queue can't be connected to an idle agent.
550	ASA - No staffed agents	A **check** or **goto** test requested a comparison of ASA for a split/skill that has no staffed agents. The comparison was based on ASA > largest possible threshold.
560	EWT no history for split	A **goto** test requested EWT for a split/skill that has not yet acquired history. The wait time in this case is assumed to be the default value.
561	EWT no split queue	A **goto** test requested EWT for a split/skill that has no queue. The wait time is assumed to be infinite. The comparison was based on EWT > largest possible threshold.
562	EWT split queue full	A **goto** test requested EWT for a split/skill whose queue is currently full. The wait time is assumed to be infinite. The comparison was based on EWT > largest possible threshold.

Event type	Event description	Explanation
563	EWT split no working agents	A **goto** test requested EWT for a split/skill that has no agents logged in or all logged in agents are in the AUX work mode. The wait time in this case is assumed to be infinite and the comparison was based on EWT > largest possible threshold.
564	EWT split locked	A **goto** test requested EWT for a split/skill that is currently locked. The wait time is assumed to be infinite. The comparison was based on EWT > largest possible threshold.
565	EWT call no working agents	A **goto** test for a call or converse data passing wait requested EWT for a call that is queued only to splits/skills that have no agents logged in or that have all logged in agents in AUX work mode. In this case, the wait time was assumed to be infinite and the comparison was based on EWT > largest possible threshold.
1760	Conference COR restrict	Check authorization on calling and called parties for non-PCOL calls.
2034	Illegal TSC interaction	A BSR polling over ISDN without B-Channel attempt has resulted in an illegal TSC interaction. Either an AT&T TSC was routed to a QSIG interface, or vice versa. The call is dropped and the denial event is logged.
2035	NCA-TSC not available	A BSR polling over ISDN without B-Channel attempt has been denied for one of the following reasons: • The terminated administered TSC endpoint is disabled • The incoming nca-tsc call arrives at the wrong signaling group • The max number of nca-tsc is set to 0.
2075	Var-in-vec COS restricted	The station that is attempting to change the value type variable with a Facility Access Code (FAC) does not have console permission.
2404	Var-in-Vec No adm for VAC	There is no Variable Access Code (VAC) administered for the variable in the Variable for Vector Table.
2405	Var-in-Vec Invalid digit	While attempting to change the value type variable to a new assignment, an invalid DTMF digit (for example, #), was entered. You can only enter digits 0-9 or *.
3201	Meet-Me Access chg TMO	The user changing the access code allowed the call to timeout to intercept treatment. The access code was not changed.
3202	Invld Num Digits MM Acc	The user changing the access code entered too many digits. The access code was not changed.

Event type	Event description	Explanation
3203	MM Extension not valid	The user changing the access code did not enter a valid extension.
3204	MM Access Chg Not a VDN	The user changing the access code entered a non Meet-me Conference VDN extension.
3205	MM Invalid Access Entered	The user changing the access code did not enter the correct access code. The access code was not changed.
3206	MM Access Obj/SAT Busy	An administrator is making changes to the Meet-me Conference VDN, so the user cannot change the access code using a feature access code. Try again later.
3207	Merge Meet-me Conf call	A user tried to access an existing Meet-me Conference call and was denied.
3208	Serv Observ Meet-me VDN	A user tried to service observe a Meet-me Conference call. This is not allowed.
3209	Meet-me Conf call full	A user tried to access a Meet-me Conference call that was already full.
3210	Wrong MM Acc. code dialed	A user trying to access a Meet-me Conference call dialed the wrong access code.
3211	Chg Station no Cons/ Perm	The station attempting to change the access code does not have console permissions COS.
3212	VDN not a meetme type	The VDN that was called is not a Meet-me Conference VDN.
3213	MM Invalid Conf Ctrlr Sta	If controlling extension is filled in and the station and controller do not match.
3214	MM Inv Trk not Remote Acc	The trunk used to access the Meet-me Conference is not a remote access trunk.
3215	MM Invalid Station Type	If controlling extension is blank and the station type is invalid (for example, and attendant console).
3216	Conf/Transfer 2 Meet-me	A user cannot conference or transfer another call into a Meet-me Conference call.
3217	MM Abbrev Dial Invalid	When changing a Meet-me Conference access code, the only entry that can be set up for Abbreviated Dialing is the feature-access-code (FAC). Any other entry generates the vector event.

Clearing events

When you have finished your review of the event log, you can remove events from the error log. You must use a super user login ID to clear events.

To clear events from the error log, enter **`clear events`** at the command prompt and press **ENTER**. This command clears all events from the event buffer space within the error log. It does not delete any other entries in the error log.

Global variables can change during processing

The collect global vector variable value is susceptible to being unintentionally changed and read by different vectors being processed for multiple calls - especially during high traffic periods. This can result in unexpected behaviors, such as callers hearing the wrong announcement.



CAUTION:
Global vector variables are accessible by all calls currently in vector processing and are susceptible to be overwritten by vectors associated with other active calls.

It is good programming practice to copy a global variable to a local variable before using it in a vector. This secures a snapshot of the global variable value that is used for subsequent vector processing.

Example:

1. Use A as a global collect type variable.
2. Define Z as a local collect type variable. Use Z as the scratch pad variable to get the current value of A. The variable Z can be used for testing the value obtained from A later in the vector.
3. Use the following command in the beginning of the call processing vector program:

   ```
   set Z = A ADD none
   ```

4. Use the following command when the testing the value of A is required later in the vector:

   ```
   goto step 20 if Z = 123
   ```

Also, if you are modifying the value of a global variable, it is important to complete the manipulation of the global variable within fifteen steps. This is due to vector operation that temporarily suspends vector processing for 0.2 seconds after processing fifteen steps under certain conditions. Therefore, the time period during which the value of a global variable can change could be greater than expected.

For more information, see Additional information about the collect variable on page 132.

Appendix E: Advanced multi-site routing

This section supplements the Look-Ahead Interflow (LAI) and Best Service Routing (BSR) sections.

This section is intended for users whose call center networks meet either or both of the following criteria:

- Five or more switches in the network
- Combination of low- and high-volume locations

This section includes the following topics:

- Application architecture in multi-site BSR on page 729
- User adjustments on page 730
- Status polling in BSR on page 731
- Efficient polling patterns in large networks on page 735
- Considerations for low volume splits/skills on page 738

Application architecture in multi-site BSR

Multi-site applications may be structured in a variety of ways. In general, however, most applications will fit one of two models: distributed or centralized. When each switch in a network may interflow calls to other switches and receive interflows, this is called a distributed system. A centralized system, by contrast, is one in which all calls are initially delivered to a single call center (the hub) and distributed from this site to queues at remote switches. A centralized system requires greater inter-switch trunking, since a greater percentage of calls need to be redirected. However, it may be an appropriate configuration if your organization has a significant investment in VRU and CTI technology at the hub.

Which architecture you choose for an application has direct implications for your choice of user adjustments and polling patterns.

User adjustments

User adjustments in **consider split** and **consider skill** steps may be set at the user's discretion. In distributed multi-site applications, however, adjustments must be carefully considered because of their potential affect on costs and inter-switch trunk capacity. In centralized applications all calls are redirected anyway so it's OK to use adjustments of 0. In distributed applications, though, a user adjustment of 0 for a **consider location** step is almost never practical or efficient.

In distributed applications, the smaller the adjustment the closer the load balance across the network, but the greater the percentage of calls redirected between switches (and thus the greater the demands on inter-switch trunking). Higher adjustments reduce interflows, but at the cost of allowing greater imbalance in the load between switches. It will take some time and effort to find the best combination of user adjustments in any particular network, but Recommended initial user adjustments on page 730 contains recommended ranges for initial user adjustments under different conditions. Adjustments may vary between different call center applications so apply these guidelines for each of your applications separately.

Recommended initial user adjustments

Recommended adjustments…	If the following criteria apply…
10-15	• You want to balance wait times across the network as much as possible. • Trunk facilities between switches are plentiful. • Each switch receives more than 1 call every 10-15 seconds (more than 240-360 calls/hour) for this application.
20	• Balancing wait times across the network is important to you. • Adequate trunk facilities are available to support the desired balance. • Each switch receives more than 1 call every 20 seconds (more than180 calls/hour) for this application.
30 or higher	• Gains in agent efficiency are more important to you than balancing wait times across the network. • Trunk facilities are scarce. • Call interflow is costly. • Each switch receives no more than 1 call every 30 seconds (around 120 calls/hour or lower) for this application.

In your first multi-site application, it is recommended that you begin with a remote adjustment of 30. This can easily be reduced later if inter-switch trunking is under-utilized. On the other hand, if trunk exhaustion is a common occurrence then user adjustments are probably set too low. Care should be taken not to lower remote user adjustments to such an extent that all trunk resources are regularly exhausted. When trunks are exhausted, no further load balancing can take place and the overall balance may deteriorate.

User adjustments should also be set high enough that calls are not interflowed to gain the equivalent of a fraction of a queue position. The following equation will give you the minimum recommended user adjustment for each remote switch:

$$\frac{\text{AverageCallHandlingTime}}{\text{NumberOfFullTimeEquivalentAgents}} \leq \text{UserAdjustment}$$

Adjustments for remote locations will probably be in the range of 10-30 in most distributed applications.

User adjustments and the balance in wait times

Changing conditions can produce significant variations between user adjustments and the balance in wait times across a network, but on average you can predict the balance in wait times for a given user adjustment.

Let's say a user adjustment of 20 is chosen for all remote resources in a network and all the remote sites are polled. When waiting times are short (< 100 secs), the highest and lowest EWTs for this application on the network should stay within a range of approximately 20 seconds (30-50 seconds, for example). When waiting times are long (> 100 secs), the highest and lowest EWTs for the application should stay within a range of approximately 20% (5 to 6 minutes, for example).

Status polling in BSR

This section includes the following topics:

- About status polling on page 732
- How long do status polls take? on page 732
- Intelligent polling on page 734

About status polling

Status polls are the key element in multi-site BSR applications. Status polls provides the communication links between a switch that wants to interflow a call and the switches that might service that call.

The vectors you write in multi-site applications must balance the costs of time and trunk usage with the benefit of better customer service. BSR is designed to help you achieve this balance, incorporating mechanisms to maximize improvements in customer service while minimizing inter-switch communications with its attendant delays and trunk usage. This section explains those mechanisms and the benefits they provide as you write vectors.

How long do status polls take?

One **consider location** step polls one remote location. Does this mean that an optimal multi-site BSR application polls every switch in a network? No.

Let's look at an example of a moderately large network, containing 16 switches. The primary vector on switch #1 could be written as shown in the following vector example. Polling response times are variable. Let's assume that this is a slow response network and that each status poll takes 1 second. The consider series in this vector could add as much as 15 seconds to a call's time in vector processing! In fact, the vector shown below is provided as an example of what NOT to do. The benefits of BSR can be obtained much more efficiently.

Intelligent polling for multi-switch networks

```
1.wait time 0 secs hearing ringback
2.consider skill 1 pri m adjust-by  0
3.consider skill  2 pri madjust-by 20
4.goto step 20 if expected-wait for best = 0
5.consider location 1adjust-by 30
6.consider location 2adjust-by 30
7.consider location 3adjust-by 30
8.consider location 4adjust-by 30
9.consider location 5adjust-by 30
10.consider location 6adjust-by 30
11.consider location 7adjust-by 30
12.consider location 8adjust-by 30
13.consider location 9adjust-by 30
14.consider location 10adjust-by 30
15.consider location 11adjust-by 30
16.consider location 12adjust-by 30
17.consider location 13adjust-by 30
18.consider location 14adjust-by 30
19.consider location 15adjust-by 30
20.queue-to best
21.announcement 1001
22.wait time 60 secs hearing music
23.goto step 21 if unconditionally
```

First, even in very large networks you can obtain nearly all of the possible benefits in agent utilization with very few polling connections. In a network of 16 switches, 99% of the total benefits possible with BSR can be obtained if each switch polls just 4 others. For more information, see How many switches should one switch poll? on page 735.

Now our vector looks like the following. Is polling time now cut from 15 seconds to 4 seconds, proportional to the reduction in **consider** steps?

```
1.wait time 0 secs hearing ringback
2.consider skill 1 pri m adjust-by  0
3.consider skill  2 pri madjust-by  0
4.goto step 9 if expected-wait for call = 0
5.consider location 5adjust-by 30
6.consider location 10adjust-by 30
7.consider location 13adjust-by 30
8.consider location 15adjust-by 30
9.queue-to best
10.announcement 1001
11.wait time 60 secs hearing music
12.goto step 10 if unconditionally
```

In fact, polling time in this vector may be around 0.4 seconds per call because of mechanisms in BSR that constantly react to network conditions and resource usage to minimize the number of status polls. These mechanisms, whose combined operation is called *intelligent polling*, also function to make each status poll as productive as possible.

Intelligent polling

A BSR application will only poll the switches that are likely to provide the best service at any given time. If a remote switch is polled and returns an adjusted EWT greater than that of the current best resource, polling of the remote switch will be suppressed for a period of time proportional to the difference between the two adjusted EWT values. (In other words, polling of a given location is suppressed whenever the adjusted EWT returned by that location is subsequently replaced by a better adjusted EWT from another resource.) The **consider** step for this location will be skipped during this period and vector processing will continue at the next step. When the suppression period is over, the **consider** step will once again poll this location. If the location returns the best adjusted EWT, the next call processed by the vector will also cause this location to be polled. If it is not the best, polling will again be temporarily suppressed, and so on.

If no calls are in queue at the remote location an agent might become available at any moment, and thus BSR will never suppress polling for longer than 5 seconds in such situations. BSR will never suppress polling of any remote location for more than 60 seconds, regardless of the differences between adjusted EWT returned by different switches.

Other conditions can also suppress status polls to a location:

- resource exhaustion (no trunks available, queue full)
- administration errors (badly written vectors, or no application plan)

This feature significantly reduces the average number of status polls placed per call. The greater the call volume, the greater the percentage reduction. Let's take another look at the vector in Screen 2.

Let's assume that the network is operating in a balanced state. EWTs are 30 seconds at all locations, and a call arrives every 3 seconds at each site. Adjusted EWTs are 30 seconds at the origin switch and 60 seconds for each remote switch. After each status poll under these conditions, polling will be suppressed for 30 seconds. Each remote location is polled therefore, by every 10th call. On average, this means that each call polls any one location 0.1 times. Since there are four **consider** steps, each call makes 0.4 polls. Remembering the 1-second polling response time given at the beginning of the example, the average time added to call processing for each call is 0.4 seconds.

The 1st-found available agent strategy, discussed in Best Service Routing (BSR) on page 297, can cut average polling times further. With the 1st-found strategy, BSR will skip all subsequent **consider** steps in a series if a resource with an available agent is found and deliver the call to that resource.

Efficient polling patterns in large networks

Unless you have a small network, you won't benefit by having every switch poll every other switch. This section explains how many remote locations each switch needs to poll, and it provides guidelines for selecting which locations any given switch should poll.

This section includes the following topics:

- How many switches should one switch poll? on page 735
- Which remote switches should each switch poll? on page 736
- Minimizing variations in wait time on page 740

How many switches should one switch poll?

It's not necessary to poll every switch in larger networks. Because of BSR's intelligent polling capabilities, you can obtain 99% of the possible benefits in agent utilization with very few polling connections.

For an example, let's look at a laboratory network of 16 switches that is used for simulations of BSR multi-site applications. As shown in the following table, approximately 99% of the possible benefits were obtained when any one switch polled 4 others.

Effectiveness of status polls in a 16-switch network

Number of remote sites polled by each switch	ASA across the network (seconds)	Approximate percentage of total benefits obtained
0	192.8	0%
1	26.2	89%
2	10.6	95%
3	7.6	98%
4	6.5	99%
15	4.7	100%

For each switch to poll the other 11 switches in the network would only produce an additional 1% gain in ASA and agent utilization—an improvement which would be more than offset by the cost of additional messaging and trunking.

In most situations, you'll obtain the optimal results with your multi-site BSR applications if you follow the polling guidelines shown in the following table.

Recommended number of locations to poll

If there are this many switches in the network…	Each switch should poll…
2-4	all the other switches
5-10	3 other switches
11-20	4 other switches
21-40	5 other switches
41 or more	6 other switches

Which remote switches should each switch poll?

In networks with fewer than 5 switches, each switch can productively poll all the other switches in the network. In larger networks, each switch need not poll every other switch. But which switches should each switch poll? We'll use the term polling patterns to describe the relationships between switches in multi-site BSR applications.

Here are two patterns to avoid. They're simple and seem intuitively obvious, but they don't usually yield the best possible results:

- Mutual polling: As much as possible, 2 switches shouldn't poll each other. This is unavoidable in small networks, but in large networks it can and should be minimized.
- Polling chains: For example, if switch A polls B & C, B polls C & D, and so on, this is a polling chain.

You may want to experiment with polling patterns appropriate to your own network and applications (if you're not constrained by the physical structure of your network). The following table provides a template for creating polling patterns for applications of up to 12 switches. In the majority of situations, these patterns will produce results that are close to optimal. To use this table, first assign a number from 1 to x to each switch in your application. Next, find the column that matches the number of switches in your application. As you read down that column, you'll see which switches each particular switch in the application should poll.

Polling patterns for networks of 5-12 switches

This switch …	Should poll the specific switches shown in the column for your network size							
	5	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	2,4,5	2,4,5	2,4,6	2,4,7	2,4,6	2,4,7	2,4,8,10	2,4,8,9
2	3,5,1	3,5,6	3,5,7	3,5,8	3,5,7	3,5,8	3,5,9,11	3,5,9,10
3	4,1,2	4,6,1	4,6,1	4,6,1	4,6,8	4,6,9	4,6,10,1	4,6,10,11
4	5,2,3	5,1,2	5,7,2	5,7,2	5,7,9	5,7,10	5,7,11,2	5,7,11,12
5	1,3,4	6,2,3	6,1,3	6,8,3	6,8,1	6,8,1	6,8,1,3	6,8,12,1
6		1,3,4	7,2,4	7,1,4	7,9,2	7,9,2	7,9,2,4	7,9,1,2
7			1,3,5	8,2,5	8,1,3	8,10,3	8,10,3,5	8,10,2,3
8				1,3,6	9,2,4	9,1,4	9,11,4,6	9,11,3,4
9					1,3,5	10,2,5	10,1,5,7	10,12,4, 5
10						1,3,6	11,2,6,8	11,1,5,6
11							1,3,7,9	12,2,6,7
12								1,3,7,8

In applications of more than 12 switches, the following table provides the formulae you need to figure out the optimal polling pattern.

Polling pattern formula for large networks

Number of switches in application	Switch i should poll…
13 or 16	i + 1, i + 3, i + 7, i +11
14 or 19	i + 1, i + 3, i + 7, i + 9
15	i + 1, i + 3, i + 7, i + 10
17 or 20	i + 1, i + 3, i + 7, i + 12
18	i + 1, i + 3, i + 7, i + 13
21-23	i + 1, i + 3, i + 7, i + 15, i + 17

Polling pattern formula for large networks (continued)

Number of switches in application	Switch i should poll…
24	i + 1, i + 3, i + 7, i + 15, i + 19
25	i + 1, i + 3, i + 7, i + 15, i + 20

To use one of these formulae, first assign a number from 1 to x to each switch in your application. Then, in the left-hand column of the table, find the number of switches in your application. The corresponding formula in the right-hand column is the one you should use.

In the formulae, i is the number of the switch for which you're calculating a polling pattern. For example, let's say you want to calculate the polling patterns in an application with 16 switches. The formula to use is

$$i + 1, i + 3, i + 7, i + 11$$

as shown in the first row of the table. Here are the actual results of this formulae for the first 5 switches in this 16-switch application.Notice that the numbers wrap (start over at 1) after you've polled the last switch in the network: switch 5 polls switch 16 as its fourth poll, and then the polling pattern for switch 6 has switch 1 in the fourth position.

Switch number…	Should poll switches…
1	2 ,4, 8, 12
2	3, 5, 9, 13
3	4, 6 ,10, 14
4	5, 7, 11, 15
5	6, 8, 12, 16
6	7, 9, 13, 1
7	8, 10, 14, 2

Considerations for low volume splits/skills

This section includes the following topics:

- About low volume splits/skills on page 739
- Minimizing variations in wait time on page 740

About low volume splits/skills

Very small resources (for example, 2-3 agents) have special needs. With BSR, it is easy to obtain a very close balance of wait times across a network of call centers. however, for very small splits/skills, wait times for each call can vary significantly.

To see why this is, let's take an extreme example of a split with a single agent logged in with one call active and none in queue. Average call handling time is 3 minutes. Now, if a new call arrives in queue, that call could be answered almost immediately—or it might wait for 3 minutes or more. The variation in wait times is perhaps 5-180 seconds.

In general, the fewer agents logged into a split/skill, the greater the variability in wait times because agents become available less often. BSR will naturally favor large resources, steering calls away from smaller resources when there are no available agents or wait times are not the best in the application. This tendency helps reduce the possibility that an individual caller might have a disproportionately long wait at a small resource.

If your network includes very small splits/skills, you have three options:

- If your operation is not badly affected by a small percentage of calls having variable wait times, simply use BSR normally across the network.
- If your principal concern is that a call does not wait in queue while an agent is available elsewhere, use BSR normally but write primary vectors at smaller locations to perform rapid look-ahead attempts to other resources once the call has been queued. (Rapid LAI vector loops use the **interflow-qpos** conditional, which is an enhancement to LAI. For more information on LAI and the **interflow-qpos** conditional, see Look-Ahead Interflow (LAI) on page 273.) For an example of this type of vector, Using LAI as a backup on page 740.
- If you want to answer every caller quickly, then the following configuration is recommended. Do not deliver or queue calls directly to the very small resources. Deliver or queue all incoming calls to larger resources, and use BSR to balance the load across these larger locations. Some or all of the larger locations should then perform rapid look-ahead attempts to one or more of the smaller resources. In this way, the members of the very small resource become an extension of the agent pool at one of the larger call centers. For an example of this design, see Single-queue FIFO hybrid configuration on page 741.

In any network, avoid having several large resources poll or make look-ahead attempts to a very small resource. Since the status at the very small resource changes infrequently, frequent polls to that resource are wasteful. A very small resource should receive look-ahead attempts or be polled only by other small resources or by one large resource.

Minimizing variations in wait time

When a network contains (or when a call center application combines) large resources and very small resources, BSR and LAI can be effectively combined. This section presents two sample vectors. The first example shows a primary vector intended for the smaller resources in a network when you want to avoid having a call in queue at one call center while an agent is available at another. This design will reduce wait time variation as well. The second example illustrates a primary vector for larger locations: this example shows you the best way to minimize wait times across a network.

This section includes the following topics:

- Using LAI as a backup on page 740
- Single-queue FIFO hybrid configuration on page 741

Using LAI as a backup

As noted above, if your principal concern is that a call not wait in queue while an agent is available elsewhere, use BSR at all locations in the network. At smaller locations, write primary vectors that will perform rapid LAI attempts to other (preferably larger) resources once the call has been queued.

```
1.  wait time 0  secs hearing ringback
2.  consider skill 1st pri m adjust-by  0
3.  consider location 12     adjust-by 30
4.  consider location 22     adjust-by 30
5.  goto step 7 if expected-wait for call < 600
6.  disconnect after announcement 3501 "Due to heavy call volume..."
7.  queue-to skill best
8.  announcement 3500 "Thanks for calling...."
9.  goto step 13 if expected-wait for call < 90
10. wait time 45 secs hearing music
11. announcement 3502 "Still busy..."
12. goto step 9 if unconditionally
13. route-to-number 913031234567 with cov n if interflow-qpos = 1
14. wait time 5 secs hearing music
15. goto step 13 if unconditionally
```

Steps 1 to 4 comprise a typical BSR vector. The origin switch considers a local resource and 2 remote resources. Before queuing or routing the call, however, the vector checks the expected wait time for the best resource. If this is 10 minutes or more, the caller receives a busy announcement. Otherwise, the queue-to best step sends the call to the best resource. Two vector loops follow: one 45-second loop with music and a delay announcement, and one 5-second loop that uses LAI. If the call is queued successfully in step 7 the first announcement loop (steps 9-12) executes until the call gets within a certain range of the head of the queue (at

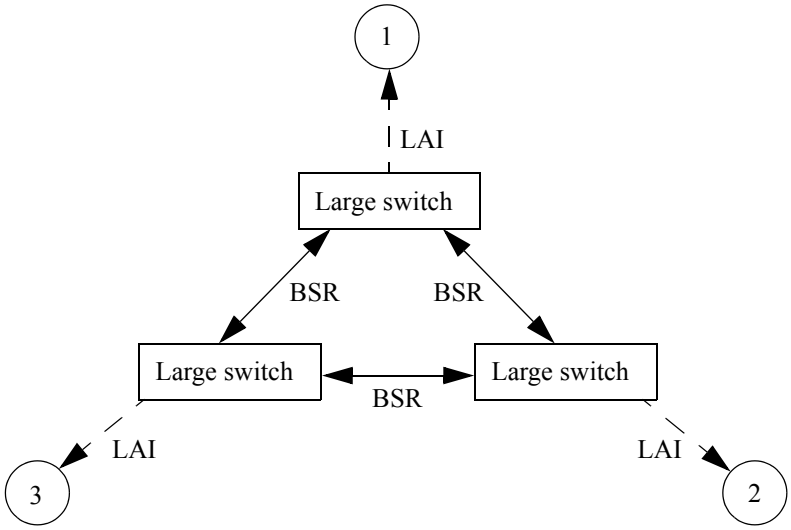
which point EWT is less than 90 seconds). At this time, step 9 sends the call to the second loop, where LAI attempts are placed every 5 seconds for the call at the head of the interflow eligible queue (**`interflow-qpos=1`**). If an agent becomes available at the larger remote resource, any call at the head of the eligible queue at the smaller location is outflowed to the larger resource, normally within a period of 5 seconds.

Single-queue FIFO hybrid configuration

To minimize variations in wait time across a network, the best strategy may be to let only the call centers with the larger resources receive calls. The following figure shows a network of 3 large and 3 small resources (call centers with large splits/skills and call centers very small splits/skills in the same application).

The large locations use BSR and all poll each other, while each location with a small resource (numbered 1, 2, 3) is treated as a satellite of one of the larger locations and only receives calls interflowed from that location. (Mutual polling is not optimal in larger networks, but it's OK for switches in such a small network to poll each other.) So BSR is used to balance the load between the locations with the larger resources. Then, each large switch executes a rapid LAI vector loop to one small switch to look for available agents. Since calls never queue at the small switches, the problem of highly variable wait times at the small resources is eliminated. This strategy will also give the best balance in wait times across resources.

Hybrid application architecture



The following vector example shows the primary vector that would be used at the large locations with this strategy. This vector is almost identical to the vector shown in Using LAI as a backup on page 740 above. The differences are at the application level. In contrast to the previous example:

- Only the locations with the larger resources receive calls.
- The primary vector shown here resides on the larger switches.

Steps 1 to 4 comprise a typical BSR vector. The origin switch considers a local resource and 2 remote resources. Before queuing or routing the call, however, the vector checks the expected wait time for the best resource. If this is 10 minutes or more, the caller receives a busy announcement. Otherwise, the queue-to best step sends the call to the best resource. Two vector loops follow: one 45-second loop with music and a delay announcement, and one 5-second loop that uses LAI. If the call is queued successfully in step 7, the first announcement loop (steps 9-12) executes until the call gets within a certain range of the head of the queue. At this time, step 9 sends the call to the second loop, where LAI attempts are placed every 5 seconds (only for the call at the head of the interflow eligible queue). If an agent becomes available at the smaller resource, any call at the head of the eligible queue at the larger location is outflowed to the smaller resource, normally within a period of 5 seconds.

Vector combining BSR and LAI

```
1.  wait time 0  secs hearing ringback
2.  consider skill 1st pri m adjust-by  0
3.  consider location 120    adjust-by 30
4.  consider location 220    adjust-by 30
5.  goto step 7 if expected-wait for best < 600
6.  disconnect after announcement 3501 "Due to heavy call volume..."
7.  queue-to skill best
8.  announcement 3500 "Thanks for calling...."
9.  goto step 13 if expected-wait for call < 90
10. wait time 45 secs hearing music
11. announcement 3502 "Still busy..."
12. goto step 9 if unconditionally
13. route-to-number 913031234567 with cov n if interflow-qpos = 1
14. wait time 5 secs hearing music
15. goto step 13 if unconditionally
```

Similar vector loops can be added to the interflow vectors at each of the large switches. In other words, each vector that processes calls at the larger locations can use rapid LAI loops to interflow calls to its satellite resource. This system maximizes agent utilization and the distribution of call load while evening out wait times across the network.

Appendix F: Advanced information forwarding for ISDN trunks

This section explains ISDN (BRI or PRI) trunk group setting interactions with Information Forwarding, UCID, and Multi-Site Routing.

This section includes the following topics:

- About advanced information forwarding on page 743
- Non-QSIG protocol on page 744
- QSIG trunk group on page 744
- Send Codeset 6/7 LAI IE option interactions on page 745

About advanced information forwarding

User information included in the SETUP message for an outgoing call (at the sending switch) or DISCONNECT message sent back for an incoming call (at the receiving switch) is based on the trunk group settings at the sending or receiving sites.

The shared user information forwarding supports various data items (UCID, collected digits and In-VDN-Time) in addition to shared forwarding of LAI Info (VDN-Name and Other-LAI) and ASAI provided user data. Shared forwarding over non-QSIG ISDN trunks packs the data items in a codeset 0 UUI IE (called shared UUI), where each item consists of a two-byte header (application ID and data length). Shared forwarding over QSIG trunks transports the data items as Manufacturer Specific Information (MSI) in codeset 0 Facility IEs.

BSR and shared data forwarding (UCID and other new data items) requires QSIG or the shared **UUI Treatment** setting with non-QSIG trunk groups on both the sending (outgoing trunks) and receiving (incoming trunks) at the switch. Shared settings are also required on tandem trunk connections through the switch that routes these calls. LAI Info, UCID, collected digits, In-VDN-Time and ASAI provided user data can be forwarded with a call in the SETUP message (LAI or BSR interflowed call, a tandemed call, for UCID with any outgoing call and for ASAI user data any adjunct routed outgoing call). Only BSR reply-best data is returned with a BSR poll call and only ASAI user data is returned for a non-poll call in a DISCONNECT message (both types of data will not be included in the same DISCONNECT message). Shared UUI Priority settings do not affect what is put in the DISCONNECT message or data forwarded over QSIG trunks.

The protocol (QSIG or non-QSIG) is set on page 1 of the ISDN trunk group screen using the **Supplementary Service Protocol** field. QSIG type as defined for shared MSI is protocol type b (another protocol type d, ECMA QSIG is considered non-QSIG for Information Forwarding). The **Send Codeset 6/7 LAI** field on page 2 indicates whether or not to include an LAI IE in the SETUP message. The codeset used (6 or 7) is determined by the **Codeset to Send TCM, Lookahead** field on page 1. The **Send UCID** field on page 2 indicates whether or not the UCID data item should be included as user information with calls routed over this trunk group. The **Send Codeset 6/7 LAI IE** field is ignored for BSR polls over the trunk group (an LAI IE will never be included with BSR calls).

Non-QSIG protocol

UUI Treatment set to **service-provider** includes any application provided UUI in a codeset 0 UUI IE on a non-shared basis. That is, the data portion of the UUI IE only includes user info in the SETUP or DISCONNECT messages as provided by an application such as ASAI without the shared App-ID and length header fields. User data from only one application can be included in non-shared UUI. This setting would be used for non-QSIG trunk groups when service-provider functionality is wanted (for example, where shared forwarding of the new data items is not required or for trunk groups to other vendor switches or network services that need user information from the trunk group in a non-shared UUI IE such as provided by ASAI). Incoming calls received with shared user information (shared UUI IE) that are routed outgoing over a non-QSIG service-provider trunk group will forward only ASAI provided user data in a non-shared UUI IE.

UUI Treatment set to **shared** allows all applications to include data items in the UUI IE on a shared forwarding basis. The Shared UUI Feature Priorities page settings along with the **Max. Size of UUI Contents** field on page 2 and the features configured for the system determines what actually is included in the UUI IE. This is the normal setting for non-QSIG trunk groups that route calls to the switch over private or public networks when information forwarding is required and must be used for BSR.

QSIG trunk group

UUI Treatment set to **service-provider** forwarded ASAI provided user data in a non-shared codeset 0 UUI IE and all other user data in codeset 0 Facility IEs as MSI. In this case the **Max. Size of UUI Contents** field is not relevant and the Shared UUI Feature Priorities page does not show nor apply. This setting would only be used for QSIG trunk groups to pre-R6.3 DEFINITY switches for compatibility with existing ASAI applications or when service-provider functionality is wanted (e.g., where shared forwarding of the new data items is not required or for trunk groups to other vendor switches that need user information from the trunk group in a non-shared UUI IE such as provided by ASAI). Incoming calls received with shared data (shared UUI IE) routed out over a QSIG service-provider trunk group, will separate any ASAI provided user data included in the shared UUI IE and forward it in a non-shared UUI IE.

UUI Treatment set to **shared** will forward all user information including ASAI provided user data in codeset 0 Facility IEs as MSI in the SETUP or DISCONNECT message. The UUI IE is never included over a shared QSIG trunk group. In this case, the **Max. Size of UUI Contents** field and the Shared UUI Feature Priorities page do not apply. This is the normal setting for QSIG trunk groups to the switch when information forwarding is required and must be used for BSR.

Send Codeset 6/7 LAI IE option interactions

The **Send Codeset 6/7 LAI IE** option is independent of the **Supplementary Service Protocol** and **UUI Treatment** settings to allow additional flexibility. The switch can have a mix of trunk groups set with non-QSIG or QSIG protocol and with **service-provider** or **shared** settings. Calls interflowed over the shared non-QSIG trunk groups will contain the data items to be forwarded with the call in the UUI IE while calls interflowed over the non-QSIG service-provider trunk groups will not (except for ASAI which can always be sent in UUI). Calls interflowed over the QSIG trunk groups will always have MSI user information (except for ASAI whose transport method depends on the **UUI Treatment** setting).

When a call is LAI interflowed over a non-QSIG service-provider trunk group, the `Send Codeset 6/7 LAI IE` option being active will result in just the LAI IE being forwarded with the call in a SETUP message. When interflowed over a non-QSIG shared trunk group, setting the **Send Codeset 6/7 LAI IE** to **yes** includes a codeset 6/7 LAI IE in the SETUP message in addition to the same LAI information included as shared data in the UUI IE. If necessary and appropriate, the LAI information fields (and others) can be set to blank on the Priorities page to exclude these data items from the UUI IE. For details, see Determining user information needs on page 196. When interflowed over a QSIG service-provider or shared trunk group with **Send Codeset 6/7 LAI IE** active, the LAI information will be included as both MSI and in the LAI IE. However, in this case there is no mechanism to eliminate the duplication of data if the codeset 6/7 LAI IE is required.

These combinations can be used when calls are LAI interflowed to the switches previous to the switch with existing ASAI applications using ASAI provided UUI that may or may not be using the LAI IE. Note that codeset 6/7 IEs are not defined for QSIG and other vendor switch treatment of calls with a LAI IE is undefined (could be ignored, blocked, or misinterpreted).

When the trunk group is set to non-QSIG and shared or to QSIG (service-provider or shared), it is recommended that the **Send Codeset 6/7 LAI IE** option should not also be set to y due to the overhead of sending duplicate information. In some cases, this configuration could exceed the SETUP message and/or user information byte count limits for the network and result in the user information being dropped. Also, transport could cost more in networks which charge for user transport by quantity of bytes transported. An administration warning message will be given when this combination is set for the trunk group. In fact this combination is not recommended except in very limited cases where a mix of early and later switches can be reached over the same trunk group (using a public or switched private network) using Look-Ahead Interflow, and where BSR or UCID is not active or being used and the data that needs to be forwarded with the call can be limited to that supported by the network.

The **Send Codeset 6/7 LAI IE** option must not be set to y with trunk groups (or in switches) where calls will be interflowed over public networks or virtual private networks that do not support codeset 6/7 transport. In these cases, the codeset 6/7 IE will not be forwarded or the calls may not be routed by the network (blocked due to protocol errors). This can happen in some international situations, notably over networks in Germany.

Summary of what is included in the SETUP message

		Supplementary services protocol	
UUI Treatment	**Send Codeset 6/ 7 LAI IE**	**Non-QSIG (other than b)**	**QSIG (SS b) [1]**
service-provider	n	ASAI provided user info in codeset 0 UUI IE	ASAI provided user info in a codeset 0 UUI IE and all other user info in codeset 0 MSI
	y	ASAI provided user info in codeset 0 UUI IE & a codeset 6/7 LAI IE	ASAI provided user info in codeset 0 UUI IE, all other user info in codeset 0 MSI and a codeset 6/7 LAI IE [2]
shared	n	All user info in a shared codeset 0 UUI IE	All user info in codeset 0 MSI
	y	All user info in a shared codeset 0 UUI IE & a codeset 6/7 LAI IE [3]	All user info as codeset 0 MSI and a codeset 6/7 LAI IE [3]

1. MSI is sent in codeset 0 Facility IEs.
2. With this combination, the LAI information (LAI Name and Other LAI) will be sent both as MSI (in a Facility IE) and in the LAI IE. Note that LAI IE and shared MSI operation with other vendor switches is undefined.
3. With this combination, the LAI information (VDN-Name and Other-LAI) will be sent in both the UUI IE and in the LAI IE (setting the UUI Priorities for these items to blank can eliminate the duplication).

When to use specific trunk group options

	Trunk group option settings		
	UUI treatment		**Send Codeset 6/7 LAI IE**
Situation	**Non-QSIG**	**QSIG**	
Trunk groups over which information forwarding is not required (for LAI, BSR or UCID transport).	service-provider	service-provider	n

When to use specific trunk group options (continued)

Non-LAI interflow or tandem calls to service providers or other vendor switches that do not recognize shared UUI.	service-provider	service-provider	n
LAI to pre-R6.3 switches over networks that block codeset 6/7 IE calls.	service-provider	service-provider	n
LAI to pre-R6.3 switches over networks that allow codeset 6/7 (traditional LAI) with or without ASAI applications that use UUI and/or LAI Info	service-provider	service-provider [1]	y
LAI over public/virtual private network to mixed R6.3 and earlier switches, where the Avaya switches have shared information forwarding. The pre-R6.3 switches may use LAI Info in an ASAI application, but must not use UUI.	shared [2]	shared [2]	y
LAI over public/virtual private network to mixed R6.3 and earlier switches. The R6.3 and earlier switches may use LAI info or UUI in an ASAI application.	service-provider [3]	service-provider [2]	y
BSR and/or LAI to all R6.3 or newer switches [4]	shared	shared	n

1. With this combination, the LAI information will be sent both as MSI (in Facility IEs) and in the LAI IE.
2. With this combination, the LAI information (LAI Name and Other LAI) will be sent in both the UUI IE and in the LAI IE.
3. The LAI IE and ASAI non-shared UUI is supported, but BSR, UCID and other new data items are not.
4. All switches interflowed to must be R6.3 or newer with shared incoming and outgoing trunk group settings. Tandeming/interflowing through R6.3 or later switches requires shared settings. Switches tandemed through can be older than R6.3 (or other vendor switches that pass codeset 0 UUI or MSI transparently). This is the only combination that supports BSR and new data items information forwarding. In this scenario it is recommended to never set **Send Codeset 6/7 LAI IE** to **y** in order to save SETUP message space and to ensure operation over networks that do not allow codeset 6/7 IEs. This combination is the recommended setup for Multi-Site Routing.

Appendix G: Functional differences for DEFINITY G2 and Communication Manager

The Call Vectoring functional differences between the DEFINITY G2 switch and the Avaya Communication Manager system are no longer relevant. If this information is required, please refer to the Release 5.0 (or earlier) version of this guide. These documents are available in the Call Center product documentation section of the Avaya support Website at:

http://www.avayadocs.com

.

Appendix H: Call Vectoring/EAS and BCMS/CMS interactions

This section includes the following topics:

- About Call Vectoring/EAS and BCMS/CMS interactions on page 751
- CMS/BCMS tracking in a Call Vectoring environment on page 752
- Using CMS and BCMS reports to evaluate Call Vectoring activity on page 761
- Using CMS in an EAS environment on page 763

About Call Vectoring/EAS and BCMS/CMS interactions

Call Vectoring and Expert Agent Selection (EAS) interact with a management information system that helps to monitor and report on the activity within Call Vectoring and EAS. In most cases, the management system is either the Call Management System (CMS) or the Basic Call Management System (BCMS).

The CMS, which resides on an adjunct processor, collects and processes ACD information to generate reports. BCMS, which resides on the switch, also collects ACD information and generates a limited number of reports. The CMS reporting and data storage capabilities are much more extensive than those of the BCMS.

BCMS collects and processes ACD information to generate various reports.

This section is intended to illustrate how this system interprets these management systems interpret and reports report on activity within Call Vectoring and EAS. Special emphasis is placed on interpreting and reporting on this activity as it occurs within splits during a series of Call Vectoring or EAS events.

Note:
Call Vectoring commands on page 521 provides a summary of the CMS/BCMS interactions with each Call Vectoring command (where applicable).

CMS/BCMS tracking in a Call Vectoring environment

This section includes the following topics:

- About CMS/BCMS tracking on page 752
- Defining and interpreting call flows on page 752

About CMS/BCMS tracking

Tracking is the identifying of call flows and other actions relevant to call handling. There are three classes of call flows: split flows, VDN flows, and vector flows. We are most concerned with tracking in the Call Vectoring environment. The specific types of call flows and actions in this environment that are tracked by the CMS/BCMS include the following:

- Inflows (flow ins)
- Outflows (flow outs)
- Dequeues
- Abandons
- Answers
- Busies
- Disconnects

The split supervisor can use VDN and vector flows to evaluate how effective vector programming is at the site in question. The supervisor can use split flows to determine the manner in which the splits at the site are handling incoming telephone calls.

Defining and interpreting call flows

The manner in which specific call flows are defined and interpreted depends upon the call flow class in question, the management system in effect, and the version of the switch being used. Management systems include CMS and BCMS.

The following sections define and interpret specific call flows according to these parameters.

Answered and abandons

The most important tracking items for most VDNs and vectors are the number of calls answered and the number of calls abandoned. The CMS provides VDN profiles that show when calls are answered and abandoned. Ten service level intervals are administered for these profiles. These intervals can have smaller time intervals around the time most calls are answered and when most calls abandon to get more detailed information.

This data can be used to determine what an acceptable service level is for most callers. The percentage answered within the administered acceptable service level is also shown on the Call Profile reports. For VDNs, the calculation is ACD calls answered and non ACD calls connected within the service level divided by calls offered to the VDN (including calls that inflow to the VDN).

For split/skill statistics, the calculation is ACD calls answered within the service level divided by calls queued to the split/skill (answered calls, abandoned calls, calls that flow out, calls that dequeue). In most cases the VDN percentage will be higher then the split percentage since calls dequeued from a split/skill are counted as answered, abandoned, or outflows for the VDN.

Changes made to a vector or to staffing will typically affect the VDN call profile. Even the wording of an announcement can affect the abandon profile. It is worthwhile to review the VDN's call profile before and after any change to determine if the change had a positive impact.

Busies and disconnects

Busy calls and forced disconnects reported on the CMS indicate how many calls this VDN/vector turned away. If forced disconnect is used out of business hours, this item would indicate how many customers expected you to be operating during a specific time interval. If busies are given when the queues are full or waiting times are long, the number of busies in an interval might suggest a staffing change is needed. If disconnect is used to deny a look-ahead interflow attempt, a large number of denials would indicate a busy time at multiple sites.

VDN inflows and outflows

The following section discusses the specific VDN flows for CMS and BCMS.

CMS and BCMS standards

The following table illustrates how CMS and BCMS interprets specific VDN flows from the switch:

CMS and BCMS standards for interpreting VDN flows		
Flow type	**Management system**	**Interpretation**
VDN flow in	CMS	Calls that flow into the VDN using a **route-to VDN** command or by Redirection on No Answer to a VDN.
	BCMS	(Not tracked.)
VDN flow out	CMS	Calls that successfully flow out of a VDN to another VDN or to an external location using a **route-to** command.
	BCMS	Same as for CMS.

Vector inflows and outflows

The following section discusses the specific vector flows as recorded by the CMS.

CMS standards

Vector flow in pertains to calls that flow into a vector from another vector using a route to or a goto vector command. Vector flow out pertains to calls that successfully flow out of a vector using a route to or a goto vector command.

Split inflows, outflows, and dequeues

The following sections discuss the various split flow types for CMS and the BCMS.

CMS and BCMS standards

The CMS and the BCMS are grouped together because both of these systems interpret two split flow types identically. These flows include inflow and outflow. The CMS interprets another split flow type, dequeue. The BCMS does not interpret this split flow type because it does not have a dequeue tracking item. This means that in a situation where the CMS tracks a dequeue, BCMS does not because it is unable to do so.

Before we detail how the CMS and the BCMS interpret split flows, we should discuss the term primary split, since this concept plays a significant role in tracking. Primary split is defined as the first split in a VDN to which a call actually queues. Therefore, this split is not necessarily the first split referenced in the vector.

Another split becomes the primary split if either of the following events occurs:

- Call cannot queue to the originally-targeted split because the split has no queue slots available.
- Call leaves the VDN (using a **route-to** VDN command, for example) and is queued to another split as a result.

If the call leaves vector processing and does not queue to another split (as a result of a **route-to extension** command, for example), there is no new primary split.

With this discussion in mind, let's take a look at the following table to see how CMS and BCMS interpret split flows for the switch.

CMS and BCMS standards for interpreting split flows

Flow type	Management system	Interpretation
Inflow	CMS	Calls that ring at an agent in a split other than the primary.
	BCMS	Same as for CMS.
Outflow	CMS	Calls that are dequeued from a primary split using a **route-to** or **messaging split** command, or by ringing at or being answered by an agent in another split to which the call is also queued.
	BCMS	Same as for CMS.
Dequeue	CMS	Calls that are dequeued from and not answered by any split other than the primary split in a VDN.
	BCMS	Not tracked.

When a call is not answered (due to an outflow, abandon, busy, or disconnect), the call's disposition is tracked for the primary split as long as the call is still queued when the call abandons, outflows, etc. However, if the call abandons or outflows from ringing, the disposition is recorded for the split for which it was ringing. On the CMS, the other splits to which the call is queued track a dequeue when the call outflows, abandons, is given busy treatment, or is disconnected.

If the primary split in a VDN is unmeasured, an outflow, abandon, busy, or disconnect is not tracked for the call. Also, an answer is not tracked if the call is answered by an agent in the primary split.

Examples of split flow tracking

The following sections provide some examples of tracking in CMS and BCMS. Each section first presents a scenario of Call Vectoring events. The scenario is then followed by a table in which the tracking for the various splits involved is recorded. Following each tracking table, an explanation of the tracking procedure is provided.

The scenarios presented include the following:

- Call answered by a primary split.
- Call answered by a non primary split.
- Call abandoned from queue.
- Call answered by a primary split after a route to VDN.
- Call answered by a non primary split after a route to VDN.
- Call answered after a route to split.

Note:
Inflows, outflows, and dequeues are not tracked for splits administered by the **converse-on split** command. However, if a call is answered both by a converse split and (subsequently) by a non converse split, an answer is tracked for each split. However, a call is really considered answered only when it is answered by a non converse split. Therefore, traffic measurements for converse splits should be used only to measure converse split traffic and not to calculate the total number of calls.

Call answered by a primary split: The following scenario involves a call answered by the primary split. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Call is answered in split 1.

The following table shows the tracking table for this scenario:

Tracking for call answered by primary split

Split tracking			
	1	**2**	**3**
CMS	answer	dequeue	dequeue
BCMS	answer		

Comments:

- CMS: Dequeue is tracked in split 2 as well as in split 3 because the call is answered by the primary split (split 1) and is thus dequeued from splits 2 and 3 without being answered in these splits.
- BCMS: No dequeue tracking item is available.

Call Answered by a non-primary split: The following scenario involves a call answered by a non primary split. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Call is answered in split 2.

The following table shows the tracking table for this scenario:

Tracking for call answered by non-primary split

	Split tracking		
	1	**2**	**3**
CMS	outflow	inflow answer	dequeue
BCMS	outflow	inflow answer	

Comments:

- CMS: Outflow is tracked in split 1 because the call is answered by an agent in another split to which the call is queued (that is, split 2). Although the call is obviously removed from split 1 after it is answered in split 2, dequeue is not tracked in split 1 because split 1 is the primary split. Inflow is tracked in split 2 because the call is answered in this split and the split is not the primary split. Dequeue is tracked in split 3 because the call is removed from the split without being answered there. When the call is removed from split 3, outflow is not tracked in split 3 because this split is not the primary split.
- BCMS: Outflow is tracked in split 1 because the call is answered by an agent in another split to which the call is queued (that is, split 2). Inflow is tracked in split 2 because the call is answered in this split and the split is not the primary split. When the call is removed from split 3, outflow is not tracked in split 3 because this split is not the primary split.

Call Abandoned: The following scenario involves a call abandoned by the caller. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2, 2 and 3.
2. Call is abandoned.

The following table shows the tracking table for this scenario:

Tracking for Abandoned Calls

	Split Tracking		
	1	**2**	**3**
CMS	abandon	dequeue	dequeue
BCMS	abandon		

Comments:

- CMS: Abandon is tracked in split 1 because this split is the primary split. Dequeue is tracked in splits 2 and 3 because the call is dequeued from these splits without being answered in either split.
- BCMS: Abandon is tracked in split 1 because this split is the primary split. Tracking is not recorded in splits 2 and 3 because no dequeue tracking item is available.

Call answered by a primary split after a route to VDN: The following scenario involves a call answered by the primary split after a **route-to** VDN command is executed. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Vector executes a **route-to VDN** step.
3. Call is then queued to splits 4, 5 and 6.
4. Call is answered in split 4.

The following table shows the tracking table for this scenario.

Tracking for call answered by primary split after route to VDN

	Split tracking					
	1	**2**	**3**	**4**	**5**	**6**
CMS	outflow	dequeue	dequeue	answer	dequeue	dequeue
BCMS	outflow			answer		

Comments:

Split 1 is the original primary split, because this is the first split to which the call actually queues. However, split 4 becomes the new primary split because:

- Call leaves the original VDN upon execution of the **route-to VDN** step.
- Split 4 is the first split to which the call queues upon execution of this step.
- CMS: Outflow is tracked in split 1 because this split is the original primary split, and the call is dequeued from this split using a **route-to VDN** step. Dequeue is tracked in splits 2, 3, 5, and 6 because the call is dequeued from each of these splits without being answered in any one of them.
- BCMS: Outflow is tracked in split 1 because this split is the original primary split.

Call answered by the non-primary split after a route to VDN: The following scenario involves a call answered by the non primary split after a **route-to VDN** command is executed. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Vector executes a **route-to VDN** step.
3. Call is then queued to splits 4, 5 and 6.
4. Call is answered in split 5.

The following table shows the tracking table for this scenario:

Tracking for call answered by non-primary split after route to VDN

	Split tracking					
	1	**2**	**3**	**4**	**5**	**6**
CMS	outflow	dequeue	dequeue	outflow	inflow answer	dequeue
BCMS	outflow			outflow	inflow answer	

Comments:

- CMS: Outflow is tracked in split 1 because this split is the original primary split, and the call is dequeued from this split using a **route-to** VDN step. Dequeue is tracked in splits 2, 3, and 6 because the call is dequeued from each of these splits without being answered in any one of them. Outflow is tracked in split 4 because this split becomes the new primary split after the **route-to VDN** step is executed and the call is subsequently dequeued from this split by being answered in another split (split 5) to which the call is also queued. Finally, inflow is tracked in split 5 because the call is answered in this split, and the split is not the primary split.
- BCMS: Outflow is tracked in split 1 because this split is the original primary split. Outflow is tracked in split 4 because this split becomes the new primary split after the **route-to VDN** step is executed. Finally, inflow is tracked in split 5 because the call is answered in this split, and the split is not the primary split.

Call answered after a route to split: The following scenario involves a call answered after it is routed to a split using a **route-to digits** or **messaging split** command. The scenario is as follows:

1. Call comes into a VDN whose vector queues the call to splits 1, 2 and 3.
2. Vector executes a **route-to digits** (or **messaging split**) step.
3. Call is queued to split 4 and answered by an agent in split 4.

The following table shows the tracking table for this scenario:

Tracking for call answered after route to split

	Split tracking			
	1	**2**	**3**	**4**
CMS	outflow	dequeue	dequeue	answer
BCMS	outflow			answer

Comments:

- CMS: Outflow is tracked in split 1 because this split is the original primary split, the call is dequeued from this split using a **`route-to digits`** (or **`messaging split`**) step, and the call is answered in split 4, which becomes the new primary split. Dequeue is tracked in splits 2 and 3 because the call is dequeued from each of these splits without being answered in any one of them.
- BCMS: Outflow is tracked in split 1 because this split is the original primary split, and the call is answered in split 4, which becomes the new primary split.

Evaluating split performance

By using the information presented to this point, along with the information from various reports (as discussed in the next section), the split supervisor can answer one or more questions concerning split performance and then make adjustments, if necessary. Here are some of the questions the supervisor can answer:

1. How many ACD calls offered to my split were mine (that is, were offered to this split as the primary split)?

   Note:
   Split ACD calls include direct agent calls for BCMS, but not for CMS, which tracks direct agent calls separately.

2. How many of my ACD calls did my split not answer?
3. How many ACD calls that I didn't answer weren't mine?

The following sections present the answers to these questions from the perspective of the CMS and BCMS.

CMS: The following answers reflect the use of the CMS:

- The number of calls offered to my (primary) split that were mine can be determined by an examination of the CMS Split Summary Report. The algorithm is as follows: CALLSOFFERRED - INFLOWCALLS - DEQUECALLS (that is, the total number of calls offered minus the number of calls not mine that I answered minus the number of calls not mine that I didn't answer.)

- The number of my calls that my split didn't answer can be determined by an examination of the CMS VDN Report. The algorithm is as follows: ABNCALLS + BUSYCALLS + DISCCALLS + OUTFLOWCALLS (that is, the number of abandoned calls plus the number of busy calls plus the number of disconnected calls plus the number of calls outflowed from my split tagged as a primary split).
- The number of calls not mine that my split didn't answer is DEQUECALLS, which is indicated in the CMS Split Summary Report.

BCMS: The following answers reflect the use of BCMS:

- The number of calls offered to my split that were mine can be determined by an examination of the BCMS Split Report. The algorithm is as follows: ACDCALLS + ABNCALLS + OUTFLOWCALLS - INFLOWCALLS (that is, the total number of calls answered plus the total number of calls abandoned from my split tagged as a primary split plus the number of calls that outflowed my split tagged as a primary split minus the number of calls answered that were not directed to my split tagged as a primary split).

Using CMS and BCMS reports to evaluate Call Vectoring activity

There are a number of CMS and BCMS reports that allow you to evaluate Call Vectoring activity. Some of these facets include the call flows present within Call Vectoring as well as the speeds at which calls are answered. The sections that follow identify and discuss the CMS and BCMS reports that indicate this activity.

This section includes the following topics:

- CMS reports on page 761
- BCMS reports on page 762

CMS reports

CMS has real-time, historical, and integrated reports. Most of the CMS historical reports are available in four versions: intra-hour, daily, weekday, and monthly. The following list identifies and describes several CMS reports that summarize Call Vectoring activity. For further details on these and other related reports, see *Avaya CMS Supervisor Reports*.

Note:
The reports described in this section are generated in CMS R3 and newer releases of the CMS. Corresponding CMS R2 reports do not provide information that reflects capabilities that are new to the switch (for example, internal/external call tracking).

- Split Summary Report summarizes the call activity for an entire split. Among other information, the report provides the number of calls answered, the total number of flow ins (inflows), flow outs (outflows), dequeues, and abandoned calls.

  The report also indicates the average speed of answer (interval ASA) for calls. This refers to the sum of the queue time and ring time for a call within the answering split only. Finally, the report indicates the dequeued average queue time, which is the average time a call waits until it is answered by another split to which the call is also queued.

- VDN Report summarizes VDN activity for specific vectors. Among other information, the report provides calls answered, connected, abandoned, the number of VDN Flow Ins/Outs, calls forced busy, and calls forced disconnect. VDN Flow In pertains to calls that flow into a VDN from another VDN using a **`route-to`** command. VDN Flow Out pertains to calls that successfully flow out of VDN to another VDN or external location using a **`route-to`** command.

- Vector Report summarizes vector activities. Among other information, the report provides the number of calls offered, calls answered, calls abandoned, Vector Flow Ins/Outs, calls forced busy, and calls forced disconnect. Vector Flow In pertains to calls that flow into a vector from another vector using a **`route-to`** or **`goto vector`** command. Vector Flow Out pertains to calls that successfully flow out of a vector using a **`route-to`** or **`goto vector`** command.

BCMS reports

BCMS has a real-time split report, split historical reports, real-time VDN reports, and VDN historical reports. The following list identifies and describes several BCMS reports that summarize Call Vectoring activity. For more information on these and other related reports, refer to *Avaya Aura™ Communication Manager 5.2 Software - Basic Call Management System (BCMS) Operations*.

BCMS Split Report: Summarizes the call activity for an entire split. The information can be requested either daily or by the administered time period. Among other information, the report provides the total number of flow ins (inflows) and flow outs (outflows), the calls answered and calls abandoned. The report also provides the average speed of answer time for calls handled by the split during the indicated time period.

VDN Summary Report: Summarizes statistical information for all internally-measured VDNs. The information can be requested by the administered time interval or daily. The **`list bcms vdn`** report gives multiple time periods or days for a single VDN. The **`list bcms summary vdn`** report gives a one-line summary per vdn (with data from the specified times or days), but can give the data for numerous vdns.

The report also indicates the total number of flow outs, specifically, the number of calls that route to another VDN or to a destination external to the switch. However, calls that encounter a **goto vector** command are not shown as outflows. No further measurements are taken on the calls once the calls have outflowed. If an outflowed call later abandons, this is not indicated in the report.

Among other information, the VDN report provides a total for offered calls, answered calls, abandoned calls, and also one for calls that were either **forced busy** or **forced disconnect**.

VDN Real-Time Report: Provides statistical information including the number of calls currently waiting and the oldest call waiting. The VDN real-time report has the same characteristics as other real-time BCMS reports.

Using CMS in an EAS environment

The same tracking and database items used within a traditional Call Vectoring environment are used within an EAS environment but there are also new items that are specific to EAS. All existing custom reports should work when you are upgrading to EAS.

The following sections explain how the following entities are tracked in an environment with EAS optioned:

- Agents and their skills on page 763
- DAC calls on page 764
- Non-ACD calls on page 764
- VDN skill preferences on page 764

Agents and their skills

The fields under the Extn column in the CMS Real-Time Agent Report show the extension that the agent is logged into. These fields can be used to locate the agent or to service observe the agent.

With EAS optioned, the Skill Status Report replaces the Split Status Report. This report indicates the skills logged into and the skill level of each skill. If too many calls are waiting, or if calls are waiting too long (also shown on the Skill Status report), it is possible that not enough agents have the skill administered at a high enough skill level.

An agent may be denied login to some skills if the maximum agents/skill number is met or if the CMS limit on agent/skill pairs logged in has been reached.

CMS reports show only the first 15 skills that an agent is logged into.

DAC calls

Waiting direct agent calls are not included in the **Calls Waiting** and **Oldest Call Waiting** report fields for skills because such calls are not skill calls. However, direct agent calls are included in these two report fields for VDNs.

The Queue/Agent Summary Real-Time Report lists separately the direct agent calls waiting in a skill queue. direct agent calls are queued to the skill that is administered as the direct agent skill. To manage the skill's queue slots effectively, it is recommended that a skill be dedicated for direct agent calls.

Since direct agent calls are not skill calls, the skill tables do not track direct agent calls; however, the tables do monitor skill queue slots. The agent's time is tracked as OTHER in the skill tables. In the agent tables, there are separate direct agent call items. The standard CMS agent reports add the direct agent calls and the skill ACD calls and report these calls as ACD calls. The VDN tables track direct agent calls as ACD calls.

Non-ACD calls

The first measured skill that an EAS agent is logged into is used by CMS to track non-ACD calls unless the agent has an ACD call on hold. If an ACD call is on hold, outgoing non-ACD calls are counted for the skill of the held ACD call.

VDN skill preferences

VDN skill preference data is collected to provide information on what groups of agents (skills) are handling calls and on how effectively each skill group handles a particular VDN.

Real-time and historical VDN Skill Preference reports can be used to compare the percentage of calls being answered by the 1st, 2nd, and 3rd VDN preferences against an objective. If too few calls are being answered by the 1st skill preference, the vector can be adjusted to allow more time for the 1st skill preference group to answer calls; another alternative is to train or hire more agents with the 1st skill preference.

You can use VDN skill preference data to compare the average talk time and average ACW time for agents in the 1st, 2nd, and 3rd skill groups. If these times vary too much across groups, more training may be needed for the backup groups (that is, the 2nd and 3rd skill groups).

VDN skill preference data is tracked according to the skill preferences (1st, 2nd, 3rd) assigned to the VDN. Whenever a vector step either references a 1st, 2nd, or 3rd skill or specifies a skill number that matches the 1st, 2nd, or 3rd skill administered, the new database items are tracked. For example, if VDN 1000 has Skills 21, 22, and 23 administered as the 1st, 2nd, and 3rd skills, respectively, and if the vector associated with VDN 1000 has a queue to main skill 22 step, tracking occurs for the 2nd VDN skill preference if the call is answered by an agent in Skill 22. Skill preference tracking also occurs for Skills 21 and 23. This allows users who prefer to specify the actual skill number in the vector to take advantage of the tracking for VDN skill preferences.

EAS administration from CMS

CMS can be used to administer vectors as well as skills for agents and VDNs. The ACD Administration: Change Agent Skills CMS screen is used to display and modify the skills and levels assigned to an agent, as well as the assigned direct agent skill and call handling preference.

The ACD Administration: Change VDN Skill Preferences screen is used to request a VDN's skill preferences and to modify the VDN's skills.

The CMS Vector Contents screen is used to create and modify vectors. CMS supports the Call Vectoring commands that queue calls to the 1st, 2nd, or 3rd VDN skill.

Appendix I: Operation details for the route-to command

The **route-to** command can be programmed with or without coverage. The following table summarizes the operation of the **route-to** command for each of the destination types and conditions associated with the commands.

Switch route-to command operation

Condition	cov = n Any Step	cov = y Any Step[1]
Invalid Destination[2]	Goes to next step, else stop	Goes to next step, else stop
VDN **Extension**[3]		
Vector Assigned	Goes to new vector	Goes to new vector
Vector Has No Steps	Stop[4]	Stop[4]
Station Extension Idle (all appearances idle)		
CF-ALL Active or CF-DA Applies	Forwards if possible, else next step, else stop[4]	Forwards if possible, else coverage, else busy
Coverage		
• DA Applies	Rings idle app.	Coverage on DA
• All Applies	Goes to next step, else stop[4]	Coverage
• SAC Applies	Rings idle appearance	Coverage
• None of Above Applies	Rings idle appearance	Call delivered and is allowed to cover

Switch route-to command operation (continued)

Condition	cov = n Any Step	cov = y Any Step[1]
Station Extension Active (with idle 2-way app)		
CF-ALL Active	Forwards if possible, else next step, else stop[4]	Forwards if possible, else coverage, else busy
Coverage		
• DA Applies	Rings idle app (no DA timing)	Coverage on DA
• Ext Act Applies	Rings idle appearance	Coverage
• All Applies	Goes to next step, else stop[4]	Coverage
• SAC Applies	Rings idle appearance	Coverage
• None of Above Applies	Rings idle appearance	Call delivered and is allowed to cover
Station Extension Busy (no idle 2-way app)		
Extension in Hunt Grp (also see ACD Hunt Grp)	Queues if possible, else next step, else stop[4]	Queues if possible, else coverage, else busy
CF-ALL Active or CF-DA Applies	Forwards if possible, else next step, else stop[4]	Forwards if possible, else coverage, else busy
Call Waiting to Analog Sta Would Apply	Goes to next step, else stop[4]	Call waits
Coverage		
• Ext Act Applies	Goes to next step, else stop[4]	Coverage
• Ext Bsy Applies	Goes to next step, else stop[4]	Coverage
• All Applies	Goes to next step, else stop[4]	Coverage
• SAC Applies	Goes to next step, else stop[4]	Coverage
• None of Above Applies (or hunt, fwd, or cov destination is unavailable)	Goes to next step, else stop[4]	Busy tone given
Extension with Incompatible COR	Goes to next step, else stop.	Goes to next step, else stop.

Switch route-to command operation (continued)

Condition	cov = n Any Step	cov = y Any Step[1]
Terminating Extension Group		
All Members Idle	Rings idle appearance	Call delivered and is allowed to cover
A Member Active on TEG	Goes to next step, else stop[4]	Coverage, else busy
No Idle App on Any Member	Goes to next step, else stop[4]	Coverage, else busy
Hunt Group Extension		
Idle Agent	Rings idle appearance	Call delivered and is allowed to cover
No Idle Agent		
• Call cannot queue	Goes to next step, else stop[4]	Busy tone given
• Call can queue	Call is queued	Call is queued
Extension on Another Node (Uniform Dialing Plan - UDP DCS **or non-**DCS**)**		
Trunk available	Call delivered	Call delivered
Trunk not available	Goes to next step, else stop[4]	Queues if possible, else reorder
No DCS Buffer for Routing	Call delivered w/o DCS msg	Call delivered w/o DCS msg
Trunk Access Code (TAC**) Destination**		
Trk Grp No Dial Access	Goes to next step, else stop[4]	Routes to local attendant
Trunk Available	Call delivered	Call delivered
Trunk Not Available	Goes to next step, else stop[1]	Queues if possible, else reorder

Switch route-to command operation (continued)

Condition	cov = n Any Step	cov = y Any Step[1]
AAR/ARS FAC **Dest. (including Subnet Trkng)**		
Trk Grp No Dial Access	Tries next route	Routes to local attendant
Trunk Available	Call delivered	Call delivered
Other Routes Avail	Call delivered	Tries next route
All Routes Busy		
• No Pattern Queuing	Goes to next step, else stop[4]	Reorder tone given
• Queuing Assigned	Goes to next step, else stop[4]	Queues to pattern
Attendant Queue (dial 0)		
Idle Atnd	Rings idle appearance	Call delivered and is allowed to cover
No Idle Atnd		
• Not In Night Svc	Call is queued	Call is queued
• In Night Svc		
– Dest. assigned	Delivered to night service	Delivered to night service
– Not assigned	Call is queued	Call is queued
Individual Attendant Access		
Attendant idle	Rings idle appearance	Call delivered and is allowed to cover
Attendant busy	Queues if possible else Goes to next step, else stop[4]	Queues if possible, else Busy tone given
CAS **Attendant With Caller on Branch**		
RLT available	Rings idle appearance	Call delivered and is allowed to cover
All RLTs busy	Queues if possible, else next step, else stop[4]	Queues if possible, else busy tone

Switch route-to command operation (continued)

Condition	cov = n Any Step	cov = y Any Step[1]
Inter-PBX **Atnd Calling**		
Trunk Grp Controlled	Routes to local atendant	Routes to local attendant
Trunk Available	Call delivered	Call delivered
Trunk Not Available	Goes to next step, else stop[4]	Reorder tone given
Look Ahead Interflow (LAI) (feature active & routes over ISDN-PRI facility)[5]		
B-Channel Not Available	Goes to next step, else stop[4]	Queues if possible, else reorder
B-Channel Available and Receiving Switch:		
• Accepts	Interflow succeeds[6]	Call cut-through
• Rejects	Goes to next step, else stop[4]	Call gets busy/disconnect
Receiving Switch w LAI Acting as Tandem Sees from Remote Receiving Switch:		
• Call Accepted	Interflow succeeds[6]	Call cut-through
• Call Rejected	Goes to next step at receiving switch, else sending switch considers call rejected after 2-minute timeout	Call gets busy/disconnect
• **if interflow-qpos**	Determines if queued call is eligible for interflow	Determines if queued call is eligible for interflow

1. When the **with coverage** option is set to **y**, the call is removed from vector processing when the route-to step is reached, regardless of facility or remote switch availability. The call is taken out of any split queue, and any feedback, such as music or ringback, is removed, even if the destination is not available. If the call is subsequently rejected by the receiving switch vector, subsequent call treatment is defined by the rejection command (either busy or forced disconnect). The call is treated as though the destination is directly dialed (see footnote 3 for related information). This includes coverage, forwarding, treatments for calls that cannot be completed (busy reorder, and intercept) and displays. The answering station sees only caller name and number, unless the Display VDN for route-to DACS option is enabled (for more information, see Displaying VDN names for vector-initiated DACs on page 677). A call routed with an **adjunct routing link** command is treated the same way as a call that is routed using a **route-to with coverage y** command.

2. Invalid destinations include the following: empty (for example, zero collected digits) or invalid route-to destination number, unassigned extension number, incomplete number of digits for AAR/ARS pattern, non-AAR/ARS feature access code (FAC), maintenance busy station extension, COR of the VDN that prevents access (for example, origination restricted), FRL of a VDN that is lower than required for the AAR/ARS pattern access, no routes assigned to the AAR/ARS pattern, incompatible calling and destination partitions, ACTGA trunk group destination, or an off-net forwarding destination. If a TAC (trunk access code) destination is involved, and if the TAC is for a CO/FX trunk with a **route-to with coverage n** step, the digits entered must match a valid ARS analysis string. If not, the destination is considered invalid. For other trunk types with a **route-to number** or **route-to digits with coverage n** step, the step succeeds when the trunk is seized (that is, vector processing stops). For a **route-to with coverage y** step, the step succeeds if the TAC is assigned.
3. A call that routes to a VDN using the **route-to number with cov = y unconditionally** command behaves like a directly- dialed call instead of a VDN call. Therefore, the terminating station's display only shows the originating station information and does not show the VDN information (for other types of VDN calls, the terminating station would see the VDN name).
4. The interaction Stop means the following: vector processing is stopped, the call remains queued to a split, and the caller continues to hear feedback initiated by a previous step. In the case where the **route-to** command fails and processing stops (due to a busy station or trunk group destination), retry can be implemented in the vector. Retrying is accomplished by including an unconditional **goto** step as the last step to allow for a loop back to the **route-to** command. Use of an intermediate **wait-time** command step with appropriate feedback and delay interval is strongly recommended in order to reduce processor occupancy.
5. With one exception, any **route-to with cov= y** step that routes over ISDN-PRI facilities cancels Look-Ahead Interflow. The exception occurs when a call reaches a vector with coverage to a VDN. Calls that cover to a VDN will not be further forwarded or otherwise redirected. For covered calls, a **route-to** command with coverage set to **y** functions as though coverage were set to **n**. Thus, a **route-to with coverage y** will route covered calls with LAI over ISDN facilities if LAI is enabled.
6. On the sending switch, the call is removed from vector processing (that is, the call is taken out of any split queue and any feedback, such as music or ringback, is removed).

Appendix J: Advanced set command rules and applications

This appendix provides detailed descriptions and examples of the set command for advanced applications.

This section includes the following topics:

- Arithmetic operations on page 773
- String operations on page 776
- MOD10 operations on page 779

Arithmetic operations

The arithmetic operations are ADD, SUB, MUL, and DIV.

This section includes the following topics:

- About arithmetic operations on page 773
- Rules and considerations for arithmetic operations on page 774
- Invalid results for arithmetic operations on page 775
- The length parameter in arithmetic operations on page 775

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 = # (see Invalid results for arithmetic operations on page 775) • Less than 0 = # (see Invalid results for arithmetic operations on page 775) • 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

For more information, see:

- ADD examples on page 784
- SUB examples on page 785
- MUL examples on page 785
- DIV examples on page 786

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.

String operations

The string operations are CATL, CATR, and SEL.

This section includes the following topics:

- About string operations on page 776
- Rules and considerations for CATR and CATL operations on page 777
- Rules and considerations for SEL operations on page 778
- Invalid results for string operations on page 778
- Start and length parameters in string operations on page 779

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. See Rules and considerations for SEL operations on page 778.

	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 • Leading zeros allowed (007 = 007) • No invalid results. See Invalid results for string operations on page 778.	• none = 0 (null string) • # = none • Leading zeros allowed (007 = 007)	
Start and length parameters	See Start and length parameters in string operations on page 779.	N/A	N/A

For more information, see:

- CATL examples on page 787
- CATR examples on page 788

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. See Rules and considerations for CATR and CATL operations on page 777.

	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. See Invalid results for string operations on page 778.	• none = none • # = none • Leading zeros retained (007 = 007)	• none = 0 • # = 0 • Greater than 16 = 16 • Leading zeros ignored (007 = 7)
Start and length parameters	See Start and length parameters in string operations on page 779.	N/A	N/A

See SEL examples on page 788.

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.

MOD10 operations

This section includes the following topics:

- About the MOD10 operation on page 780
- Information about the MOD10 algorithm on page 780
- Rules and considerations for MOD10 operations on page 780
- MOD10 results on page 781
- Invalid results for MOD10 operations on page 782
- Start and length parameters in MOD10 operations on page 782

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 #. See Determining the number of digits on page 641.

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 about this algorithm see:

http://www.ee.unb.ca/tervo/ee4253/luhn.html

This Website describes how this algorithm works and provides a list of merchants and organizations that use this 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)	

	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. See MOD10 results on page 781.	• 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. See Start and length parameters in MOD10 operations on page 782.	N/A	N/A

See MOD10 operation examples on page 789.

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 for 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 has less or more of the number of digits specified by operand2. • There was 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 more than two digits in operand2.

Invalid results for MOD10 operations

During MOD10 operations, a variable or **digits** buffer may 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.

Appendix K: Set command examples

This appendix includes the following topics:

- Calculation examples on page 783
- Application examples on page 790

Calculation examples

This section provides examples of using the **set** command for calculations and includes the following topics:

- Arithmetic operation examples on page 783
- String operation examples on page 787
- MOD10 operation examples on page 789

Arithmetic operation examples

This section provides examples for the ADD, SUB, MUL and DIV arithmetic operators and includes the following topics:

- ADD examples on page 784
- SUB examples on page 785
- MUL examples on page 785
- DIV examples on page 786

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 A = 000123	123 - 1 = 122 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 (#).

Command	Result
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.
set A = 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.
set A = 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

This section provides examples for the CATL, CATR, and SEL string operators.

This section includes the following topics:

- CATL examples on page 787
- CATR examples on page 788
- SEL examples on page 788

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 Variable definition: • A = collect type, length = 4, start = 3 • B = 56789	56789 CATL 0123 = 012356789 = 2356 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 • B = 1234567890123	1234567890123 SEL 12 = 234567890123 = 4567890123 Ten digits were selected starting at position 3.
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.

Command	Result
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.
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.

Command	Result
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

This section provides examples of how to use the set command in business applications and includes the following topics:

- Validating numbers on page 790
- A 19-digit credit card validation on page 794
- Using bilingual announcements on page 794
- Collecting an account number on page 795
- Percentage routing using VDN variables on page 796
- Assigning ASAI UUI values on page 803

Validating numbers

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/Carte Blanche	300xxxxxxxxxx 305xxxxxxxxxx 36xxxxxxxxxxx 38xxxxxxxxxxx	14
American Express	34xxxxxxxxxxxx 37xxxxxxxxxxxx	15
VISA	4xxxxxxxxxxx 4xxxxxxxxxxxxxx	13, 16
MasterCard	51xxxxxxxxxxxxx 55xxxxxxxxxxxxx	16
Discover	6011xxxxxxxxxxx	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. 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.

See MOD10 results on page 781.

Using bilingual announcements

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

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, so
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, so
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 using VDN variables

This chapter provides an example of how to use VDN variables to implement percentage routing application. Note that starting withCommunication Manager 5.2 Percentage Allocation Routing using Policy Routing Tables (PRTs) assigned to VDNs is available. This is much simpler and direct approach to implement Percent Allocation in vectoring. For an example illustratration of how a complicated application can be implemented using variables and vectoring, see Percentage allocation routing on page 373.

The XYZ Company wants to:

- Route 25% of calls to the ABC Company in India
- Route 25% of calls to the DEF Company in China
- Route 50% of calls to the XYZ Company's local call center through XYZ Company's Avaya Communication Manager system

They can allocate call types into three groups of call handlers - one for India, one for China, and one for the local call center.

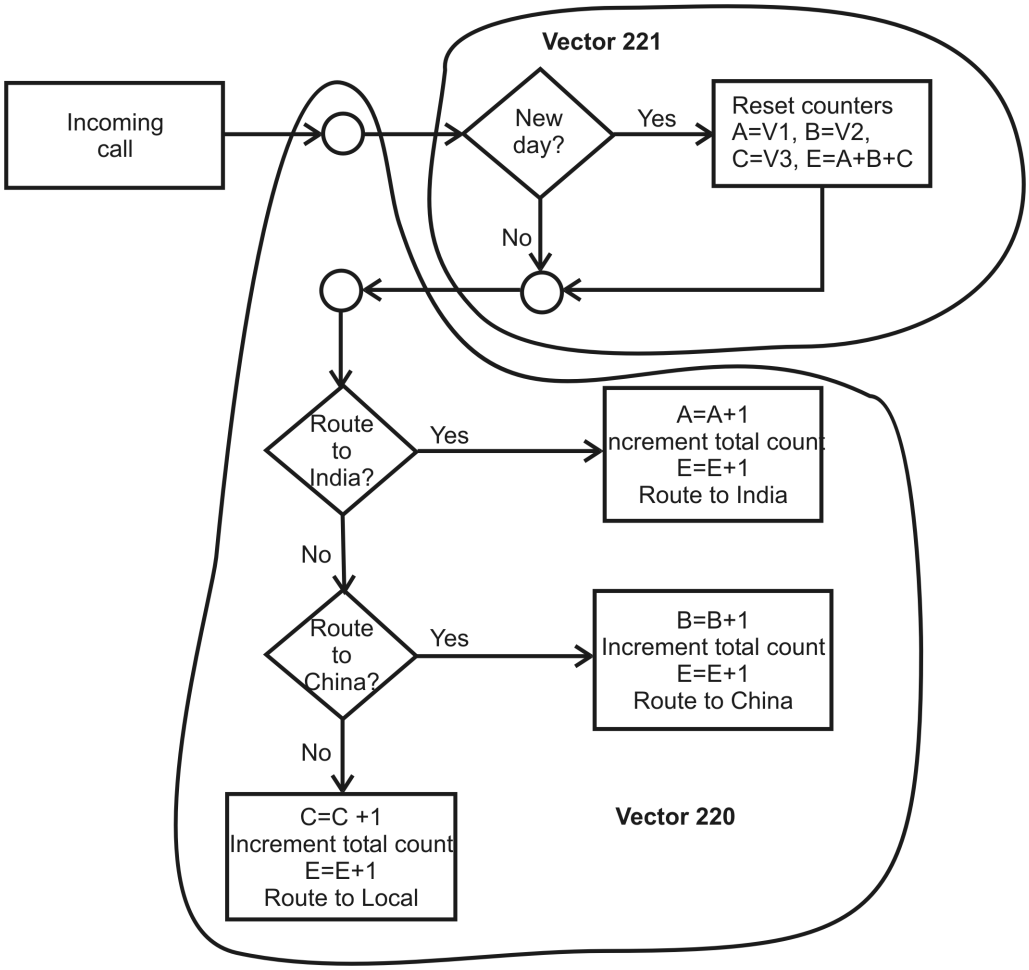
Overview of tasks

In this example, XYZ Company will use the **set** command and VDN variables to do the following tasks:

- Setup the Variables for Vectors and VDN variable definitions
- Use vector 220 as the primary vector to calculate percentages and route calls accordingly
- Use vector 221 as a subroutine vector to initialize the routing for the initial call that is performed every day
- Use VDN 2220 as the VDN that calls vector 220 with assigned VDN variable values associated with Percentage Routing. In this example, the following types of calls are routed to VDN 2220:
  - Toll-free number calls
  - Direct-inward-dialing calls
  - Calls screened by Interactive Voice Response (IVR)
  - Calls transferred by a local agent

Diagram of tasks

The following flowchart provides an overview of this example.



Legend:

- 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

To set up percentage routing:

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
                      Vector Number: 220


                 Allow VDN Override? n
                               COR: 1
                                TN: 1
                          Measured: external


        VDN of Origin Annc. Extension:
                            1st Skill:
                            2nd Skill:
                            3rd Skill:
```

```
change vdn 2220                                                  page 2 of 3
                     VECTOR DIRECTORY NUMBER

                  Return Destination:
               VDN Timed ACW Interval:
                     BSR Application:15
         BSR Available Agent Strategy: 1st -found

                Observe on Agent Answer?:n

             Display VDN for Route-to DAC?:n
VDN Override for ISDN Trunk ASAI Messages?:n

                     BSR Local Treatment?:n
```

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
```

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 3
                                  CALL VECTOR

             Number: 220         Name: Percent Route
Multimedia? 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 vector 221 @ step 1 if unconditionally
02 set D = A MUL 100 [D = calculated %, A = India routed calls]
03 set D = D DIV E [D = calculated %, E = total routed calls today]
04 goto step 11 if D < V1 [D = calculated %, V1 = India with an assignment of 25]
05 set D = B MUL 100 [D = calculated %, B = China routed calls]
06 set D = D DIV E [D = calculated %, E = total routed calls today]
07 goto step 15 if D < V2 [D = calculated %, V2 = China with an assignement of 25]
08 set D = C MUL 100 [D = calculated %, C = local routed calls]
09 set D = D DIV E [D = calculated %, E = total routed calls today]
10 goto step = 19 if unconditionally
11 set A = A ADD 1 [A = India routed calls]
12 set E = E ADD 1 [E = total routed calls today]
13 route-to number 97051001 with cov n if unconditionally
14 stop
15 set B = B ADD 1 [B = China routed calls]
16 set E = E ADD 1 [E = total routed calls today]
17 route-to number 97051002 with cov n if unconditionally
18 stop
19 set C = C ADD 1 [C = local routed calls]
20 set E = E ADD 1 [E = total routed calls today]
21 route-to number 97051003 with cov n if unconditionally
```

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 3
                                   CALL VECTOR

             Number: 221         Name: Reset New Day
Multimedia? 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 vector step 8 if F = G [F = day of week for first call, G = day of week1]
02 set F = none ADD G [F = day of week for first call, G = day of week1]
03 set A = none ADD V1 [A = India routed calls, V1 = India with an assignment of 25]
04 set B = none ADD V2 [B = China routed calls, V2 = China with an assignment of 25]
05 set C = none ADD V3 [C = local routed calls, V3 = local with an assignment of 50]
06 set D = A ADD B [D = total routed calls today, A = India routed calls, B = China
routed calls]
07 set E = D ADD C [E = total routed calls today, C = local routed calls]
08 return
```

1. 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.

Assigning ASAI UUI values

This section presents examples of using the set command assignment to alter ASAI UUI values.

Pass In-VDN Time to an Adjunct

The ABC Call Center needs to provide to the ASAI adjunct the amount time that a call has been in-processing in the call center system. The ASAI adjunct applies this 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 could spend a significant amount of time at previous call center locations before being interflowed to this local site. The application uses this in-processing time along with other information about the call to give higher priority to calls that have been waiting beyond a certain amount of time.

With CM 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 in-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, caller ANI, etc. The adjunct can now decide what VDN to send the call to via the route_select message.

Display Combined Wait Time and Account Number to Agent

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 6 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 CM 3.1. This can be in addition to collected digits already associated with the call which will be 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. Assume for this example the in-VDN time for the call is 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, then places the value (0025) in the ASAI UUI string (123456) starting at position 7. The resultant ASAI UUI contains 1234560025. This string is then seen by the agent when the UUI-INFO button is pressed.

Appendix L: Notifying callers of queue position example

This example explains how to notify callers of their position in queue without using a VRU or an IVR system.

Scenario

The 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 IVR or VRU equipment.

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).

Prerequisites

Before using the interflow-qpos conditional, consider the following 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 this field to 0 will not exclude calls at the top of the queue.

Tips for setting up the interflow-qpos conditional

Consider the following tips when setting up the interflow-qpos conditional.

- Use the same priority for queuing all of the calls; otherwise a lower priority call could get moved down in queue due to higher priority calls coming in later.
- If you have a release earlier than 3.0, the loop (steps 4 through 25) must fit in a single vector (within 32 steps) because of no subroutine or goto vector @step capability.
- If you need to add a test in the beginning, such as for working hours, use another 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
```

Appendix M: 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 on page 807
- Data passing on page 808
- VRU data collection on page 813
- Script execution on page 813
- Data return on page 814
- Script completion on page 816
- Switch data collection on page 817

Note:

If, during any phase of this call flow, a **converse-on** step is executed while the caller is in the split queue and an agent becomes available to service the caller, the VRU port is dropped, vector processing is terminated, and the calling party is immediately connected to the available agent.

Converse call placement

The first action taken by the **converse-on** step is to deliver the call to the converse split. Ringback tone is not heard by the caller. Any audible feedback supplied by vector processing remains until the VRU answers the call and all digits (if administered) have been outpulsed to the VRU. Vector processing is suspended. Callers remain in any non converse split queues, and they 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 kept.

Calls to busy converse splits are allowed to queue. The priority of the call in queue is administrable within the **converse-on** step. Again, any audible feedback supplied by vector processing continues until the call is answered by the VRU and any data is outpulsed. Calls to busy converse splits have either no queue or a full queue fail. For this scenario, a vector event is logged, and vector processing continues at the next vector step.

Whenever a **converse-on** step places a call to an auto-available split whose agents are all logged out, the call is not queued. Instead, the **converse-on** step fails, a vector event is logged, and vector processing continues at the next vector step.

Note:
Usually, this scenario occurs whenever the Voice Response Unit (VRU) goes down, the ports are members of an Auto-Available Split (AAS) and the Redirection on No Answer (RONA) feature has taken all the ports out of service.

The originator's display is not changed by the terminating or answering of a converse call. Also, whenever a call is delivered to a display station 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:
Even though AUDIX hunt groups are valid destinations for converse calls, they do not need 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 may 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.

This section includes the following topics:

- Using the pound sign on page 809
- How the outpulse sequence works on page 809
- Values administered for <data_1> and <data_2> on page 810
- Administering time delays on page 811
- When the VRU hangs up during data passing on page 812
- Ensuring robust operation of VRU data passing on page 812

Using the pound sign

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

Sending the pound sign prevents excessive delays and other problems caused by digit timeouts.

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:

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

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:

The use of this keyword is not recommended with multiple split queuing because any queue position value sent may not be meaningful. However, if the call is queued to multiple non converse splits, the value of the caller's queue position in the first non converse split is sent.

This data may be used by the voice information system to inform callers of their position in queue or to decide whether to execute a long or short version of a voice response script.

wait: This data type sends the expected wait time for a call in vector processing that is queued to at least one split. It is a value from 0 to 9999 seconds (variable length that is not padded with zeros) always followed by a 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.

Administering time delays

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 System Parameter-Features screen. These delays may range from 0 through 9 seconds, with a default of zero seconds for the converse first data delay and a default of two seconds for the converse second data delay. The delays may be needed to give the VRU time to invoke an application and allocate a touch-tone receiver to receive the passed digits.

- If <data_1> is not **none**, the converse first data delay timer starts when the call is answered by the VRU. Once the timer expires, the data_1 digits are outpulsed in-band to the VRU, followed by the end-of-string pound sign (#).
- If <data_2> is not **none**, the converse second data delay timer starts when the end-of-string pound sign from the first digit string is outpulsed. Once the timer expires, the data_2 digits are outpulsed in-band to the VRU, followed by the end-of-string pound sign.

No time delays are invoked when the keyword **none** is administered.

Note:
The outpulsing of digits is not heard by the caller.

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 the robust operation of the VRU data passing operation, be sure to implement the following recommendations:

- Include the prompt and collect command in the VRU script for each data field passed in the converse-on step.
- Administer each prompt and collect 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.

  Also, ensure no announcement is played in these prompt and collect steps.
- Ensure the first digit timeout 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 secs and the converse second data delay is 4 secs, the first digit timeouts for the two prompt and collect commands should be at least 5 and 9 seconds, respectively.
- Ensure the inter-digit timeout in the prompt and collect steps is at least five seconds.
- Administer the converse first data delay to give a VRU under a heavy load sufficient time to allocate a DTMF touch-tone receiver after answering the call.
- Administer the converse second data delay to give a VRU under a heavy load sufficient time to complete any tasks between the first and second prompt and collect command. For example, the VRU can invoke a new application if the first data field passed is used to identify the application script to be executed.
- In general, for converse-on steps 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 and prompt and collect. No announcement is programmed for the prompt and collect command, and the pound sign (#) is programmed as the end-of-string sign. If two sets of digits (that is, <data_1> and <data_2>) are passed by the switch, there will be two prompt and collect commands on the VRU to receive them.

If the first digit string (<data_1>) passed to the VRU is for application selection, the Avaya Interactive Response Script Builder exec command invokes the appropriate script. If a second digit string (<data_2>) is also used to pass an argument to this selected application, the first command in the 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.

The **Converse second data delay** is used to give the VRU time to invoke the selected application before the <data_2> digit string is outpulsed.

The application developer should ensure the administered converse first data delay and converse second data delay timers allow sufficient time for the VRU to successfully collect all outpulsed digits, even during periods of heavy call volume. Loss of digits from <data_2> is an indication the converse second data delay timer needs to be increased or the VRU timing values may need 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 may be displayed using the CALLR-INFO button.

The entire call is dropped if the caller abandons during the execution of a **converse-on** step.

Data return

This phase is optional and is in effect only if the application calls for the VRU to return information to the switch before returning control to vector processing.

Digits returned by the VRU are treated as dial-ahead digits. The rules for collecting and processing VRU-returned digits are identical to those for collecting and processing Call Prompting digits (see Call Prompting on page 237).

VRU data return is done in a manner similar to an analog transfer. Specifically, the VRU does an analog switchhook flash, outpulses DTMF digits, and then hangs up. If converse data is returned, the DTMF digits comprise two parts. The first sequence of digits is the converse data return feature access code administered on the Feature-Access-Codes 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, be sure to follow these recommendations:

- Set the analog flash timing to 600 msecs.
- Ensure DTMF tones last at least 70 msec and interdigit pauses last at least 50 msec. This results in an outpulsing rate up to 8.33 digits per second.
- (Avaya Interactive Response only) Use the converse_data external function to return data to the switch.
- Hang up line to switch after outpulsing digits. Assume that switch will wait between 1.2 and 1.5 secs to determine that the hang-up is a disconnect.

For applications involving VRUs other than Avaya Interactive Response VRUs, be sure to follow these recommendations:

- After the flash, ensure the VRU performs dial tone detection (stutter dial tone) for a sufficient period of time to ensure accurate detection (typically 0.6 to 1.0 secs) before outpulsing the converse data return feature access code.

- If no dial tone is received before the timeout, 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 dial tone is detected, ensure the digits of the converse data return feature access code are outpulsed.
- After the converse data return feature access code is outpulsed, the returned digits can be outpulsed without waiting for the second dial tone.
- After the VRU digits are outpulsed, the line to the switch is dropped.

Assuming an outpulse rate of 8 digits per sec (0.125 secs per digit), a 3-digit feature access code and stutter dial tone detection time of 0.6 secs, the maximum of 24 digits passed to switch should take about 6 secs (1.2 secs disconnect plus 8 secs plus 0.125 secs per digit).

The Call Classifiers required by the Call Prompting feature are not required for returning digits in-band from the VRU to the switch. Instead, general purpose 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 avoid the ten second timeout delay that occurs when the digits are collected. If the digit string being returned is multi-part (that is, to be collected by multiple **collect digits** steps), and if some of the parts are of variable length, the pound sign (#) can be used to terminate each of the variable length parts.

Note:

An asterisk (*) may be included as part of the converse data return code. However, since the asterisk is interpreted as a delete character by the switch, it makes little sense to use it as a returned digit. If it is used as such, all characters returned prior to the asterisk are discarded.

During the data return phase, the caller is temporarily put on hold. Music-on-hold, if administered, is suppressed. Since the caller hears silence during this phase, feedback should be provided to the caller as soon as possible after the **converse-on** step is executed.

Any touch-tone digits dialed by the calling party during the data return phase are discarded. These digits do not cause data corruption, and they are not collected as dial-ahead digits by the switch.

If an interdigit timeout occurs during the data return phase, the switch logs a vector event, keeps the digits already returned, drops the VRU, and reactivates vector processing at the next vector step.

If the timeout occurs before the converse data return code is returned, the operation is the same except that no discarded digits will be available.

Script completion

The VRU script returns control to vector processing on the switch by simply hanging up the line. In cases where no data is returned to the switch, this is done usually by executing the **quit** command. In cases where data is returned, this occurs whenever the VRU hangs 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 may 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 it may involve either a final message to a host or a final update to a local database. For this scenario, the VRU port (channel) is still associated with the running script even though there is no longer a voice connection.

From the switch point of view, the agent (port) is available for the next call. If a call is delivered to this port, the VRU does not answer the call until the previous script has completed. As long as the VRU script's after call work is short in duration, this poses no significant problem for the VRI feature. However, high volume VRI applications with lengthy after call work periods should be avoided, especially if such periods are so lengthy they approach the administered timeout period on the switch for the Redirection on No Answer (RONA) feature. In such a case, RONA might think the VRU ports are faulty and might therefore start to take these 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 has completed and vector processing is reactivated, the returned digits are collected and processed by vector commands in the usual manner. Since the digits must be collected by a **collect digits** command, data may be returned and processed only if the Call Prompting option is enabled.

The data returned can consist of multiple parts. For example, the VRU could return a stream of seven digits in which a single digit success/fail code is followed by a six-digit account code. For this scenario, the **converse-on** step would be followed by a sequence of vector steps including two **collect digits** steps. The first **collect digits** step would collect one digit and then check the result code; the second **collect digits** step would collect the six-digit account code.

Any touch-tone digits dialed by the calling party during the data collection phase are discarded, do not cause data corruption, and are not collected as dial-ahead digits by the switch.

If VRU data is returned, the calling party is able to touch-tone a response to a switch prompt only after the data collection phase is completed and another **collect digits** step is executed. This is true because each executed **collect digits** step does not allocate a 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.

Appendix N: Security issues

Call Vectoring can be integrated into the security of your switch. For example, Call Vectoring and Call Prompting can be used to help prevent unauthorized users from gaining access to the switch using the Remote Access feature. This section explains how this is done.

This section includes the following topics:

- Remote access on page 819
- EAS on page 821
- Limiting outside access using VDN COR restrictions on page 821
- Vector-initiated service observing on page 822
- Voice response integration on page 822
- Attendant Vectoring on page 823
- Remote logout of agent on page 823

Remote access

Abuse of remote access on the switch is one of the main methods by which unauthorized users obtain telephone services illegally. This section explains how a number of Call Vectoring features can be used to prevent unauthorized use of the remote access feature. No new development is required for any of these services.

Two methods are available:

- Front-ending remote access on page 819
- Replacing remote access on page 820

Front-ending remote access

This method gives authorized external callers a VDN extension to call instead of the remote access extension, which is kept private. The corresponding call vector can then implement a number of security checks before routing callers to the remote access extension. Routing can be done using a **`route-to number`** or **`route-to digits`** step.

The following advantages are possible using this method:

- Call Vectoring can introduce a delay before the dial-tone is provided to the caller. Immediate dial-tone is often one criterion searched for by a hacker's programs when the hacker is trying to break into a system.
- A recorded announcement declaring that the use of the switch services by unauthorized callers is illegal and that the call is subject to monitoring and/or recording can be played for the caller.
- Call Prompting can be used to prompt for a password. In such a case, the call is routed only if there is a match on the password.
- Use of the remote access extension can be limited to certain times of the day or certain days of the week.
- Real-time and historical reports on the use of the remote access feature can be accessed from CMS or from BCMS.
- Different passwords can be used on different days of the week or at different times during the day.
- Many VDNs that call the remote access extension can be identified. Accordingly, individuals or groups can be given their own VDN with unique passwords, permissions and reports. Any abuse of the system or security leak can then be attributed to an individual or a group.
- The caller can be routed to a VRU using the **converse-on** step where more sophisticated security checking, such as speaker recognition, can take place.
- Anyone failing any of the security checks can be routed to a security VDN that routes the caller to security personnel with a display set or to a VRU. Such a call would show security and possibly also the attempted password on the display. If the call is passed to a VRU, the VDN, the ANI and/or the prompted digits can be captured. CMS and BCMS reports on this security violation VDN will give information on how often and when security violations occur.

Replacing remote access

For this method, the remote access extension is not used. One or more VDNs are designed to access call vectors that can employ all of the security checks described in the previous section. The same reports and monitoring/recording capabilities described in the previous section can also be used. Instead of routing to the remote access extension, the vector collects digits from the caller and then routes to the given destination if there is a match on the password.

Again, multiple VDNs can be created for individuals or groups with different security checks and different permissions and/or restrictions. Destination numbers provided by callers can be screened by the vectors and denied if the user does not have permission to access that destination. For example, an individual user could be restricted to placing calls to numbers beginning with area codes 303 and 908.

EAS

With EAS, agent stations can be locked when they are not staffed. This is accomplished by assigning the station a Class of Restriction that does not allow outbound calls or it could be restricted from toll calls.

EAS agents have an optional password of up to nine digits to log in. This password is not displayed on DCP terminals when the agent is entering the password on the dial pad.

Limiting outside access using VDN COR restrictions

Routing calls through the communication server with Call Vectoring can raise some security issues. A VDN has a Class Of Restriction (COR). Calls processed by the vector carry the permissions and restrictions associated with the COR of the VDN.

For example, if a vector in the switch is written to collect digits, and then to route to the digits dialed, the restrictions on what calls can be placed are determined by the COR of the latest VDN. Also, checks can be made on the digits that are dialed, using **goto if digits** vector commands (for example, **goto if digits = 123**) to disallow routing to undesired destinations. The collect digits step can also be limited to collect only the number of digits required (for example, only collecting five digits for internal dialing).

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

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

Vector-initiated service observing

The following restrictions can be used with vector-initiated service observing to guard against unauthorized use.

- Call prompting commands can be used in service observing vectors to provide passcode protection, and to limit access to observing specific destinations or verified caller entered digits.
- Time of Day/Day of Week checks can be incorporated in service observing vectors.
- A vector can be created to be used exclusively for service observing.
- For a VDN to be observed as the result of a route-to command, the VDN must have a COR that allows it to be observed.
- The calling permissions of the COR assigned to the service observing VDN in conjunction with the can be observed settings of the COR assigned to the destination determine what agents, stations, or VDNS can be observed.

Voice response integration

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

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

Voice Response Integration Security Example

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

Attendant Vectoring

Security Violation Notification (SVN) referral calls can be directed to an attendant group. These are priority calls and, as such, cannot terminate to a VDN. However, when these calls are sent to the attendant group, they are treated as ordinary calls - priority does not apply to attendant group processing. So, these will be treated as normal attendant group calls and will be sent through vector processing.

Remote logout of agent

See Remote access on page 819 for issues associated with accessing the switch from a remote location.

Appendix O: Setting up a call center

This section includes the following topics:

- About setting up call centers on page 825
- Call Vectoring/non-EAS option on page 826
- Non-EAS Worksheet #1: Call center objectives on page 830
- Non-EAS Worksheet #2: Current split operation on page 831
- Non-EAS Worksheet #3: Customer needs on page 832
- Non-EAS Worksheet #4: Vector design on page 833
- EAS Worksheet #1: Call center objectives on page 834
- EAS Worksheet #2: Current split operation on page 836
- EAS Worksheet #3: Customer needs on page 837
- EAS Worksheet #4: Individual Agent Skills on page 838
- EAS Worksheet #5: Agent Skills on page 839
- EAS Worksheet #6: VDN Skill Preferences on page 840
- EAS Worksheet #7: Vector Design on page 841

About setting up call centers

Call center managers need some key indicators to measure ACD performance at their site. Usually, in setting up a call center, several factors involving call management are considered. The following list identifies and defines the most common of these factors, and it provides a typical question that might be asked. In addition, an insurance company example will be used to discuss the different options in this section.

Volume: Number of calls going in or out of the ACD. (How many calls did Split 1 answer?)

Productivity: Call volume per unit of time. (How many calls did Split 1 answer between 8 a.m. and 9 a.m.?)

Utilization: Overall use of the phone center. (What was my agent occupancy?)

Accessibility: Availability of lines and agents when customers call the ACD (this is an area that the Avaya CMS can probably most clearly define and help improve). (Were lines busy when customers called or did they have to wait too long?)

Quality of Service: Accuracy of information, a pleasant manner, responsiveness to caller concerns, successful completion of business, and efficient time utilization (not all measured directly by the CMS). (Was the caller given good service?)

This section explains how to set up a call center for customers with Call Vectoring and/or Expert Agent Selection (EAS).

Also see, Announcement recording tips for high traffic volume applications on page 541.

Call Vectoring/non-EAS option

To set up a call center that has Call Vectoring but not EAS, do the following:

1. Determine your call center's objectives. Think about how you want your call center to handle calls and also about what you want your call center to achieve. For more information, see Non-EAS Worksheet #1: Call center objectives on page 830.

   A company's basic goals are to increase profits and market share and to decrease costs. The purpose of setting up a call center is to monitor these goals using the CMS/BCMS reports. It is best to have more than one objective. (Some customers set and then live by only one objective.) Call center objectives must then be created to meet the goals. These objectives must be communicated to the split supervisor or to the administrator managing the call center.

   The following list provides an example set of call center objectives:

   - Establish the following measured entities:
     - Average Speed of Answer = 15 seconds
     - Abandon Rate <= 3%
     - Average Talk Time = 2 1/2 minutes
     - ACD calls per agent = 80 to 90 per day
     - Number of calls in queue = 6
     - Percentage of calls answered within the service level = 95%
     - Agent occupancy > 90%
     - Percentage of trunks busy < 3%
   - Generate revenue through the call center.
   - Train agents to back up each other.
   - Adequately train agents to provide service that meets customer expectations.
2. Review your existing operation and determine your call center needs (see Non-EAS Worksheet #2: Current split operation on page 831 and Customer/call center needs guidelines on page 827).

3. On the switch, assign a unique Hunt Group number and Call Distribution method to each caller need. This number will be your split number (see Non-EAS Worksheet #3: Customer needs on page 832 and Customer/call center needs guidelines on page 827).
4. Assign DNIS (Dialed Number Identification Service) (that is, the number dialed) as a Vector Directory Number (VDN) (see Customer/call center needs guidelines on page 827).

   As an option, you can assign one VDN for a main number and use Call Prompting to route the call to the proper split.

   The following table illustrates the guidelines given up to this point.

Customer/call center needs guidelines

Need	Split number (hunt group)	Call distribution[1]	VDN
New policy	1	UCD	555-6543
Questions about policy, Rate Quotes, Billing	2	UCD	555-6432
Spanish speaking for policy, service, and claims	3	DDC	555-6321
Claims	4	UCD	555-6210

1. Options include Direct Department Calling (DDC) and Uniform Call Distribution (UCD).

   Notice that this call center has only one split for all Spanish calls. However, resources permitting, you could create a New Policy split, a Service split, and a Claims split, each containing agents who speak Spanish. As an alternative, you could use one main VDN to point to a Call Prompting vector designed to route the calls to the splits.

5. On the switch, assign extensions to the agents' physical terminal locations (see the Extension/LoginID assignments table).
6. In CMS: Dictionary: Login Identifications, assign each agent a unique loginID. Agents are known to the CMS by the login ID. If assigned, reports refer to an agent by name, not by login ID.

   The following table illustrates the assignments described in the previous items:

Extension/LoginID assignments

Agent name[1]	Extension	LoginID[1]
Randy Tyler	1231	2000
Cathy Smith	1232	2001
Carla Silva	1238	2002

1. = assigned in the CMS Dictionary

Note:
When you are adding names to extensions on the switch, the agent name should be the same name as the loginID assigned in the CMS.

7. On the switch, assign agent extensions to splits. More than four splits can be assigned to an agent; however, the agent can log into a maximum of four splits. An agent assignment to splits can be changed in the CMS: ACD Administration: Move Extensions Between Splits if the agent is logged off.

   The following table illustrates the assignment of agent extensions to splits:

Agent extension/split assignments

Split (hunt group)	Agent extensions
1 - Sales	1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239
2 - Service	1231, 1232, 1234, 1238, 1239, 1240
3 - Spanish	1238, 1240, 1245
4 - Claims	1238, 1239, 1240, 1241, 1242

8. On the switch or in the CMS: ACD Administration: VDN Assignments, assign a vector to each VDN. A VDN can point to only one vector. However, a vector can have more than one VDN pointing to it.

   The following table illustrates VDN/vector assignments.

VDN/vector assignments

VDN	Vector
6543	1 (Sales)
6432	2 (Service)
6321	3 (Spanish)
6210	4 (Claims)

9. On the switch or in the CMS: ACD Administration: Vector Contents, write your vectors. For more information, see Non-EAS Worksheet #4: Vector design on page 833.

   Your vectors should match your call center objectives. To meet these objectives, you must make a number of relevant decisions (for example, you may decide how soon you want to enlarge an agent pool or what kind of treatment the caller should receive). If your VDN and vector reports do not satisfy your call center objectives, you must consider your alternatives (for example, you may deem it necessary to train agents or to increase the amount of time elapsed from when a call queues to one split and then to another split).

   The following lists indicate the actions produced by two different vectors:

   Actions Produced by Vector #1:

   a. Tell the caller to select one of the following prompts:
      - 1 = Sales
      - 2 = Service
      - 3 = Spanish
      - 4 = Claims
      - Nothing or 0 = Service
   b. Queue the call.
   c. Provide an announcement to the caller.

10. Actions Produced by Vector #2:

    a. Queue the call to the correct service at a medium priority.
    b. If no agents are available, provide a message and then play music.
    c. If the call is not answered within 10 seconds, provide a second message and then play music.
    d. If the call is not answered within 7 more seconds, queue the call to the Service split.
    e. If the call is not answered within 7 more seconds, queue the call to the Spanish split at a high priority.

    Note:
    A **`check split`** command queues the call to up to three splits if the conditions are met. If the conditions are not met, the **`check split`** command may not get read again (if the vector step in which it appears is not executed again).

11. In the CMS: Dictionary, assign names to the splits, VDNs, and vectors.

12. Once your system is up and operational, you will need to monitor it to ensure you are meeting your call center objectives. The call management system can be used to monitor many of your objectives. Some objectives will need to be monitored and have adjustments made in real time. For example, if the number of calls waiting, average speed of answer, or percent answered within a service level is not meeting your objectives, you might want to immediately move some agents, direct calls to another vector, or look-ahead interflow some calls. Other items such as agent occupancy and percent all trunks busy may only need to be monitored daily to look for trends.

Non-EAS Worksheet #1: Call center objectives

What are my call center objectives?

Non-EAS Worksheet #2: Current split operation

Split _____

Primary Backup _____ **Secondary Backup _____** **Tertiary Backup _____**

List your customer/caller needs and your agent knowledge levels for this split.

1. _____

2. _____

3. _____

4. _____

5. _____

6. _____

Split _____

Primary Backup _____ **Secondary Backup _____** **Tertiary Backup _____**

List your customer/caller needs and your agent knowledge levels for this split.

1. _____

2. _____

3. _____

4. _____

5. _____

6. _____

Non-EAS Worksheet #3: Customer needs

Call center needs	Split number (hunt group)	Call distribution	VDN

Non-EAS Worksheet #4: Vector design

Vector #	Name:	Description:
Assigned VDNs:		
Assigned Trunk Groups:		

EAS Worksheet #1: Call center objectives

What are my call center objectives?

What are my call center objectives?

EAS Worksheet #2: Current split operation

Split: _____

Primary backup:		Secondary backup:		Tertiary backup:	

Customer/caller needs and agent skill sets in this split:	Is agent expertise available? (y/n)	Do you want to separate skill sets with EAS? (y/n)
1.		
2.		
3.		
4.		
5.		
6.		

Split: _____

Primary backup:		Secondary backup:		Tertiary backup:	

Customer/caller needs and agent skill sets in this split:	Is agent expertise available? (y/n)	Do you want to separate skill set with EAS? (y/n)
1.		
2.		
3.		

4._____ _____ _____

5._____ _____ _____

6._____ _____ _____

EAS Worksheet #3: Customer needs

Call center needs	Split name	Split number (hunt group)	Call distribution	VDN

EAS Worksheet #4: Individual Agent Skills

				Logged-in coverage			Logged-out coverage		
Agent name	**Login ID**	**Direct agent?**	**COR[1]**	**Point 1**	**Point 2**	**Point 3**	**Point 1**	**Point 2**	**Point 3**

1. Class of restriction

EAS Worksheet #5: Agent Skills

Agent Name	Login ID	1st skill	2nd skill	3rd skill	4th skill	COR[1]

1. Class of restriction

EAS Worksheet #6: VDN Skill Preferences

VDN extension	VDN Name	COR[1]	Skill preferences			Vector
			1st skill	2nd skill	3rd skill	

1. Class of restriction

EAS Worksheet #7: Vector Design

Vector #	Name:	Description:
Assigned VDNs:		
Assigned Trunk Groups:		

Appendix P: Converting a call center to EAS

The procedures in this section provide guidelines for upgrading a call center from a non-EAS ACD environment to an EAS ACD environment. The primary activities involved in this conversion are:

- Before you begin on page 843
- Step 1: Pre-EAS cutover administration for the system on page 844
- Step 2: Pre-EAS cutover administration for the CMS on page 848
- Step 3: Pre-EAS cutover administration for AUDIX on page 848
- Step 4: Pre-EAS cutover administration for ASAI on page 848
- Step 5: EAS cutover on page 849

Before you begin

Before the transition to EAS takes place, decisions must be made concerning:

- Which area of the current dial plan is to be used for EAS agent login IDs. EAS agent login IDs cannot conflict with already defined extension numbers (for example, an EAS agent login ID cannot be the same as a station extension number).
- Whether the current incoming call routing through VDNs and vectors will remain the same after the EAS upgrade, or whether new VDNs and/or vectors are required.
- How incoming call traffic is to be handled during EAS cutover.

Once these decisions are made, the pre-EAS cutover administration activities can be started in preparation for the conversion of the call center to EAS.

Note:
Even though EAS administration changes are being made, non-EAS ACD call handling and agent operations are unaffected. When the cutover to EAS is completed, all non-EAS ACD call handling and agent operations will cease.

Step 1: Pre-EAS cutover administration for the system

Perform the following activities to prepare the DEFINITY for the cutover to EAS:

1. At administration terminal display the System-Parameters Customer-Options screen and verify that the **ACD**, **Expert Agent Selection**, and **Vectoring (Basic)** fields are set to **y**. If you will be using the increased capacities of **EAS-PHD**, verify that this option is set to **y**.
2. If you haven't already done so, display the Feature Access Code screen and administer the ACD Agent Feature Access Codes (for example, login, logout, and auto-in) as required for agent operations.
3. Using the CDR System Parameters screen, administer whether the EAS login ID, or the terminal extension where the EAS agent is logged in, should appear on CDR reports by setting the **Agent Login ID - Record** field to **y** or **n**. This field affects the CDR tracking for incoming calls only; outgoing calls made by a logged-in EAS agent are always recorded by CDR using the agent's login ID.
4. If new VDNs are desired for the EAS environment, using the VDN administration screen, administer the VDN Skills and other VDN information for the VDNs used to route calls to EAS agents. If the 1st, 2nd, and/or 3rd skill options are to be used in the vectors or for Avaya CMS tracking associated with these VDNs, then administer the 1st Skill, 2nd Skill, and 3rd Skill fields as required.
5. If new vectors are desired for the EAS environment, using the Vector administration screen, administer the vectors associated with the VDNs added in the previous step. As part of the EAS feature, the 1st, 2nd, or 3rd skill options may be used in the vector step fields where a skill hunt group is entered (rather than entering an absolute skill hunt group number). Refer to Expert Agent Selection on page 455 for more information concerning vector programming for the EAS feature.
6. If new skill hunt groups are required, using the Hunt Group administration screen, administer the desired skill hunt groups.

   Note:
   Entering a **y** in the **Skills** field automatically causes the **ACD** and **Vector** fields to be set to **y**. With EAS optioned, it is not possible to administer members for a skill hunt group.

7. If coverage paths are to be administered for EAS agents, using the Coverage Path administration screen, set up the coverage paths to be assigned to EAS agent login IDs.

   Note:
   There is a difference between coverage treatment for an EAS direct agent call (where both the calling party and called login ID have the **Direct Agent Calling COR** option set to **y**), and an EAS personal call (where either the calling party or called login ID does not have the **Direct Agent Calling COR** option set to **y**).

Note:

A direct agent call is routed to an EAS agent as an ACD-type call, and therefore its coverage behavior is considerably different from the coverage for a normal station call. For example, if an EAS agent is not available for an ACD call when a direct agent call is made to that agent, the direct agent call is queued to the direct agent skill administered on the Agent Login ID screen (after initiating a ring-ping and then fluttering the active work-mode button at the agent's terminal). On the other hand, a personal call to an EAS agent is not an ACD-type call, and its coverage behavior is similar to the coverage treatment for a call to a station extension. For example, a personal call to an EAS agent who is busy on any call appearance will result in the call being sent to an idle call appearance at that agent's terminal.

Depending on the type of coverage criteria desired for direct agent and personal calls to EAS login IDs, administer the desired coverage path criteria as follows:

- To provide coverage for a non-ACD personal call to an EAS login ID when the agent is logged in and active on any call appearance, set the **Active** coverage criteria to **y**. The **Active** coverage criteria does not apply for a direct agent call to an EAS login ID.
- To provide coverage for calls to an EAS login ID when the agent is logged out, set the **Busy** coverage criteria to **y**. Busy coverage will also be applied to a logged-in EAS agent when either of the following conditions occur:
  - A direct agent call is made to the EAS agent and there are no available queue slots in the agent's first skill hunt group;
  - A personal call is made to an EAS agent and the agent's station has no idle call appearances.
- To provide coverage for calls to an EAS login ID when the agent is logged in but does not answer after a certain number of ring cycles, set the **Don't Answer** coverage criteria to **y**, and enter a number for the desired ring time-out in the **Number of Rings** field.
- To provide immediate coverage for calls to an EAS login ID whether the agent is logged in or logged out, set the **All** coverage criteria to **y**.
- To provide coverage for calls to EAS login IDs when the call is to a logged-in agent who has activated the Send All Calls or Go To Cover features, set the **DND/SAC/Goto Cover** coverage criteria to **y**.

8. Up to three coverage paths for different types of call coverage criteria may be linked together by administering the Next Path Number field on the Coverage Path screen. If the criteria for the first coverage path are not met, then the criteria for the second linked coverage path are checked by the system, and so on. This can be used to provide different coverage paths for calls to an EAS login ID when the associated agent is logged in or logged out.

Note:
If a call to a logged-in EAS login ID is a personal call and coverage goes into effect, the redirected call maintains a simulated bridged appearance at that agent's terminal. The agent may still answer the call after redirection takes place by going off-hook on this line appearance. However, if a call to a logged-in EAS login ID is a direct agent call, the redirected call does not maintain a simulated bridged appearance at the agent's terminal. The agent may not then answer the call after redirection takes place.

Note:
If the Redirection on No Answer (RONA) feature is enabled for skill hunt groups, set the ring time-out interval for the RONA feature such that it does not conflict with the coverage ring time-out criteria.

9. If coverage paths are administered for EAS login IDs, using the Feature-Related System Parameters screen, set the Coverage - Subsequent Redirection No Answer Interval field to the desired ring time-out interval for calls routed to administered coverage points.

   Note:
   EAS login IDs may be administered as coverage points for a coverage path, and this administered coverage no-answer interval applies to direct agent or personal calls made to these coverage points as well.

10. Using the COR administration screen, set the **Direct Agent Calling** field to **y** for any COR to be assigned to a trunk or station user who may initiate a direct agent call to an EAS agent, or to be assigned to an EAS login ID that may receive direct agent calls.
11. If EAS agent login ID passwords are to be administered, using the Feature-Related System Parameters screen, set the **Minimum Agent-LoginID Password Length** field to the desired number of minimum password digits (0 to 9) which must be specified when agent passwords are administered using the Agent Login ID screen. The total number of digits which may be assigned to a password is between the value of the **Minimum Agent-Login ID Password Length** field and 9 digits. If a password is administered for an agent, this password must be entered in addition to the agent's login ID to log in.
12. Using the Agent Login ID screen, add the desired EAS login IDs to be associated with human agents, AUDIX ports, and/or AAS (Auto-Available Split) VRU ports. For human agents, the following fields are administered:
    - Name
    - COR
    - **Coverage Path** (optional)
    - **Security Code** (optional for Demand Print feature)
    - **LWC Reception** (optional)
    - **AUDIX Name** (for G3r only, if the **LWC Reception** field is set to **audix**, or if administered coverage path for the agent has an AUDIX coverage point)
    - **Password** (optional)
    - **Skills** - Skill Level (for at least one skill)

13. For AUDIX and AAS VRU port extensions, when these ports are associated with ACD-type hunt groups, these extensions must be associated with skill hunt groups as part of the cutover to EAS. Additionally, for skill hunt groups used for AAS ports, the **AAS** field must be set to y for these hunt groups before any EAS AAS agents can be administered.

    Note:
    AUDIX hunt groups do not need to be vector-controlled. This allows for ASAI monitoring of the skill hunt group.

    If AUDIX port extensions (such as for the Embedded AUDIX product) are not associated with an ACD hunt group, no administration is required for these ports as part of the cutover to EAS. For the AUDIX and/or AAS ports that are associated with ACD hunt groups, add EAS agent login IDs for these ports, where only the following fields need to be administered:

    - Name
    - COR
    - **Coverage path** (optional)
    - **AUDIX** (set to **y** for AUDIX ports)
    - **AAS** (set to **y** for AAS VRU ports)
    - **Port Extension** (set to the AUDIX or AAS port extension administered in the non-EAS environment)
    - **Skills** - Skill Level (where a single skill is entered for the skill hunt group associated with the AUDIX or AAS station ports)

14. Using the Station Administration screen, administer any stations to be used by EAS agents and the desired work-mode buttons for each station (if not already administered).

    Note:
    If stations are already administered with work-mode buttons associated with splits, it is not necessary to readminister these buttons for EAS. If new work-mode buttons are added to a station, it is not possible to enter data in the Grp field after EAS is enabled except for the AUX work-mode button (which may be administered with a hunt group number if the entered hunt group is a non-ACD hunt group).

    Note:
    Also, if more than one set of work-mode buttons is administered on a station set, these buttons may be left as is until after the cutover to EAS. After the cutover, it is desirable to remove the extra sets of work-mode buttons since EAS requires only one set of work-mode buttons for agent operations.

Step 2: Pre-EAS cutover administration for the CMS

See *Avaya CMS Administration* for the procedures used to configure the CMS for the EAS feature. This document is also helpful in providing overall planning strategies for implementing call center operations.

Step 3: Pre-EAS cutover administration for AUDIX

If EAS agents' login IDs are administered with coverage paths that route to an AUDIX coverage point, the login IDs for these agents must be administered using the AUDIX console so that the caller will hear the appropriate AUDIX voice responses for calls made to EAS login IDs.

Note:
On the G3r, the **AUDIX Name** field on the Agent Login ID screen must be set to the correct AUDIX name to provide proper AUDIX coverage of calls made to EAS agents, or to leave LWC messages for EAS agents if LWC reception to AUDIX is set up for the agents' login IDs.

Refer to Step 1: Pre-EAS cutover administration for the system for information on how to administer EAS login IDs for AUDIX port extensions on the DEFINITY.

Step 4: Pre-EAS cutover administration for ASAI

With ASAI-based applications for call center operations, the cutover to EAS may necessitate an upgrade of the ASAI-related application software on the adjunct. With OCM (Outgoing Call Management), the upgrade to EAS requires that specialized vectors be administered to handle the launching of calls from VDNs (as opposed to the non-EAS environment where OCM calls are launched from splits). For more information on the procedures to convert an ASAI application for EAS, see *Avaya Communication Manager CallVisor ASAI Technical Reference*.

Step 5: EAS cutover

After you complete all pre-EAS activities, you can activate the EAS feature. Prior to the EAS cutover, make a backup tape of the current DEFINITY translations for possible recovery in case you have difficulty during cutover. In particular, since the transition to EAS results in the removal of all ACD hunt group members, the pre-EAS tape backup could save time when restoring non-EAS hunt group translations if the cutover to EAS is not completed.

Block incoming ACD call traffic to prevent the queuing of new ACD calls to existing splits during the cutover from the non-EAS to EAS environment. Block new incoming calls by doing one or both of the following tasks:

- Busy out the appropriate trunk groups.
- Set the first vector step for actively-used incoming call vectors to the busy step. Use the Vectoring screen.

Once this is accomplished, perform these activities:

1. Make sure all EAS agents are logged out of all splits. If CMS or BCMS is operational, you can use the CMS real-time reports for splits or the **`mon bcms split`** command to identify the terminals where agents could still be logged in.
2. Issue the **`busy mis`** command at the administration terminal to busy-out the CMS link.
3. Issue the **`busy link n`** command at the administration terminal to busy-out any AUDIX switch-to-adjunct links.
4. Issue the **`busy station x`** command at the administration terminal to busy-out any AAS ports.
5. Using the Hunt Group screen, convert any ACD splits to skill hunt groups by setting the **Skilled** field to **`y`** for these hunt groups.
6. Using the Feature-Related System Parameters screen, set the **Expert Agent Selection (EAS) Enabled** field to **`y`**, and set the **Adjunct CMS Release** field.
7. If CMS installed, release the link to the CMS by entering the **`release mis`** command at the administration terminal.
8. Do the following CMS administration tasks:
   a. Check the CMS for EAS authorization.
   b. Change the switch setup in CMS to track the data and to establish the communications link between CMS and the communication server.

   For more information, see *Avaya CMS Administration*.
9. Inform the on-site agents that they can log into their terminals using the EAS login procedure and become available to receive ACD calls using the auto-in or manual-in work-mode operations.

10. Using the Vectoring screen, restore any vector steps temporarily changed to busy to their previous vector step format. The vector steps were temporarily changed to busy to block incoming calls.
11. Using the Trunk Group Administration screen, if the routing for incoming trunks is to be changed to EAS-related VDNs, administer the **Incoming Destination** field for any trunk groups to the appropriate VDN extension number.
12. Issue the **`release station x`** command at the administration terminal to release any AAS ports (where the EAS login ID associated with each AAS port will be automatically logged in).
13. Issue the **`release link n`** command at the administration terminal to release any adjunct AUDIX links (where the adjunct will cause the associated ports to be logged in).

At this point, the cutover to EAS is complete. It is recommended that a backup of the switch translations be performed as soon as possible after the cutover to preserve the EAS-related administration changes. Also, if agent stations are administered with multiple sets of work-mode buttons, it is recommended that all but one set of work-mode buttons be removed from these stations. Also, multiple queue lights are required for EAS.

Appendix Q: Feature availability

This section lists available vectoring enhancements. For a detailed description of any item see the referenced section of this guide.

Vectoring (G3V4 Enhanced): Provides the following additional capabilities:

- The ability to specify a priority level with the oldest-call-wait conditional on the **`check`** and **`goto`** commands. For more information about these commands, see Call Vectoring commands on page 521.
- The use of enhanced comparators (<>, >=, and <=) with the **`goto`** and **`route-to`** commands as well as use of **none** as an entry for digits checking, and active or latest VDN thresholds for indirect VDN references. For more information, see these commands in Call Vectoring commands on page 521.
- The use of the interflow-qpos conditional with the **`goto`** and **`route-to`** commands to achieve FIFO or FIFO-like call processing. For more information, see Look-Ahead Interflow (LAI) on page 273.
- The use of wildcards in digit strings for matching on collected digits and ANI or II-digits. For more information, see Call Vectoring commands on page 521.
- The use of Vector Routing Tables for matching on collected digits and ANI or II-digits. For more information, see Vector Routing Tables on page 243 or ANI /II-digits routing and Caller Information Forwarding (CINFO) on page 169.
- Multiple Audio/Music Sources for use with the wait-time command. For more information, see Rolling Average Speed of Answer (ASA) on page 266.

Vectoring (G3V4 Advanced Routing): Provides the following additional capabilities (Vectoring [G3V4 Enhanced] must also be enabled):

- Rolling Average Speed of Answer (ASA) Routing. For more information, see Rolling Average Speed of Answer (ASA) on page 266.
- Expected Wait Time (EWT) Routing. For more information, see Expected Wait Time (EWT) on page 259.
- VDN Calls Routing. For more information, see VDN Calls on page 269.

Vectoring (ANI/II-Digits Routing): Provides the following additional capabilities (Vectoring [G3V4 Enhanced] must also be enabled):

- ANI Routing. For more information, see ANI /II-digits routing and Caller Information Forwarding (CINFO) on page 169.
- II-Digits Routing. For more information, see II-digits routing on page 174.

Vectoring (CINFO): Provides the following additional capabilities (Call Prompting must also be enabled):

- The ability to collect ced and cdpd from the network. For more information, see Caller Information Forwarding on page 180.

Vectoring (Best Service Routing): Automatically compares splits or skills in ACD environments to find the one that can provide the best service to each caller. BSR can operate at a single site, or it can be used with Look-Ahead Interflow to integrate a network of geographically distributed locations into a virtual call center. For more information, see Best Service Routing (BSR) on page 297.

Vectoring (Best Service Routing) without LAI enabled (single-site 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, the wait-improved conditional gives you the ability to make any subsequent queuing conditional on the improvement in EWT as compared to the call's EWT in its current queue.

Vectoring (Best Service Routing) with LAI enabled (multi-site 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, the wait-improved conditional gives you the ability to make any subsequent queuing conditional on the improvement in EWT as compared to the call's EWT in its current queue.

Enhanced information forwarding provides the transport of existing call information and new call information such as Universal Call ID and Best Service Routing. For more information, see Information Forwarding on page 189.

Timed ACW provides the ability to assign a timed ACW interval to a VDN. For more information, see Vector Directory Number on page 38.

Vectoring (Holidays): Simplifies vector writing for holidays. It is designed for customers who need to reroute or provide special handling for date-related calls on a regular basis.

This feature provides the user with the capability to administer 99 different Holiday Tables, then use those tables to make vectoring decisions. Each table can contain up to 15 dates or date ranges. All of this can be done in advance to ensure seamless call routing over holidays when staffing is reduced or call centers are closed.

Vectoring (Variables): Creates variables that can be used in vector commands to:

- Improve the general efficiency of vector administration.
- Provide increased manager and application control over call treatments.
- Allow you to create more flexible vectors that better serve the needs of your customer and call center operations.

The vector variables are defined in a central variable administration table. Values assigned to some types of variables can also be quickly changed by means of special vectors, VDNs or FACs (Feature Access Codes) that you administer specifically for that purpose.

Different types of variables are available to meet 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, an administered variable can be reused in many vectors.

Vectoring (3.0 Enhanced): Provides the following additional capabilities:

- An increase in the number of auxiliary (AUX) work time reason codes from the previous limit of 10 codes (0-9) to 100 codes (0-99) as a system option.
- For S8700, S8710, and S8500 Communication Manager systems, an increase in the capacity for agents logged into the same skill from a maximum of 1500 agents to a maximum of 3000 agents.
- The ability to set the following ACD options for individual agents:
  - MIA Across Skills
  - ACW Agent Considered Idle
  - AUX Work Reason Code Type
  - Logout Reason Code Type
- The addition of the Forced Agent Logout from After Call Work (ACW) feature that automatically logs out an Expert Agent Selection (EAS) agent who spends too much time in ACW mode. The timeout period is specified on a per system basis and on a per agent basis. The timeout is reported with a customer-assignable reason code set on a system basis.

  For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .
- The addition of the Location Preference Distribution feature that tries to route incoming Automatic Call Distribution (ACD) calls to agents located at the same location as the incoming trunk on which the call originated whenever possible. 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, service level, and so on.

  For more information, see *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

- The use of locally-sourced music and announcements. This feature allows call centers to use any or all of their 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 *Avaya Aura™ Call Center 5.2 Automatic Call Distribution (ACD) Reference* .

- The use of vector subroutines that use common vector programs. Subroutines 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 use of VDN variables that allow a single vector to support multiple VDNs.
- The addition of the **`return`** command that can return vector processing to the step following the **`goto vector`** command after a subroutine has processed.
- The addition of the **`set`** command that can do 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
- 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. 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
- An increase in Holiday Vectoring that allows administrators to administer 99 different Holiday Tables instead of only 10 to make vectoring decisions.
- The addition of the VDN Time-Zone Offset feature that is designed for call centers with locations in different time zones. The administrator can program a single vector that handles each time zone based on the active VDN for the call.

Appendix R: Improving performance

This section provides recommendations on how to write vectors that promote favorable performance practices. This section includes the following topics:

- About improved performance on page 855
- Looping examples on page 856
- Other examples on page 860

About improved performance

Improved performance depends on the following basic principles:

- Minimize the number of vector steps to process a call.
- Avoid vector steps which have a substantial probability of failure, such as:
  - Calls made outside of business hours
  - Queues to groups with less than desirable resources or characteristics.

The most wasteful use of processing resources is frequently caused by inefficient looping. For example, performance could be compromised when a vector loops through steps too often. This is especially true with long queue times.

Some examples with looping are discussed and recommendations are given on how to maximize performance. They are:

- Audible Feedback
- Look-Ahead Interflow
- Check

Examples other than looping are also discussed. They are:

- After Business Hours
- Look-Ahead Interflow

All looping examples in this section use only loops within a single vector. It is important to also be aware of looping to other vectors through the use of vector chaining. The same principles can be extrapolated from the looping examples. Creating a flow diagram is often helpful for identifying looping errors.

In addition to the example vectors, tables rating the relative performance costs of specific vector commands are also included.

Note:
Remember to test vectors for performance in addition to call flow.

Looping examples

This section includes the following topics:

- Audible feedback on page 856
- Look-Ahead interflow on page 857
- Check on page 859

Audible feedback

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

Also see Announcement recording tips for high traffic volume applications on page 541.

The first example repeats the, *All representative are busy. Please hold* 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 next example repeats the announcement only every 60 seconds, thus 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
```

The next example adds a second announcement, *All representatives are still busy. Please hold* in addition to the initial announcement and repeats the second announcement less often (every 120 seconds), thus improving performance again.

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 while processing the call. Assumption is that the first announcement is 3 seconds long and the second announcement is 4 seconds long.

Approximate number of vector steps executed for the audible feedback examples

Initial conditions	Example: 10-second announcement interval	Example: 60-second announcement interval	Example: Follow-up announcement
An agent is available in split 1	1	1	1
Queueing time of 5 minutes	70	15	9

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

Look-Ahead interflow

Recommendation 1: Use the **interflow-qpos** conditional to achieve FIFO (first in, first out) or near-FIFO call processing. For more information, see Look-Ahead Interflow (LAI) on page 273.

Recommendation 2: If you do not have the **interflow-qpos** conditional, add a wait period between successive look-ahead interflow attempts and make the waiting period as long as feasible.

The following example continuously attempts a look-ahead interflow 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 example shown above adds a delay so that the look-ahead interflow attempt occurs only every 10 seconds.

Example: look ahead with 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 next example increases performance even more by increasing the delay between look-ahead interflow attempts to 30 seconds.

Example: look ahead with 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 while processing the call. Assumption is that the announcement is 5 seconds long.

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

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

Check

Recommendation: When using check commands to queue a call to backup splits, ensure that an adequate amount of time has elapsed before checking the backup splits again.

Note:
With the Expected Time Wait Time feature, the style of programming used in this example is not optimal. The best approach is to use the Expected Time Wait feature to locate the most appropriate split for the call and queue it there.

The next example checks backup splits continuously as long as the call is in queue.

Example: Continuous check

```
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
```

The next example adds a delay of 10 seconds to ensure that some time has elapsed before checking the backup splits again.

Example: Check with 10 second delay

```
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
```

Since the agent availability status may not be likely to change every 10 seconds, it may make sense to increase the wait time to 30 seconds, as shown in the example in The following example.

Example: Check with 30 second delay

```
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.

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
Queueing time of 5 minutes	up to 1,000	190	65

When a call is queued for 5 minutes, the number of vector steps drops dramatically when a delay is added before checking the backup splits again, and drops even more 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.

Other examples

This section includes the following topics:

- After business hours on page 861
- Look-ahead interflows on page 861

After business hours

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 the call is made 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 is made 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 is queued after business hours and the call is terminated only when the caller hangs up. As shown in the second example, it is more economical to test for business hours before queuing a call.

Look-ahead interflows

Recommendation: When using a look-ahead interflow, first test to see if the receiving office is open for business.

The scenario is a sending switch in Los Angeles, with office hours from 8:00 AM to 5:00PM (8:00-17:00) PST and the receiving switch is in New York, with office hours from 8:00 AM to 5:00PM EST (5:00-14:00 PST). There is a three hour difference between the two switches

The following example routes calls to the New York switch. If there are no agents available at the Los Angeles switch, it is possible for calls to be interflowed during hours that the agents in New York are not available, thus doing unnecessary processing.

Unconditional Look-ahead interflow

```
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 switch is open before requesting a queue to the New York switch, thus avoiding unnecessary processing.

Look-ahead interflow with time-of-day 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
```

The next example can be used if you have Advanced Routing optioned. In this case, the Expected Wait Time feature may be used to determine whether it is worthwhile placing a look-ahead interflow call attempt.

Look-ahead interflow with expected wait time and time-of-day 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 shown above, note that there is no reason to attempt an interflow if the call will be answered quickly at the main switch. Therefore, vector steps that do not facilitate rapid call response are avoided.

Glossary

AAR	See Automatic Alternate Routing (AAR).
abandoned call	An incoming call in which the caller hangs up before the call is answered.
Abbreviated Dialing	A feature that allows callers to place calls by dialing just one or two digits.
ACA	See Automatic Circuit Assurance (ACA).
access code	A 1-, 2-, or 3-digit dial code used to activate or cancel a feature, or access an outgoing trunk.
access trunk	A trunk that connects a main communications system with a tandem communications system in an Electronic Tandem Network (ETN). An access trunk can also be used to connect a system or tandem to a serving office or service node. Also called an *access tie trunk*.
ACCUNET	A trademarked name for a family of digital services offered by AT&T in the United States.
ACD	See Automatic Call Distribution (ACD).
ACD agent	See agent.
ACD work mode	See work mode.
ACW	See After Call Work (ACW) mode.
adjunct	A processor that does one or more tasks for another processor and is optional in the configuration of the other processor. See also application.
Adjunct Routing	A means of evaluating calls before the calls are processed by requesting information from an adjunct. The communication server requests instructions from an associated adjunct and makes a routing decision based on agent availability or the caller information.
adjunct-controlled split	An Automatic Call Distribution (ACD) split that is administered to be controlled by another application. Agents logged into such splits must do all telephony work, ACD login/ logout, and changes of work mode through the adjunct (except for auto-available adjunct-controlled splits, whose agents may not log in/out or change work mode).
adjunct-monitored call	An adjunct-controlled call, active-notification call, or call that provides event reporting over a domain-control association.
Adjunct-Switch Application Interface (ASAI)	A recommendation for interfacing adjuncts and communications systems, based on the CCITT Q.932 specification for layer 3.
adjusted EWT	A Best Service Routing (BSR) term for Expected Wait Time (EWT) plus a user adjustment set by a **`consider`** command.

administration terminal	A terminal that is used to administer and maintain a system.
Administration Without Hardware (AWOH)	A feature that allows administration of ports without associated terminals or other hardware.
Advocate	See Avaya Business Advocate.
After Call Work (ACW) mode	A mode in which agents are unavailable to receive ACD calls. Agents enter the ACW mode to perform ACD-related activities such as filling out a form after an ACD call. Also see, auto-in work mode, manual-in work mode, and aux-work mode.
agent	A member of an ACD hunt group, ACD split, or skill. Depending on the ACD software, an agent can be a member of multiple splits/skills.
agent report	A report that provides historical traffic information for internally measured agents.
ANI	See Automatic Number Identification (ANI).
appearance	A software process that is associated with an extension and whose purpose is to supervise a call. An extension can have multiple appearances. Also called call appearance, line appearance, and occurrence. See also call appearance.
application	An adjunct that requests and receives ASAI services or capabilities. One or more applications can reside on a single adjunct. However, the communication server cannot distinguish among several applications residing on the same adjunct and treats the adjunct, and all resident applications, as a single application. The terms application and adjunct are used interchangeably throughout this document.
application plan	A plan used only in multi-site Best Service Routing (BSR) applications. The application plan identifies the remote switches that may be compared in a consider series. The plan also specifies the information used to contact each communication server and to interflow calls to the communication server.
applications processor	A micro-computer based, program controlled computer providing application services for the switch. The processor is used with several user-controlled applications such as traffic analysis and electronic documentation.
ARS	See Automatic Route Selection (ARS).
ASAI	See Adjunct-Switch Application Interface (ASAI).
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. Also see attendant console.

attendant console The workstation used by an attendant. The attendant console allows the attendant to originate a call, answer an incoming call, transfer a call to another extension or trunk, put a call on hold, and remove a call from hold. Attendants using the console can also manage and monitor some system operations. Also called console. Also see attendant.

Audio Information Exchange (AUDIX) An Avaya messaging system. AUDIX has been replaced by Message Manager.

AUDIX See Audio Information Exchange (AUDIX).

auto-in work mode A mode in which an agent is ready to process another call as soon as the current call is completed. Auto-in work mode is one of four agent work modes. Also see, aux-work mode, manual-in work mode, and After Call Work (ACW) mode.

Automatic Alternate Routing (AAR) A feature that routes calls to a different route than the first-choice route when facilities are unavailable.

Automatic Call Distribution (ACD) A feature that answers calls, and then depending on administered instructions, delivers messages appropriate for the caller and routes the call to an agent when one becomes available.

Automatic Call Distribution (ACD) split A method of routing calls of a similar type among agents in a call center. Also, a group of extensions that are staffed by agents trained to handle a certain type of incoming call.

Automatic Callback A feature that enables internal callers, upon reaching a busy extension, to have the system automatically connect and ring both originating and receiving parties when the receiving party becomes available.

Automatic Circuit Assurance (ACA) A feature that tracks calls of unusual duration to facilitate troubleshooting. A high number of very short calls or a low number of very long calls may signify a faulty trunk.

Automatic Number Identification (ANI) A display of the calling number so that agents can access information about the caller.

Automatic Route Selection (ARS) A feature that allows the system to automatically choose the least-expensive way to send a toll call.

automatic trunk A trunk that does not require addressing information because the destination is predetermined. A request for service on the trunk, called a seizure, is sufficient to route the call. The normal destination of an automatic trunk is the communications-system attendant group. Also called automatic incoming trunk and automatic tie trunk.

auxiliary trunk A trunk used to connect auxiliary equipment, such as radio-paging equipment, to a communications system.

aux-work mode A mode in which agents are unavailable to receive Automatic Call Distribution (ACD) calls. Agents enter aux-work mode when involved in non-ACD activities such as taking a break, going to lunch, or placing an outgoing call. Also see, auto-in work mode, manual-in work mode, and After Call Work (ACW) mode.

available agent strategy	A strategy that determines how Best Service Routing (BSR) commands in a vector identify the best split or skill when several have available agents.
Avaya Business Advocate	A product that establishes different levels of service for different types of calls. For example, a company may decide that a premium customer gets faster service than other types of customers.
AWOH	See Administration Without Hardware (AWOH).
barrier code	A security code used with remote access to prevent unauthorized access to the system.
Basic Call Management System (BCMS)	An application on the communication server that monitors the operations of an Automatic Call Distribution (ACD) application. BCMS collects data related to the calls on the communication server and organizes the data into reports that help manage ACD facilities and personnel.
BCC	See Bearer Capability Class (BCC).
BCMS	See Basic Call Management System (BCMS).
Bearer Capability Class (BCC)	A code that identifies the type of a call (for example, voice and different types of data).
best	The split, skill, or location that provides the most advantageous service for a caller as determined by Best Service Routing (BSR).
Best Service Routing (BSR)	An Avaya communication server feature based on call vectoring that routes Automatic Call Distribution (ACD) calls to the split, skill, or contact center best able to service each call. BSR can be used on a single communication server, or it can be used to integrate resources across a network of communication servers.
bridge (bridging)	The appearance of a telephone extension at one or more other telephones.
bridged appearance	A call appearance on a telephone that matches a call appearance on another telephone for the duration of a call.
Business Advocate	See Avaya Business Advocate.
call appearance	1. For the attendant console, the six buttons labeled a-f used to originate, receive, and hold calls. Two lights next to the button show the status of the call appearance. 2. For the telephone, a button labeled with an extension and used to place outgoing calls, receive incoming calls, or hold calls. Two lights next to the button show the status of the call appearance.
Call Detail Recording (CDR)	A feature that uses software and hardware to record call data.
Call Management System (CMS)	An application that enables customers to monitor and manage telemarketing centers by generating reports on the status of agents, splits, trunks, trunk groups, vectors, and VDNs. CMS enables customers to partially administer the Automatic Call Distribution (ACD) feature for a communications system.
call vector	A set of vector commands used to process an incoming or internal call.

call work code	A number entered by ACD agents to record the occurrence of customer-defined events (such as account codes, social security numbers, or phone numbers) on ACD calls.
callback call	A call that automatically returns to a voice-terminal user who activated the Automatic Callback feature.
cause value	A value that is returned in response to requests or in event reports when a denial or unexpected condition occurs.
CCS or hundred call seconds	A unit of call traffic. Call traffic for a facility is scanned every 100 seconds. If the facility is busy, it is assumed to have been busy for the entire scan interval. There are 3600 seconds per hour. The Roman numeral for 100 is the capital letter C. The abbreviation for call seconds is CS. Therefore, 100 call seconds is abbreviated CCS. If a facility is busy for an entire hour, it is said to have been busy for 36 CCS.
CCR	CCR is the internal development name for Avaya's next generation reporting platform.
CDR	See Call Detail Recording (CDR).
Central Office (CO)	A switch owned by a local telephone company that provides local telephone service (dial-tone) and access to toll facilities for long-distance calling.
Central Office (CO) trunk	A telecommunications channel that provides access from the system to the public network through the local CO.
channel	1. A circuit-switched call. 2. A communications path for transmitting voice and data. 3. In wideband, all of the time slots (contiguous or noncontiguous) necessary to support a call. Example: an H0-channel uses six 64-kbps time slots. 4. A DS0 on a T1 or E1 facility not specifically associated with a logical circuit-switched call; analogous to a single trunk.
circuit	1. An arrangement of electrical elements through which electric current flows. 2. A channel or transmission path between two or more points.
circuit pack	A card with microprocessors, transistors, and other electrical circuits. A circuit pack is installed in a switch carrier or bay. Also called a circuit board or circuit card.
Class of Restriction (COR)	A feature that allows classes of call-origination and call-termination restrictions for telephones, telephone groups, data modules, and trunk groups. See also Class of Service (COS).
Class of Service (COS)	A feature that uses a number to specify if telephone users can activate the Automatic Callback, Call Forwarding All Calls, Data Privacy, or Priority Calling features. See also Class of Restriction (COR).
CO	See Central Office (CO).

communications server A software-controlled processor complex that interprets dialing pulses, tones, and keyboard characters and makes the proper connections both within the system and external to the system. The communications system itself consists of a digital computer, software, storage device, and carriers with special hardware to perform the connections. A communications system provides voice and data communications services, including access to public and private networks, for telephones and data terminals on a customer's premises. Previously called a switch or a Private Branch eXchange (PBX).

confirmation tone A telephone tone confirming that feature activation, deactivation, or cancellation has been accepted.

connectivity A connection of disparate devices within a single system.

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 a set of two or more. A set of **consider** commands is called a consider series.

console See attendant console.

COR See Class of Restriction (COR).

COS See Class of Service (COS).

coverage answer group A group of up to eight telephones that ring simultaneously when a call is redirected to it by Call Coverage. Any one of the group can answer the call.

coverage call A call that is automatically redirected from the called party's extension to an alternate answering position when certain coverage criteria are met.

coverage path An order in which calls are redirected to alternate answering positions.

coverage point An extension or attendant group, VDN, or ACD split designated as an alternate answering position in a coverage path.

covering user A person at a coverage point who answers a redirected call.

CWC See call work code.

data link A configuration of physical facilities enabling end terminals to communicate directly with each other.

data terminal An input/output (I/O) device that has either switched or direct access to a host computer or to a processor interface.

dial-repeating tie trunk A tie trunk that transmits called-party addressing information between two communications systems.

dial-repeating trunks A PBX tie trunk that is capable of handling PBX station-signaling information without attendant assistance.

direct agent A feature, accessed only through ASAI, that allows a call to be placed in a split queue but routed only to a specific agent in that split. The call receives normal ACD call treatment (for example, announcements) and is measured as an ACD call while ensuring that a particular agent answers.

Direct Inward Dialing (DID) trunk	An incoming trunk used for dialing directly from the public network into a communications system without help from the attendant.
domain	A group of VDNs, ACD splits, and stations.
Dynamic Percentage Adjustment	An Avaya Business Advocate feature that makes automatic adjustments to agents' target allocations as needed to help meet the administered service level targets.
Dynamic Queue Position	An Avaya Business Advocate feature that gives you the ability to queue calls from multiple VDNs to a single skill, while maintaining different service objectives for those VDNs.
Dynamic Threshold Adjustment	An Avaya Business Advocate Service Level Supervisor feature that automatically adjusts overload thresholds to engage reserve agents a bit sooner or a bit later to meet the administered service levels.
EAD-LOA	See Expert Agent Distribution-Least Occupied Agent (EAD-LOA).
EAD-MIA	See Expert Agent Distribution-Most Idle Agent (EAD-MIA).
Electronic Tandem Network (ETN)	A large private network that has automatic call-routing capabilities based on the number dialed and the most preferred route available. Each switch in the network is assigned a unique private network office code (RNX), and each telephone is assigned a unique extension.
EPN	See Expansion Port Network (EPN).
ETN	See Electronic Tandem Network (ETN),
EWT	See Expected Wait Time (EWT).
Exclusion	A feature that allows multi-appearance telephone users to keep other users with the same extension from bridging onto an existing call.
Expansion Port Network (EPN)	A port network that is connected to the Time Division Multiplex (TDM) bus and packet bus of a processor port network. Control is achieved by indirect connection of the EPN to the processor port network using a port-network link.
Expected Wait Time (EWT)	A prediction of how long a call waits in queue before the call is answered.
Expert Agent Distribution-Least Occupied Agent (EAD-LOA)	An agent selection method for delivery of calls. With EAD-LOA implemented, calls are delivered to the available agent with the highest skill level and the lowest percentage of work time since login (compared to other available agents with the same skill level). See also Expert Agent Distribution-Most Idle Agent (EAD-MIA), Uniform Call Distribution-Least Occupied Agent (UCD-LOA), and Uniform Call Distribution-Most Idle Agent (UCD-MIA).

Expert Agent Distribution-Most Idle Agent (EAD-MIA)	An agent selection method for delivery of calls. With EAD-MIA implemented, calls are delivered to the available agent with the highest skill level who has been idle the longest since their last ACD call (compared to other available agents with the same skill level). See also Expert Agent Distribution-Least Occupied Agent (EAD-LOA), Uniform Call Distribution-Least Occupied Agent (UCD-LOA), and Uniform Call Distribution-Most Idle Agent (UCD-MIA).
extension-in (EXT-IN)	A work state agents go into when they answer a non ACD call. If the agent is in Manual-In or Auto-In and receives an EXT-IN call, the call is recorded by the reporting adjunct as an AUX-IN call.
extension-out (EXT-OUT)	A work state that agents go into when they place a non-ACD call.
external call	A connection between a communications system user and a party on the public network, or on another communications system in a private network.
facility	A telecommunications transmission pathway and the associated equipment.
Forced Agent Logout from ACW mode	A feature used to automatically log out an Expert Agent Selection (EAS) agent who spends too much time in After Call Work (ACW) mode.
Forced Agent Logout by Clock Time	A feature used to automatically log out an Expert Agent Selection (EAS) agent at a pre-determined time. This feature is primarily used to automatically log off agents at the end of their shifts.
glare	A simultaneous seizure of a 2-way trunk by two communications systems resulting in a standoff.
ground-start trunk	A trunk on which, for outgoing calls, the system transmits a request for services to a distant switching system by grounding the trunk ring lead. To receive the digits of the called number, that system grounds the trunk tip lead. When the system detects this ground, the digits are sent.
holding time	A total length of time in minutes and seconds that a facility is used during a call.
intelligent polling	An automatic feature of Best Service Routing (BSR) that significantly reduces the number of status polls executed. When a remote location cannot be the best resource at a given moment in time, the intelligent polling feature temporarily suppresses polls to that location. Also see status poll.
intercept tone	An tone that indicates a dialing error or denial of the service requested.
interflow	An Automatic Call Distribution (ACD) term that refers to the ability to establish a connection to a second ACD and overflow a call from one ACD to the other.
internal call	A connection between two users within a system.
internal measurement	A Basic Call Management System (BCMS) measurement that is made by the system.
intraflow	An Automatic Call Distribution (ACD) term that refers to the ability for calls to redirect to other splits on the same communication server to backup the primary split.

in-use lamp	A red light on a multiappearance telephone that lights to show which call appearance will be selected when the handset is lifted or which call appearance is active when a user is off-hook.
ISDN Gateway (IG)	A feature allowing integration of the switch and a host-based telemarketing application using a link to a gateway adjunct. The gateway adjunct is a 3B-based product that notifies the host-based telemarketing application of call events.
ISDN trunk	A trunk administered for use with ISDN-PRI. Also called ISDN facility.
line	A transmission path between a communications system or Central Office (CO) switching system and a telephone.
line appearance	See appearance.
line port	A piece of hardware that provides the access point to a communications system for each circuit associated with a telephone or data terminal.
link	A transmitter-receiver channel that connects two systems.
Location Preference Distribution	A feature used to route incoming Automatic Call Distribution (ACD) calls to agents located at the same location where the trunk is located whenever possible.
maintenance	Activities involved in keeping a telecommunications system in proper working condition: the detection and isolation of software and hardware faults, and automatic and manual recovery from these faults.
major alarm	An indication of a failure that has caused critical degradation of service and requires immediate attention. Major alarms are automatically displayed on LEDs on the attendant console and maintenance or alarming circuit pack, logged to the alarm log, and reported to a remote maintenance facility, if applicable.
management terminal	The terminal that is used by the system administrator to administer the switch. The terminal may also be used to access the Basic Call Management System (BCMS) feature.
manual-in work mode	A mode in which an agent is ready to process another call manually. Also see, auto-in work mode, aux-work mode, and After Call Work (ACW) mode.
Maximum Agent Occupancy (MAO)	A feature used to set thresholds on the amount of time an agent spends on a call. MAO is used to prevent agent burnout. The MAO threshold is a system-administered value that places an agent in AUX mode when the agent exceeds the MAO threshold for calls.
message center	An answering service that supplies agents and stores messages for later retrieval.
message-center agent	A member of a message-center hunt group who takes and retrieves messages for telephone users.
messaging system	A generic name for a system that records, stores, plays, and distributes phone messages. Message Manager is the latest messaging system provided by Avaya.

minor alarm	An indication of a failure that could affect customer service. Minor alarms are automatically displayed on LEDs on the attendant console and maintenance or alarming circuit pack, sent to the alarm log, and reported to a remote maintenance facility, if applicable.
modular processor data module (MPDM)	A Processor Data Module (PDM) that can be configured to provide several kinds of interfaces (RS-232C, RS-449, and V.35) to customer-provided data terminal equipment (DTE).
Modular Trunk Data Module (MTDM)	A trunk-data module that can be configured to provide several kinds of interfaces (RS-232, RS-449, and V.35) to customer-provided data terminal equipment.
multiappearance telephone	A telephone equipped with several call-appearance buttons for the same extension, allowing the user to handle more than one call on that same extension at the same time.
Network Specific Facility (NSF)	An information element in an ISDN-PRI message that specifies which public-network service is used. NSF applies only when Call-by-Call Service Selection is used to access a public-network service.
NFAS	See Non-Facility Associated Signaling (NFAS).
Non-Facility Associated Signaling (NFAS)	A method that allows multiple T1 or E1 facilities to share a single D-channel to form an ISDN-PRI. If D-channel backup is not used, one facility is configured with a D-channel, and the other facilities that share the D-channel are configured without D-channels. If D-channel backup is used, two facilities are configured to have D-channels (one D-channel on each facility), and the other facilities that share the D-channels are configured without D-channels.
non switch-classified outbound calls	Proactive Contact outbound calls that are automatically launched by Communication Manager. See also, switch-classified outbound calls.
NSF	See Network Specific Facility (NSF).
occurrence	See appearance.
pickup group	A group of individuals authorized to answer any call directed to an extension within the group.
PMS	See Property Management System (PMS).
poll	See status poll.
poll suppression	An automatic feature of Best Service Routing (BSR) that significantly reduces the number of status polls executed. When a remote location cannot be the best resource at a given moment in time, the intelligent polling feature temporarily suppresses polls to that location. Also see status poll.
polling, intelligent	See intelligent polling.
PPN	See Processor Port Network (PPN).
primary extension	A main extension associated with the physical telephone or data terminal.

principal	A terminal that has its primary extension bridged on one or more other terminals.
principal (user)	A person to whom a telephone is assigned and who has message-center coverage.
private network	A network used exclusively for the telecommunications needs of a particular customer.
Processor Port Network (PPN)	A port network (PN) controlled by a switch-processing element that is directly connected to that PN's TDM bus and LAN bus.
Property Management System (PMS)	A stand-alone computer used by lodging and health-services organizations for services such as reservations, housekeeping, and billing.
public network	A network that can be openly accessed by all customers for local and long-distance calling.
queue	An ordered sequence of calls waiting to be processed.
queuing	A process of holding calls in order of their arrival to await connection to an attendant, to an answering group, or to an idle trunk. Calls are automatically connected in first-in, first-out sequence.
R2-MFC signaling	A signal consisting of two frequency components, such that when a signal is transmitted from a switch, another signal acknowledging the transmitted signal is received by the switch.
recall dial tone	A tone signalling that the system has completed a function (such as holding a call) and is ready to accept dialing.
redirection criteria	Information administered for each telephone's coverage path that determines when an incoming call is redirected to coverage.
Redirection on No Answer	An optional feature that redirects an unanswered ringing ACD call after an administered number of rings. The call is then redirected back to the agent.
reorder tone	A tone to signal that at least one of the facilities, such as a trunk or a digit transmitter, needed for the call was not available.
Service Level Maximizer (SLM)	An agent selection strategy that ensures that a defined service level of *X*% of calls are answered in *Y* seconds. When SLM is active, the software verifies that inbound calls are matched with agents in a way that makes sure that the administered service level is met. SLM is an optional Call Vectoring feature that is used with Expert Agent Selection (EAS), and without Business Advocate.
simulated bridged appearance	A feature that allows the terminal user (usually the principal) to bridge onto a call that had been answered by another party on his or her behalf. Also called a *temporary bridged appearance*.
SLM	See Service Level Maximizer (SLM).
split (agent) status report	A report that provides real-time status and measurement data for internally-measured agents and the split to which they are assigned.

split condition	A condition whereby a caller is temporarily separated from a connection with an attendant. A split condition automatically occurs when the attendant, active on a call, presses the start button.
split number	An identification of the split to the communication server and the Basic Call Management System (BCMS).
split report	A report that provides historical traffic information for internally measured splits.
staffed	An indication that an agent position is logged in. A staffed agent functions in one of four work modes: auto-in work mode, manual-in work mode, After Call Work (ACW) mode, or aux-work mode.
Station Message Detail Recording (SMDR)	An obsolete term now called Call Detail Recording (CDR).
status lamp	A green light that shows the status of a call appearance or a feature button by the state of the light (lit, flashing, fluttering, broken flutter, or unlit).
status poll	A call placed by a consider location vector command to obtain status data from a remote location in a multi-site Best Service Routing (BSR) application.
stroke counts	A method used by ACD agents to record up to nine customer-defined events per call when a reporting adjunct is active.
switch-classified outbound calls	Outbound calls placed by the Proactive Contact dialer and connected to the agents. See also, non switch-classified outbound calls.
system printer	An optional printer that may be used to print scheduled reports using the report scheduler.
system report	A report that provides historical traffic information for internally-measured splits.
system-status report	A report that provides real-time status information for internally-measured splits.
trunk	A dedicated telecommunications channel between two communications systems or Central Offices (COs).
trunk allocation	The manner in which trunks are selected to form wideband channels.
trunk group	Telecommunications channels assigned as a group for certain functions that can be used interchangeably between two communications systems or Central Offices (COs).
UDP	See Uniform Dial Plan (UDP).
Uniform Call Distribution-Least Occupied Agent (UCD-LOA)	An agent selection method for delivery of calls. With UCD-LOA implemented, calls are delivered to the available agent with the lowest percentage of work time since login. Also see Expert Agent Distribution-Least Occupied Agent (EAD-LOA), Expert Agent Distribution-Most Idle Agent (EAD-MIA), and Uniform Call Distribution-Most Idle Agent (UCD-MIA).

Uniform Call Distribution-Most Idle Agent (UCD-MIA)	An agent selection method for delivery of calls. With UCD-MIA implemented, calls are delivered to the available agent who has been idle the longest since their last ACD call. See also EAD-LOA, EAD-MIA, and UCD-LOA.
Uniform Dial Plan (UDP)	A feature that allows a unique number assignment for each terminal in a multiswitch configuration such as a Distributed Communications System (DCS) or main-satellite-tributary system.
VDN	See Vector Directory Number (VDN).
Vector Directory Number (VDN)	An extension that provides access to the vectoring feature on the switch. Vectoring allows a customer to specify the treatment of incoming calls based on the dialed number.
vector-controlled split	A hunt group or ACD split administered with the vector field enabled. Access to such a split is possible only by dialing a VDN extension.
work mode	A mode that an ACD agent can be in. Upon logging in, an agent enters aux-work mode. To become available to receive ACD calls, the agent enters auto-in work mode or manual-in work mode. To do work associated with a completed ACD call, an agent enters After Call Work (ACW) mode.
work state	An ACD agent may be a member of up to three different splits. Each ACD agent continuously exhibits a work state for every split of which it is a member. Valid work states are Avail, Unstaffed, AUX-Work, ACW, ACD (answering an ACD call), ExtIn, ExtOut, and OtherSpl. An agent's work state for a particular split may change for a variety of reasons. For example, an agent's work state changes when a call is answered or abandoned, or the agent changes work modes. The Basic Call Management System (BCMS) feature monitors work states and uses this information to provide BCMS reports.

Index

Symbols

character . . . 702, 775, 780, 781
character for comments . . . 527
comparisons . . . 593
@step parameter . . . 166

A

Abbreviated Dialing
  lists . . . 436
  special characters, route-to . . . 623
account number collection example . . . 795
ACD Agent Login ID screen . . . 490
ACW mode . . . 853
adapting
  to a long wait . . . 54
  to changing call traffic . . . 54
Adjunct route using NCR failed . . . 714
adjunct routing
  considerations . . . 202
  function . . . 201
  hardware and software requirements . . . 671, 672, 673
adjunct routing command . . . 47, 530
  neutral vector command . . . 535
  success/failure criteria . . . 690
  syntax . . . 530
  troubleshooting . . . 696
adjust-by . . . 309
administering duplicate vectors . . . 236
Administration Without Hardware
  and phantom calls . . . 213
Administration Without Hardware (AWOH)
  EAS interactions . . . 493
advanced vector routing
  hardware and software requirements . . . 669
  overview . . . 257
Advocate, see Avaya Business Advocate
after call work (ACW)
  buttons . . . 471
agent login ID associated capabilities . . . 491
Agent Login ID screen . . . 490
agent selection
  adjust-by . . . 309
Agent Status Info Invalid . . . 717
agents
  available . . . 34
    definition . . . 34
  direct . . . 455
  logical . . . 455, 456
  optimal utilization . . . 273
  when available . . . 227
  when not available . . . 227
algorithms . . . 780
Alternate Selection on BSR Ties . . . 303
ANI
  call types used in . . . 170
  in vector routing tables . . . 170
  internal transfer to VDN . . . 170
  string length . . . 170
  use in EAS agent calls . . . 170
  using in vector routing tables . . . 172
  vector example . . . 172
  wildcards used with . . . 170
ani type variable . . . 122
ANI/ii-digits
  hardware and software requirements . . . 670
ANI/ii-digits routing
  requirements . . . 219
announcement . . . 424
announcement command . . . 47, 424
  neutral vector command . . . 276, 543
  success/failure criteria . . . 690
  syntax . . . 538
  troubleshooting . . . 697
  variables and . . . 112, 152
announcements . . . 538
  recording . . . 542
answer supervision considerations
  adjunct routing . . . 534
  announcement . . . 543
  busy . . . 546
  collect digits . . . 561
  converse-on . . . 572
  disconnect . . . 565, 580
  goto step . . . 565
  messaging . . . 603
  queue-to . . . 611
  route-to . . . 615, 626
  stop . . . 643
  wait-time . . . 651
application
  example
    adjunct routing . . . 67, 76
    ANI routing . . . 69
    automated attendant . . . 60
    basic call vectoring . . . 59, 61, 65, 67, 69, 72, 76

call prompting 60, 61, 67, 69, 76
customer service center 59
data in/voice answer 61
data/message collection 61
distributed call centers 65
DIVA and data/message collection 61
expected wait-time 69
expert agent selection 72, 76
help desk 67
insurance agency/service agency 69
look-ahead interflow 65
resort reservation service 76
rolling ASA 69
VDN calls 69
warranty service 72
warranty service call center 74
arithmetic operations
examples 783
invalid results 775
rules 773
start and length 775
ASA
definition 762
rolling versus interval 267
split calculation 268
VDN calculation 268
when to use 267
ASAI
link failure 532
ASAI message
contents of 205
asaiuui type variable 122
used with the set command 634
Assignment not allowed 722
asterisk (*) 557
*, use of 557
AT&T In-Band Transfer and Connect. See Network Call Redirection
Attendant 423, 441
Attendant Call Waiting
call waiting tones 472
Attendant Vectoring
announcement Command 425, 443
busy Command 425, 443
Command Set 357, 424
disconnect Command 425, 443
goto step Command 358, 369, 429, 444
goto vector Command 358, 369, 430
Hunt Group Queue 434
Night Service 435
queue-to attd-group Command 426
queue-to attd-group command 426
queue-to attendant Command 427
queue-to attendant command 427
queue-to hunt-group 428
queue-to hunt-group Command 428
Redirecting Calls to Attendant VDNs 434
Restrictions 434
route-to number Command 428, 444
route-to number command 428
stop Command 430, 445
VDNs 435
wait-time Command 425, 443
wait-time command 425, 443
Automatic Call Distribution (ACD)
call handling preferences 471
automating tasks 56
Auxiliary data 436
Avaya Business Advocate 458, 502, 503, 513, 569
average speed of answer
definition 762

B

Bad resp from status poll 718
Basic Call Vectoring
command set 103
basic call vectoring
considerations 105
hardware and software requirements 668
basic components of call vectoring 52
BCMS 547
description of 751
function 751
interactions with
adjunct routing 537
busy 547
check-backup 555
converse-on 577
disconnect 582
messaging 605
queue-to 613
route-to 631
reports 762
BCMS Split Report 762
for security use 820
VDN Real-Time Report 763
VDN Summary Report 762
standards 754
for interpreting split flows 755
for interpreting VDN flows 754
benefits of call vectoring 54
Best Service Routing (BSR)
benefits 298
call vectoring
agent surplus situations 308
call surplus situations 307
commands for single-site BSR 305
commands
consider 561
goto step 583

queue-to . . . 606
reply-best . . . 614
determining the best resource . . . 307
hardware and software requirements . . . 669
local treatment feature . . . 346
multi-site
administration procedures . . . 343-345
Application Plan screen . . . 324
application plans . . . 324
applications . . . 323
examples
with 2 switches . . . 326
with 4 switches . . . 333
4 switches, limited trunks . . . 333
planning . . . 342-343
screens required . . . 320
requirements
for networks . . . 301
for switches . . . 301
single-site . . . 304
administration procedures . . . 318-319
examples
basic . . . 310
user adjustments . . . 314
planning . . . 317
screens required . . . 305
vectors
tips for writing . . . 354
bilingual announcements example . . . 794
Block
send reply-best . . . 719
blocking new incoming calls . . . 849
branching . . . 48
branching and programming . . . 47
BSR EWT is infinite . . . 718
BSR Poll
TSC not administered . . . 719
BSR poll glare retry . . . 718
BSR poll no trunks . . . 718
BSR poll seize fail . . . 718
BSR Status Info Invalid . . . 717
BSR status poll attempt failed . . . 718
BSR Ties, Alternate Selection . . . 303
Business Advocate, see Avaya Business Advocate
busy . . . 47, 424, 545
busy command
success/failure criteria . . . 690
syntax . . . 545
troubleshooting . . . 698

C

call center setup
EAS
agent skills worksheet . . . 839
current split operation worksheet . . . 836
customer needs worksheet . . . 837
individual agent skill worksheet . . . 838
objectives worksheet . . . 834
VDN skill preferences worksheet . . . 840
key factors . . . 825
non-EAS
guidelines . . . 827
steps . . . 826
call flow method . . . 32
adjunct routing . . . 33
interflow . . . 32
intraflow . . . 32
look-ahead interflow . . . 32
multiple split queuing . . . 32
call flows
answered and abandoned calls . . . 753
busies and disconnects . . . 753
classes of . . . 752
converse-VRI calls . . . 807
defining and interpreting . . . 752
split inflows, outflows, and dequeues . . . 754
types that are tracked . . . 752
VDN inflows and outflows . . . 753
vector inflows and outflows . . . 754
call group setup
guidelines . . . 827
key factors . . . 825
call handling
optimal . . . 273
call handling preferences . . . 471
Call is not incoming ISDN . . . 716
call not queued at stop step . . . 709
call prompting
call set . . . 238
command categories . . . 238
considerations . . . 254
digit entry . . . 239
entering variable length digit strings . . . 240
functions . . . 241
creating service observing vectors . . . 247
passing digits to an adjunct . . . 247
using digits on the agent's set . . . 245
using digits to collect branching information . . . 243
using digits to select options . . . 245
hardware and software requirements . . . 668
purpose . . . 237
removing incorrect digits . . . 239
variable length digit string . . . 239
with VRI . . . 237
call treatment
customizing . . . 56
personalization . . . 56
Call Vector screen . . . 431
call vectoring

benefits . . . 54
definition . . . 51
removing incorrect digits . . . 240, 251
upgrading to . . . 673
call vectoring command
neutral vector command . . . 604
call-back provisions
diagram of . . . 79
Caller Information Forwarding
buffer storage considerations . . . 182
interactions . . . 183
internal transfer to VDN . . . 182
string length . . . 182
UEC IE storage . . . 181
used with collect digits command . . . 181
vector example . . . 183
wildcards . . . 181
Caller Information Forwarding (CINFO)
answer supervision . . . 561
with collect digits command . . . 556, 560
caller needs
example table matching skills and needs . . . 468
calling
a direct agent . . . 34
during non-business hours . . . 233
calling during non-business hours . . . 233
CALLR-INFO button
format of display . . . 245
CALLR-INFO button format of display . . . 245
Can't set, no lcl var . . . 722
cdpd digits left behind . . . 711
cdpd digits not available . . . 711
ced digits left behind . . . 711
ced digits not available . . . 711
change vector screen . . . 222
changing vectors . . . 218, 674
check-backup . . . 47
check-backup command . . . 231, 548
example . . . 609
neutral vector command . . . 276, 553
syntax . . . 549
troubleshooting . . . 698
checking
queue capacity . . . 232
Chg Station no Cons/Perm . . . 725
CMS
description of . . . 751
function . . . 751
interactions
with adjunct routing . . . 535
with busy . . . 547
with check digits . . . 561
with goto vector . . . 565
reports
for security use . . . 820
Split Summary Report . . . 762
VDN Report . . . 762
Vector Report . . . 762
standards . . . 754
for interpreting split flows . . . 755
for interpreting VDN flows . . . 754
using in expert agent selection environment . . . 763
collect command
variables and . . . 113
collect digits . . . 556
collect digits command . . . 47, 238
syntax . . . 528, 556, 632
troubleshooting . . . 698, 699
collect digits for variable error . . . 712
collect type variable . . . 131
Collected dgts got bumped . . . 719
collecting and acting on information . . . 47
command category
for call prompting . . . 238
command table
for call prompting . . . 238
comments within vectors . . . 527
comparison operators . . . 50
comparisons using none, # and numeric digits . . . 593
conditionals for gateways and servers . . . 595
Conf/Transfer 2 Meet-me . . . 725
Conference COR restrict . . . 724
connecting to voice mail . . . 55
consider command . . . 561
multi-site examples . . . 333, 339
single-site examples . . . 310, 314
consider split/location adjust-by x . . . 309
considerations
adjunct routing . . . 202
basic call vectoring . . . 105
call prompting . . . 254
look-ahead interflow . . . 294
VDN return destination . . . 683
control flow
type
conditional branching . . . 45
sequential flow . . . 45
unconditional branching . . . 45
converse VRI calls
call flow phase
data passing . . . 808
data return . . . 814
script completion . . . 816
script execution . . . 813
converse-on command . . . 48, 566
function . . . 567
neutral vector command . . . 276
success/failure criteria . . . 692
syntax . . . 562, 566
troubleshooting . . . 700

variables and . . . 113
converse-VRI calls
call flow phase
VRU data collection . . . 813
counted-calls
conditional test when routing calls . . . 271
CPN prefixes . . . 185
about . . . 185
administering . . . 186
example . . . 186
prerequisites . . . 187
create a new Holiday table . . . 360
creating
a new vector . . . 218
service observing vectors . . . 242
creating a new vector . . . 218
credit card numbers . . . 791
customizing call treatment . . . 34, 56

D

Data dropped by other app . . . 719
defining desired service . . . 38
deleting
vector step . . . 224
deleting vector step . . . 224
delivery of queued calls . . . 227
dequeued average queue time
definition . . . 762
determining the number of digits example . . . 641
dial-ahead digits
ASAI provided . . . 254
dial-ahead digits and the digits buffer . . . 640
digits . . . 239
ASAI provided dial-ahead digits . . . 254
collect digits
maximum number . . . 556
collect digits command
maximum number . . . 531
collected prior to timeout . . . 557
entering . . . 239
dial-ahead digits . . . 239, 241
variable-length digit strings . . . 240
including # sign . . . 559
maximum number . . . 559
removing
incorrect digit strings . . . 239, 240
returned by VRU . . . 557
Touch-Tone . . . 557
digits buffer clearing example . . . 641
direct agent . . . 455
direct agent call
definition . . . 34
direct agent calling (DAC)
call handling preferences . . . 471
directing calls to a specific agent . . . 456
disconnect . . . 424
disconnect command . . . 48, 579
success/failure criteria . . . 693
syntax . . . 562, 579
troubleshooting . . . 700, 701
variables and . . . 114, 154
displaying digits on the agent's set . . . 241
Divide by Zero . . . 722
DNIS information displayed to answering agent . . . 291
dow type variable . . . 124
doy type variable . . . 125
Duplicate Vector command . . . 234
duplicate vectors . . . 236
during peak
heavy traffic . . . 230

E

EAS
definition . . . 455
Emergency access redirection . . . 436
enabling the vector disconnect timer . . . 673
encouraging caller to remain on-line . . . 229
entering
a command
in abbreviated screen . . . 220
a vector . . . 217
dial-ahead digits . . . 241
digits . . . 239
use of # . . . 240
variable-length digit strings . . . 239, 240
vector steps . . . 218
entering vector comments . . . 225
Ericsson AXE-10 configuration information . . . 387
evaluating
effectiveness of vector programming . . . 752
performance . . . 752
split performance . . . 760
event type
adjunct route failed . . . 713
events . . . 707, 709
EWT
factors that affect prediction accuracy . . . 265
factors that decrease estimate for split priority level 265
factors that increase estimate for split priority level 265
for individual calls . . . 261
sending wait time without using VRU . . . 263
testing for split . . . 260
Troubleshooting . . . 266
using to route to best split . . . 264
zero and infinite values . . . 260
example application
remote access with host provided security . . . 685
saving in trunk facilities between call centers . . . 686

split flow tracking . . . 755
VDN override . . . 41
warranty service call center . . . 74
example vector
accommodate a super agent pool . . . 486
automated attendant application . . . 60
call interflow . . . 625
claims application . . . 70
customer service application . . . 71
delay with multiple audio/music source feedback . 647
dial-ahead digits . . . 252
distributed call centers application . . . 66
DIVA and data/message collection application . 62, 63
emergency and routine service application . . . 96, 98
help desk application . . . 68
late caller application . . . 100
leaving recorded messages . . . 602, 611
messaging options application . . . 102
multiple split queuing . . . 609
passing digits to an adjunct . . . 247
receiving switch inflow vector . . . 280
remote access service observing vector . . . 248
return destination vector
with announcement . . . 687
with remote access . . . 685
service agency clients application . . . 71
service observing vector . . . 248, 249
stopping vector processing . . . 643
tandem switch vector . . . 288
testing
for digit . . . 244
for digits in vector routing table . . . 244
for digits not in vector routing table . . . 244
treating digits as a destination . . . 242
unconditional branching . . . 591
using digits to collect branching information . . . 243
using digits to select options . . . 245
vector for service observing . . . 626
example vector step
announcement . . . 539
converse-on . . . 567
executing VRU scripts . . . 47
Expected Wait Time
factors that affect prediction accuracy . . . 265
factors that decrease estimate for split priority level 265
factors that increase estimate for split priority level 265
for individual calls . . . 261
sending wait time without using VRU . . . 263
testing for split . . . 260
troubleshooting . . . 266
using to route to best split . . . 264
zero and infinite values . . . 260
expert agent selection
adjunct interactions . . . 498
conversion
administration for . . . 844
blocking of new incoming calls . . . 849
considerations prior to . . . 843
steps . . . 843
steps for cutover . . . 849
definition . . . 455
feature interactions . . . 493
requirements . . . 456
requires ACD . . . 456
requires call vectoring . . . 456
splits . . . 456
tracking
agents and their skills . . . 763
direct agent calls . . . 764
for VDN skill preferences . . . 764
non-ACD calls . . . 764
upgrading to . . . 843
upgrading to R5 . . . 502
using CMS . . . 763
Expert Agent Selection (EAS)
direct agent announcement (DAA)
capabilities . . . 473
forms . . . 473

F

feature interactions
with adjunct routing . . . 534
with announcement . . . 543
with busy . . . 546
with check digits . . . 561, 642
with check-backup . . . 553
with converse-on . . . 573
with disconnect . . . 581
with goto step . . . 565, 598
with messaging . . . 603
with queue-to . . . 612
with route-to . . . 626
with stop . . . 643
with wait-time . . . 651
Forced Agent Logout from ACW mode . . . 853
functions
of call prompting . . . 241
functions of call prompting . . . 241

G

gateway and server conditionals . . . 595
global variable change . . . 726
Glossary . . . 863
goto command
example . . . 591
neutral vector command . . . 276
success/failure criteria . . . 693
troubleshooting . . . 701

goto step 424
goto step command 48, 357, 424, 583
neutral vector command 598
goto vector 424
goto vector command 48, 424

H

handling multiple calls 35
holiday
table 360
vectoring 357, 367
vectors 362
Holiday table
Create a new 360
Hunt Group night destination 435
Hunt Group Queue 434

I

identifying caller needs
call prompting/VRU digits 469
direct agent calling 470
DNIS/ISDN called party 469
example prompts 469
host database lookup 470
methods of 467
table of services and DNIS digits 469
II-digits
assigned codes 175
call types that include 174
internal transfer to VDN 174
preserved with VDN Return Destination feature . . 174
string description 174
use in vector routing tables 174
wildcards 174
Illegal TSC interaction 724
improving
performance 855
the average speed of answer 55
Information Forwarding
ASAI UUI IE shared format conversion 195
bylet lengths for UUI user data 198
general feature description 189
information forwarding
backward compatibility, LAI 195
benefits 190
collected digits with interflowed call 192
data handled by 189
determining byte lengths for UUI user data 196
global transport support 193, 194
in-VDN time in interflowed call 193
network requirements 191, 192
support for BSR and LAI 192
troubleshooting 199
inserting vector steps 224
Interflow 624
Interflow VDN field is blank 717
invalid character 702, 775, 780, 781
Invalid status polling destination 718
Invalid table number 712
In-VDN time got bumped 719
Invld Num Digits MM Acc 724

L

LAI
function 273
Last coverage point in a coverage path 436
latest VDN 39
LDN and trunk night destination 436
leaving a message 234, 602
listing existing vectors 218
local treatment
BSR 346
Location not on BSR screen 717
Location Preference Distribution 853
logical agent 455
look-ahead interflow
achieving FIFO 281
ADR 292
alternate destination redirection 292
considerations 294
diagram of tandem switch configuration 287
DNIS and VDN override 290
DNIS information 291
enhanced 281
function 273
hardware and software requirements 671
interflow eligibility 283
multisite applications 293
route-to command 277
setting the minimum EWT 284
tandem switch configuration
far end switch operation 288
sending switch operation 287
tandem switch operation 288
troubleshooting 295, 695
two switch configuration
receiving switch operation 279
Look-Ahead Interflow attempt failed 717
LUHN-10 algorithm 780

M

MAO
general description of 515
reporting and 518
maximizing performance 855, 857, 859, 861
example vector 856, 857, 858, 861, 862

Maximum Agent Occupancy, see MAO
media-gateway vector conditional . . . 595
Meet-Me Access chg TMO . . . 724
Meet-me Conf call full . . . 725
Meet-me Conference
  Command Set . . . 441
  troubleshooting . . . 453
Merge Meet-me Conf call . . . 725
messaging . . . 48, 600
  ASAI
    contents of . . . 205
  example . . . 611
  leaving a message . . . 234
messaging command
  example . . . 602
  neutral vector command . . . 277
  success/failure criteria . . . 693
  syntax . . . 600
  troubleshooting . . . 701
messaging step in a vector . . . 602
MM Abbrev Dial Invalid . . . 725
MM Access Chg Not a VDN . . . 725
MM Access Obj/SAT Busy . . . 725
MM Extension not valid . . . 725
MM Inv Trk not Remote Acc . . . 725
MM Invalid Access Entered . . . 725
MM Invalid Conf Ctrlr Sta . . . 725
MM Invalid Station Type . . . 725
MOD10 algorithm . . . 780
MOD10 operations
  examples . . . 789
  invalid results . . . 782
  rules . . . 779
  start and length . . . 782
multi-national calling party number prefixes . . . 185
  about . . . 185
  administerting . . . 186
  example . . . 186
  prerequisites . . . 187
multi-national prefixes
  CPN . . . 185
    about . . . 185
    administering . . . 186
    prerequisites . . . 187
multiple call handling . . . 35

N

naming
  a vector . . . 218
naming a vector . . . 218
NCA-TSC not available . . . 724
NCR
  Bad NCR trunk admin . . . 720
  Internal system err . . . 721
  Invoke trunk not ISDN . . . 720, 722
  NCD call connect err . . . 721
  NCD invalid PSTN nmbr . . . 721
  NCT outgo trk drop . . . 720
  No NCD PSTN service . . . 721
  No NCT outgoing trk . . . 720
  No NCT PSTN service . . . 720
  PSTN NCD invoke err . . . 721
  PSTN NCD max redirs . . . 721
  PSTN NCD netwrk err . . . 721
  PSTN NCD no disc . . . 721
  PSTN NCT invoke err . . . 720
  PSTN NCT netwrk err . . . 720
  Used NCT trk-to-trk . . . 721
NCR. See Network Call Redirection
Negative Result . . . 722
Network Call Redirection
  administration . . . 397
  B-channels, reserving for redirected-to leg . . . 401
  feature
    interactions . . . 389
    limitations . . . 386, 414
  implementation
    ASAI adjunct routing . . . 397
    call vectoring . . . 391
    station call transfer or conference . . . 396
  Information Forwarding support . . . 387
  redirection options
    AT&T In-Band Transfer and Connect . . . 383
    ETSI-ECT . . . 382
    MCI . . . 381
    NCD . . . 382
    NCT-types . . . 380
    TBCT . . . 381
  route-to command . . . 393, 394
  SIP . . . 413
  SIP feature
    interactions . . . 415
  SIP Information Forwarding support . . . 414
  SIP troubleshooting . . . 420
  troubleshooting . . . 408
  trunking considerations . . . 387
neutral vector command . . . 276
Night Service . . . 435
No agent strategy found in VDN . . . 716
No AITCI storage left . . . 719
No best location found . . . 717
No BSR app num in VDN . . . 717
No BSR application plan administered . . . 717
No BSR Data in Response . . . 718
No digits in variable . . . 716
No enhanced info is sent . . . 720
No ETSI ECT linkID . . . 722
No response from status poll . . . 718
No return destination . . . 722

No room for collected dgt . . . 719
No room for in-VDN time . . . 719
No room for Other LAI . . . 719
No room for reply-best information . . . 719
No room for VDN Name . . . 719
non-business hours
call during . . . 233
non-business hours, call during . . . 233
none comparisons . . . 593
numbering
of vector steps . . . 225
numbering of vector steps . . . 225
numeric digits comparisons . . . 593

O

observing VDNs . . . 45
Operand Overflow Underflow . . . 722
option
VDN override . . . 39
option for VDN override . . . 39
originator's display . . . 292
Other LAI got bumped . . . 719
Overflow Error . . . 722

P

passing digits
to an adjunct . . . 242
to PBX . . . 38
passing digits to switch . . . 38
Path replacement . . . 211, 289, 355
percentage routing using VDN variables example . . . 796
performance
basic principles for improving . . . 855
effects of ASAI link failure . . . 206
evaluating . . . 752
effectiveness of vector programming . . . 752
for split . . . 760
improving . . . 857, 859, 861
example vector . . . 856, 857, 858, 861, 862
maximizing performance . . . 856
looping . . . 855
maximizing . . . 855, 857, 859, 861
. . . 856
processing cost
comparisons . . . 857, 858, 860
testing vectors . . . 856
personalizing call treatment . . . 56
Phantom call administration . . . 214
port-network vector conditional . . . 595
preventing unauthorized users access . . . 819
prioritizing calls . . . 34, 55, 227, 231
process
involving general number dialing
diagram of . . . 78
involving specific number dialing
diagram of . . . 77
processing calls
faster . . . 54
intelligently . . . 54
prompting a caller . . . 41
properties . . . 38
providing
call treatments . . . 46
caller feedback . . . 54
choices to callers . . . 55
faster service . . . 55
feedback . . . 228, 229, 230
initial feedback to caller . . . 35

Q

QSIG CAS . . . 436
QSIG MWI hunt group, using a messaging step in a vector 602
QSIG path replacement . . . 211, 289, 355
queue position example . . . 805
queue-to attd-group . . . 48, 424
queue-to attendant . . . 424
queue-to command . . . 48, 606
queue-to hunt-group . . . 424
queue-to main
neutral vector command . . . 612
queue-to main command
neutral vector command . . . 277, 612
success/failure criteria . . . 694
syntax . . . 606
troubleshooting . . . 698
Queue-to split command . . . 608
queue-to split command . . . 608
queuing calls
methods for . . . 32
to split . . . 33
maximum number of . . . 33

R

receiving feedback about a call . . . 228
recording announcements . . . 542
Redirect calls to VDNs . . . 434
redirecting calls
methods for . . . 32
redirecting calls, methods for . . . 32
reducing
caller hold time . . . 56
number of needed agents . . . 36
staffing requirements . . . 56
transferred calls . . . 33, 55
removing incorrect digits strings . . . 240

removing vector comments 226
reorder tone 697, 702, 770, 771
reply-best command 614
Reply-best got bumped. 719
reporting
  agent handling 38
  call handling 38
  using Basic Call Management System 38
  using BCMS 38
  using Call Management System 38
  using CMS 38
reports
  BCMS
    BCMS Split Report 762
    VDN Real-Time Report 763
    VDN Summary Report 762
  CMS
    Split Summary Report 762
    VDN Report 762
    Vector Report 762
requeuing calls 34
requirements
  software and hardware
    for adjunct routing 671, 672, 673
    for advanced vector routing 669
    for ANI/ii-digits routing 670
    for basic call vectoring 668
    for Best Service Routing 669
    for call prompting 668
    for look-ahead interflow 671
Results Truncated 722
Return command 617
return command 617
Return destination stack error. 722
route validation 533, 534
route validation failure 534
route-to
  look-ahead interflow 277
route-to command 619
  ~r vector step. 393, 394
  Network Call Redirection 393, 394
  neutral vector command. 277, 628
  summary of conditions for destination types . . . 767
  syntax 620
  troubleshooting 702
  variables and 117, 156
route-to digits 48
route-to number 48, 424
route-to number command 624
route-to requests
  multiple outstanding. 215
routing calls 33, 47, 55
  based on DNIS 36
  example table of call distribution using UCD/EAD . 489
  example table of UCD/EAD call scenario 489
  intelligently 273
  to an agent 487
    delivery from a skill hunt group 487
  to skill queue
    using call prompting 483
  using expert agent selection 487
  using super agent pool. 485

S

security
  main type of problem 819
  method
    front-ending remote access 819
      advantages 820
    replacing remote access 820
  methods for preventing remote access abuse . . 819
  preventing unauthorized users access 819
  replacing remote access 820
  with EAS 821
  with expert agent selection 821
  with remote access 819
  with service observing 822
  with vector initiated service observing. 822
SEL operations 778
Serv Observ Meet-me VDN 725
server conditionals 595
server vector conditional 595
Service Hours Table Routing 367
  administering 368
  considerations 369
  goto processing 367
  overview 367
  scenario 370
  time adjustments 368
Service Level Maximizer
  administration 510
  autoreserve agents 507
  feature interactions 513
  hardware and software requirements 503
  maximum agent occupancy 515
  reports 512
service observing 45, 248, 625
set command. 632
  variables and 118, 157
silence 46
  when occurs 36, 49
simulated bridged appearance. 846
skill
  definition 473
  example table for an auto club 474
  table for auto club application 485
  table of preferences assignments for VDN 1616 . 485
skill call
  example table of distribution for a single agent . 488

skill call queue sequence
example table . . . 488
skills
call handling preferences . . . 471
SLM. See Service Level Maximizer
SMDR. . . . 627
split
backup
definition . . . 33
main
definition . . . 33
staffed agents
check backup command . . . 34
conditional branching . . . 45
definition of . . . 34
for non-ACD hunt groups . . . 35
goto command . . . 34
number of . . . 48
Station Message Detail Recording, see SMDR
status lamp . . . 246
CALLR-INFO button . . . 246
NORMAL button . . . 246
Status Poll VDN field is blank . . . 717
stepcnt type variable . . . 125
steps
maximum number of . . . 45
stop . . . 424
stop command . . . 49
example . . . 643
neutral vector command . . . 277, 643
success/failure criteria . . . 695
syntax . . . 642
troubleshooting . . . 702
string operations
examples . . . 787
rules . . . 776
start and length . . . 779
subroutines . . . 165
system-assigned vector variables . . . 121

T

tandem switch
far end operation . . . 288
far end switch operation . . . 288
sending switch operation . . . 287
Tenant night destination . . . 436
Tenant Partitioning and agent skill . . . 497
testing call treatment . . . 55
testing vectors . . . 674
Time Zone Offset . . . 39
tod type variable . . . 127
tones
call waiting . . . 472
tracking
agents and their skills . . . 763
calls . . . 752
direct agent calls . . . 764
example
split flow . . . 755
for abandoned calls . . . 757
for call answered
after route to split . . . 760
by a primary split . . . 756
by non-primary split . . . 757
after route to VDN . . . 759
for non-ACD calls . . . 764
VDN skill preferences . . . 764
transfer call management control
caller-selected routing . . . 33
messaging . . . 33
treating digits as a destination . . . 241, 242
troubleshooting
1,000 step executed . . . 710
AAS split cannot queue . . . 723
adjunct
link error . . . 713
route cancelled . . . 713
route failed . . . 713
route link invalid . . . 713
administration change . . . 710
agent
drops converse . . . 715
not logged in . . . 713
not member of split . . . 713
receiving phantom call . . . 695
all look-ahead interflow attempts accepted . . . 696
all trunks busy on a quiet system . . . 697, 702
alternate audio/music source not heard . . . 703
ANI digits not passed . . . 700
ANI not avail - digits . . . 715
ANI not avail - table . . . 716
announcement not heard . . . 697, 700
while waiting for digits . . . 698
ASA - invalid VDN . . . 715
ASA - no staffed agents . . . 723
ASAI transfer converse . . . 715
Attd Vec Cannot requeue . . . 711
Attd Vec Mismatch-CR/Vec . . . 710
Attd Vec Mismatch-VDN/Vec . . . 710
audible feedback
lasts longer than the delay interval . . . 695
longer than delay interval . . . 703
shorter than delay interval . . . 703
AUDIX link down . . . 723
branch is not made
to the specified step . . . 701
to the specified vector . . . 701
busy step for CO trunk . . . 714
busy tone . . . 702

call apparently answered in wrong order 698
call cannot be queued. 710
call does not enter queue or terminate to agent . . 698
call dropped 702, 709
call dropped by vector disconnect timer 710
call stuck in converse 704
caller information button denied 699
Can't connect idle agent. 723
collect
  announcement
    not heard 706
    not heard and first collected digit incorrect . 700
collect step and announcement skipped 699
converse
  drop during data. 714
  no ANI digits 714
  no prompt digits 714
  no qpos digits 714
  step skipped 704
  transfer denied 715
coverage conference denied 715
data return
  no digits. 715
  timeout 715
delay before AUDIX answers 702
delay before hearing announcement 699
dial-ahead digits not recognized 699
dial-ahead discarded 711
digits incomplete 705
double coverage attempt 713
expected wait-time
  call no working agents 724
  call not queued 716
  no split queue 723
  not sent to VRU 716
  split locked 724
  split no working agents. 724
  split queue full. 723
expected wait-time no history for split 723
extra delay 701
  before hearing announcement 697
first set of digits not collected 704
ii-digits not avail - digits 716
ii-digits not avail - table 716
incomplete announcement 697, 700
insufficient digits collected
  call routed to intercept 699
invalid
  destination 713
  direct agent 713
  EAS hunt group used in the vector step 715
look-ahead
  DNIS name not displayed 696
  interflow retry 713
messages not found 701
messaging step failed 712
multiple observers
  maximum observers reached 713
  observing observer. 713
music not heard 703
network reorder 702
no announcement available 710
no available trunks 712
no data returned from VRU 700
no digits
  collected. 714
  to route-to 712
no entries in routing table 716
no look-ahead interflow attempts accepted . . . 696
no Touch-Tone Receiver available 711
no vector steps, ANI sent 715
not a messaging split 723
not all digits returned to the DEFINITY switch . . 706
not vector-controlled. 723
prompting buffer overflow 711
qpos digits not passed. 700
queue before route 713
queued to three splits 711
redirect
  of call failed 714
  unanswered call 714
retrying announcement 710
ringback heard instead of busy tone 698
route -to step failed 712
route-to step failed. 712
routing table not assigned 716
second set of digits
  is the same as the first digits passed 705
  not collected 705
service hours
  table empty 714
skill indirection used improperly 715
split queue is full 723
step skipped 697, 702
  no message left 701
  that is, default treatment 702
steps
  display event report 708
  display events screen 707
system clock change 716
time not set 714
unexpected
  busy tone 696
  intercept or reorder tone heard 697
  network reorder or intercept. 697
  silence after announcement. 697
  step skipped (that is, default treatment) . . . 696
unexpected intercept or reorder tone heard . . . 702
vector processing halted at collect step, announcement heard again upon return 699
vector processing stops 697

vector stuck . . . 695, 699, 701
with busy . . . 701
with ringback . . . 701
vector with no steps . . . 709
VRU script
not executed . . . 700
terminated prematurely . . . 700
wait digits not passed . . . 700
wait step
music failed . . . 714
ringback failed . . . 714
Trunk group incoming destination . . . 436

U

upgrading
a contact center to expert agent selection . . . 843
to a call vectoring environment . . . 673
user-assigned variable types . . . 130
using digits
to collect branching information . . . 241
UUI . . . 205
uui sent to CMS, but there were no steps in the vector 715

V

valid entries
for converse-on . . . 566
validating numbers by matching segments example . 794
value type variable . . . 135
Variable not defined . . . 712
variable types
system-assigned . . . 121
user-assigned . . . 130
Variables in Vectors
administration . . . 135
considerations . . . 135
example vectors . . . 139
FAC, using to change variable values . . . 138
failure conditions . . . 135
feature overview . . . 107
hardware and software requirements . . . 119
implementing . . . 109
interactions, with CMS . . . 135
job aid . . . 110
scope, local and global . . . 119
system-assigned variable types . . . 121
troubleshooting . . . 149
user-assigned variable types . . . 130
variable access codes for FAC . . . 109, 138
VDN override interactions . . . 128
Var-in-vec COS restricted . . . 724
Var-in-Vec Invalid digit . . . 724
Var-in-Vec No adm for VAC . . . 724
VDN . . . 38
active . . . 39
definition . . . 32, 38, 53
in coverage path
application uses . . . 44
latest . . . 39
multiple . . . 53
observing . . . 45
override
example application . . . 41
return destination
considerations . . . 683
VDN Calls
how call counts are calculated . . . 269
VDN Name got bumped . . . 719
VDN not a meetme type . . . 725
VDN Override
described . . . 39
VDN Override for ISDN Trunk ASAI Messages . . . 42
VDN Time Zone Offset . . . 39
vdn type variable . . . 127
vdntime type variable . . . 129
vector
changing existing . . . 218, 674
creating a new . . . 218
definition . . . 37, 53
disconnect timer . . . 673
entering . . . 217
entering comments . . . 225
events . . . 707, 709
example . . . 602
accommodate a super agent pool . . . 486
automated attendant application . . . 60
call interflow . . . 625
claims application . . . 70
customer service application . . . 71
delay with multiple audio/music source feedback 647
dial-ahead digits . . . 252
distributed call centers application . . . 66
DIVA and data/message collection application 62, 63
emergency and routine service application . 96, 98
help desk application . . . 68
late caller application . . . 100
leaving recorded message . . . 611
leaving recorded messages . . . 611
messaging options application . . . 102
multiple split queueing . . . 609
passing digits to an adjunct . . . 247
receiving switch inflow vector . . . 280
remote access service observing vector . . . 248
return destination vector
with announcement . . . 687
with remote access . . . 685
service agency clients application . . . 71
service observing vector . . . 248, 249
stopping vector processing . . . 643

tandem switch vector . . . 288
testing
for digits in vector routing table . . . 244
treating digits as a destination . . . 242
unconditional branching . . . 591
using digits
to collect branching information . . . 243
to select options . . . 245
vector for service observing . . . 626
listing existing . . . 218
naming . . . 218
removing comments . . . 226
testing . . . 674
variables. See Variables in Vectors
vector chaining
using the route-to number command . . . 624
vector command
adjunct routing command . . . 47, 530
advanced vector routing . . . 258, 424
announcement command . . . 538
announcements . . . 47
busy . . . 47, 545
call denial
qualification of commands . . . 276
call prompting . . . 238
command table . . . 238
check-backup . . . 47, 548
collect digits . . . 47, 556
comparison operators . . . 50
condition testing . . . 49
consider . . . 561
converse-on . . . 566
converse-on command . . . 48
disconnect . . . 48
disconnect command . . . 579
function of each . . . 526
goto step . . . 48
goto step command . . . 583
goto vector . . . 48
maximum number . . . 220
messaging . . . 48, 600
neutral
qualification of commands . . . 276
OCM predictive calls . . . 496
parameters . . . 653
queue-to . . . 48
queue-to command . . . 606
reply-best . . . 614
route-to . . . 619
route-to digits . . . 48
route-to number . . . 48
stop . . . 49
success/failure criteria . . . 690
syntax . . . 653
wait-time . . . 49, 645
vector commands . . . 305
Vector commands for single-site BSR
Single-site BSR . . . 305
vector commands that use ced and cdpd . . . 182
vector comments . . . 527
vector directory number
definition . . . 32, 38
properties . . . 38
Vector Directory Number screen . . . 38
implementation notes-list . . . 38
screen–add/change . . . 310, 314, 327, 335, 800, 801
vector event
advantages of tracking unexpected . . . 707
displaying . . . 707
unique number . . . 708
with debugging . . . 704
vector processing
ASAI link failure . . . 532
BCMS Report
description . . . 762
BDMS Report
description . . . 762
branching . . . 45, 47, 48
collecting from caller . . . 48
control flow . . . 37
types of . . . 45
failure
converse-on step . . . 574
resulting in these destinations . . . 627
maximum number of steps . . . 45
programming
collecting and acting on information . . . 47
collecting from caller . . . 47
providing treatments . . . 47
routing calls . . . 47
programming capabilities
branching . . . 45
Split Summary Report
description . . . 762
stopping . . . 32, 45, 46, 47, 103, 442
terminating . . . 611
termination . . . 47
termination vs stopping . . . 46
troubleshooting . . . 695
VDN Real-Time Report
description . . . 763
VDN Report
description . . . 762
VDN Summary Report
description . . . 762
Vector Report
description . . . 762
with coverage . . . 44, 242
vector routing table . . . 243
vector routing tables

using ANI numbers in . . . 172
vector step
conditional branching . . . 45
deleting . . . 224
entering . . . 218
example
announcement . . . 539
converse-on . . . 567
inserting . . . 224
maximum number . . . 32
numbering . . . 225
sequential flow . . . 45
stopping . . . 46
terminating . . . 46
termination vs stopping . . . 46
unconditional branching . . . 45
vector subroutines . . . 165
vector variable information, viewing . . . 221
Vector variable types . . . 121
vector variable types, see variable types
vector-controlled split . . . 609, 610
vectors and messaging steps . . . 602
viewing vector variable information . . . 221
voice response script . . . 570
VRI
advantage of . . . 571
capabilities . . . 570
VRU
advantages of . . . 570
execution of VRU script . . . 570
normal override rules . . . 577
offloading recorded announcements to . . . 575
outpulsing data . . . 568, 572, 576
passing data between VRU and DEFINITY switch 570
passing EWT to . . . 571
returning data to the switch . . . 567
service observing pending mode . . . 576
storing received data . . . 567
tandemed to ASAI host . . . 570
used as an external announcement . . . 570
using digits returned from . . . 557
VRU digits
conditional branching . . . 567
displayed using CALLR-INFO button . . . 567
extension in a route-to command . . . 567
tandemed to an ASAI host . . . 567

W

wait command
variables and . . . 118
wait-time . . . 49, 424, 645
factors that affect prediction accuracy . . . 258
predictions
circumstances that will limit . . . 258
when to use predictions . . . 258
wait-time command
example . . . 647
neutral vector command . . . 277, 651
success/failure criteria . . . 695
syntax . . . 645
troubleshooting . . . 703
wildcards . . . 243
work mode
ACW mode . . . 35
auto-in work mode . . . 35
auxiliary-work mode . . . 35
manual-in work mode . . . 35
Wrong MM Acc. code dialed . . . 725