Programming Call Vectors in Avaya Aura™ Call Center

6.0
June 2010

Notice

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

Documentation disclaimer

Avaya shall not be 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. 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 End User.

Link disclaimer

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

Warranty

Avaya provides a limited warranty on 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 to Avaya customers and other parties through the Avaya Support Web site: http://www.avaya.com/support. Please note that if you acquired the product from an authorized Avaya reseller outside of the United States and Canada, the warranty is provided to you by said Avaya reseller and not by Avaya.

Licenses

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

Avaya grants 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 originally sold by Avaya and ultimately utilized by End User.

Copyright

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

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 regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://www.avaya.com/support/Copyright/.

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/. Suspected security vulnerabilities with Avaya products should be reported to Avaya by sending mail to: securityalerts@avaya.com.

Trademarks

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

All 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

Contact 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 Web site: http://www.avaya.com/support

Contents

Chapter 1: Related documents

Other Call Center documents

These additional documents are issued for Avaya Call Center applications:

- *Avaya Aura™ Call Center Overview*: Provides a high-level overview of the features available in Call Center. This document also explains the new features added in the current release.
- *Planning an Avaya Aura™ Call Center Implementation*: Provides information on setting up a call center and converting a call center to Expert Agent Selection (EAS).
- *Avaya Aura™ Call Center Feature Reference*: Provides detailed information on the various ACD and Call Vectoring features, including the relevant command and screens for each of the features.

  Following are the Call Vectoring related topics in this document:

  - Adjunct Routing
  - Advanced Vector Routing – EWT and ASA
  - Attendant Vectoring
  - Best Service Routing
  - Call flow and specifications for converse-on command
  - Holiday Vectoring
  - Information Forwarding
  - Look-ahead Interflow
  - Meet-Me Conference
  - Percentage Allocation Routing
  - Service Hours Routing
- *Administering Avaya Aura™ Call Center Features*: Provides information on how to administer Call Center using the various screens and how to work with Time of Day Clock Synchronization, Recorded Announcements, and VRUs/IVRs as Station Ports.
- *Communication Manager Call Center 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.

- *Avaya IQ Documentation DVD*: Provides information about Avaya's software-only reporting solution for its contact center portfolio.
- *Avaya Aura™ Communication Manager System Capacities Table*: Provides information about offer-defined capacities for various Avaya Aura™ server platforms.

Associated application documentation

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

Chapter 2: Call Vectoring overview

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.

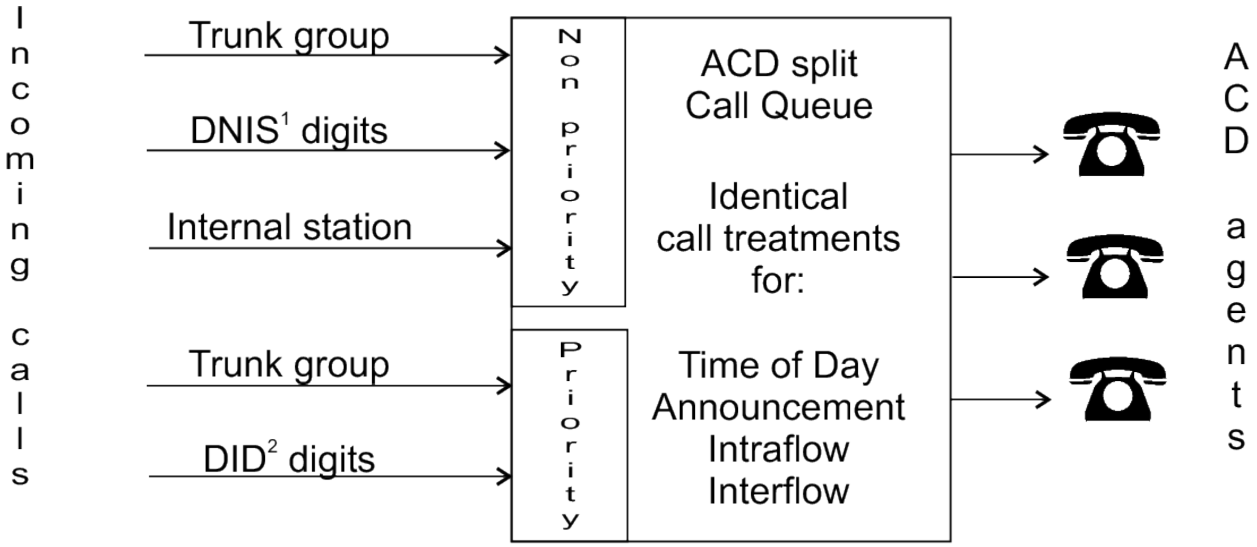
Note:

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.



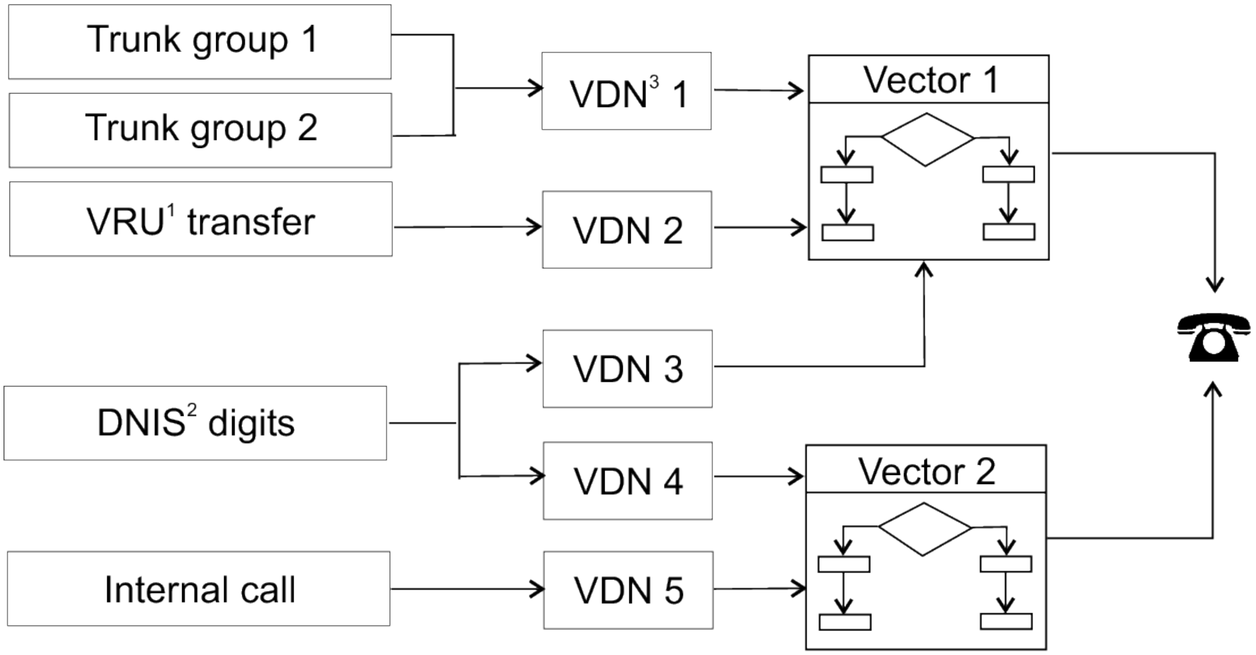
1. Dialed Number Identification Service
2. Direct Inward Dialing

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.



1. Voice Response Unit
2. Dialed Number Identification Service
3. 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 Setting up a call center in the *Planning an Avaya Aura™ Call Center Implementation* document.

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 on page 42 Conditional branching on page 187 Example application - distributed call centers on page 48
Automatically change treatment according to either how long the call has been waiting or in response to changing traffic or staffing conditions.	Automated attendant on page 43 Example application - mutual fund company on page 45 Example application - distributed call centers on page 48 Example application - help desk on page 50 About interflow routing on page 297 Using Call Prompting to route by collected digits (See*Avaya Aura*™ *Call Center Feature Reference*.) Using Call Prompting to branch by collected digits (See*Avaya Aura*™ *Call Center Feature Reference*.)
Provide appropriate caller feedback during waiting. For example, music or announcements during heavy calling periods.	Delay announcement on page 221 Forced announcement example on page 222 Information announcement example on page 222 Call delay with audible feedback on page 320 Multiple music sources on hold on page 322 Call delay with continuous audible feedback on page 321
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 on page 42 Leaving recorded messages (VDN as The coverage point option) on page 285 About interflow routing on page 297
Provide 24 hour or day, 7 day or week automated information announcements.	Information announcement example on page 222 Call delay with audible feedback on page 320
Remove selected calls (by providing busy or disconnect).	busy command for Call Vectoring on page 226 Call disconnect example on page 257

Call Vectoring Benefits	Examples
	Leaving recorded message on page 277 Unconditional branching example on page 266
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 222 Setting up a Holiday Table Changing vector processing for holidays (both examples)
Provide the caller with a menu of choices.	Example application - mutual fund company on page 45 Example application - help desk on page 50 Using Call Prompting to route by collected digits (See*Avaya Aura*™ *Call Center Feature Reference*.) Passing digits to an adjunct (See *Avaya Aura*™ *Call Center Feature Reference*) Dial-ahead digit vector examples (See*Avaya Aura*™ *Call Center Feature Reference*.)
Queue calls to up to three splits simultaneously, consequently improving the average speed of answer and agent productivity.	Customer service center on page 42 Example application - distributed call centers on page 48 Multiple split queuing on page 283
Implement routing to local or distant destinations.	Customer service center on page 42 Example application - mutual fund company on page 45 Example application - distributed call centers on page 48 Example application - help desk on page 50 About interflow routing on page 297 Using Call Prompting to route by collected digits (See the *Avaya Aura*™ *Call Center Feature Reference* document) Using Call Prompting to branch by collected digits (See the *Avaya Aura*™ *Call Center Feature Reference* document)
Connect callers to a voice-mail or messaging system either automatically or per caller request.	Example application - mutual fund company on page 45 Leaving recorded messages (VDN as The coverage point option) on page 285 Leaving recorded message on page 277
Call Routing	
Reduce call transfers by accurately routing callers to the desired destination.	Example application - mutual fund company on page 45 Using Call Prompting to route by collected digits (See*Avaya Aura*™ *Call Center Feature Reference*.) Using Call Prompting to branch by collected digits (See*Avaya Aura*™ *Call Center Feature Reference*.)

Call Vectoring Benefits	Examples
Provide up to four ACD queuing priority levels and the ability to change the queuing priority dynamically, thereby, providing faster service for selected callers.	Customer service center on page 42 Example application - mutual fund company on page 45 Example application - distributed call centers on page 48
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 45 Information announcement example on page 222 Call delay with audible feedback on page 320 Using Call Prompting to route by collected digits (See*Avaya Aura™ Call Center Feature Reference*.) Dial-ahead digit vector examples (See*Avaya Aura™ Call Center Feature Reference*.)
Information Collection	
Provide customized or personalized call treatment using information collection and messaging.	Automated attendant on page 43 Example application - mutual fund company on page 45 Example application - help desk on page 50 Using Call Prompting to route by collected digits (See*Avaya Aura™ Call Center Feature Reference*.) Treating digits as a destination (See*Avaya Aura™ Call Center Feature Reference*.) Dial-ahead digit vector examples (See*Avaya Aura™ Call Center Feature Reference*.)
Collect information for use by an adjunct or by agent display.	Example application - help desk on page 50 Passing digits to an adjunct (See *Avaya Aura™ Call Center Feature Reference*)
Collect caller-entered or customer database-provided CINFO digits from the network.	CINFO vector example on page 167

Chapter 3: Call Vectoring fundamentals

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

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 283.

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.

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.

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 Feature 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.

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

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.

Vector Directory Number implementation notes

The following list describes special situations due to the type of Communication Manager 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 *Administering Avaya Aura™ Call Center Features*.

VDN variables

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

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 Feature Reference*.

VDN Override

Overview of VDN Override

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 **Allow VDN Override?** field on the VDN screen, see the *Administering Avaya Aura™ Call Center Features* document.

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 ASAI messages on page 32
- BSR Local Treatment
- VDN Variables
- VDN Time Zone Offset

Application of VDN Override

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 110.

Detailed operation of VDN Override

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 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 trunk or incoming trunk and internal 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.

Related topics:

ASAI messages on page 32
VDN Override feature interactions on page 33

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 ASAI Messages feature is activated for an incoming call when the call is routed to a VDN that has the **VDN Override for 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 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 ASAI Messages** setting as follows:

- If set to `n`, the "Called Number" information sent for the "Call Offered," "Alerting," "Queued," and "Connect" ASAI event notification messages and the adjunct-request message is always the called VDN extension in the Called Number IE sent in the incoming ISDN SETUP message or the local call's called number and does not change after routing to the called VDN and subsequent routed to VDNs.
- If set to `ISDN Trunk`, when an incoming ISDN trunk call is routed to this VDN, the "Called Number" information sent in the ASAI event and "Adjunct Route Request" ASAI messages is the "active VDN" extension that becomes associated with the call based on the VDN Override rules. This option does not apply to local/internal calls.
- If set to `all`, the active VDN is used for the called number for all types of calls to the VDN including local/internal calls as well as external incoming ISDN trunk calls.

VDN Override feature interactions

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

- If an incoming call has the VDN Override for 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 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 284. For information about interactions, see Avaya Aura™ Communication Manager Feature Description and Implementation.

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.

You can also set up a generic VDN to process the calls redirected due to RONA, ROIF, and ROOF. For more information on generic VDNs, see Generic VDNs for redirected calls handling in the *Avaya Aura™ Call Center Feature Reference* document.

For more information on Redirect on no answer, see the Redirection on No Answer in the *Avaya Aura™ Call Center Feature Reference* document.

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 Feature 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

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.

- 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 the chapter Operation details for the route-to command.
- Route-to Number routes the call to the destination specified by the administered digit string. For more information, see the chapter Operation details for the route-to command.
- 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 Call Vectoring 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.

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 Feature availability in the *Planning an Avaya Aura™ Call Center Implementation* document.

- 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.

Chapter 4: Call Vectoring application examples

List of applications

Application examples and the primary feature that is illustrated are listed in the following table.

Example	Features used
Customer service center on page 42	Basic Call Vectoring
Automated attendant on page 43	Call Prompting
Data in/voice answer and data/message collection on page 44	Call Prompting, Basic Call Vectoring
Distributed call centers on page 48	Look-Ahead Interflow, Basic Call Vectoring
Help desk on page 50	Adjunct Routing, Call Prompting, Basic Call Vectoring
Insurance agency/service agency on page 51	Basic Call Vectoring, Call Prompting, Rolling ASA, EWT, VDN Calls, and ANI Routing
Warranty service (with EAS) on page 54	Basic Call Vectoring, EAS
Resort reservation service (with EAS)	Basic Call Vectoring, Adjunct Routing, Call Prompting, EAS
Local attendant group access code on page 63	Attendant Vectoring
Incoming trunk calls to attendant group on page 64	Attendant Vectoring
Incoming LDN calls on page 64	Attendant Vectoring
QSIG CAS example on page 65	Attendant Vectoring
Night station service example on page 66	Attendant Vectoring
Holiday Vectoring example on page 67	Holiday Vectoring
Network Call Redirection	BSR multi-site, NCR

Example	Features used
BSR using EWT and agent adjustments	BSR multi-site
Dial by Name on page 73	Basic Call Vectoring, Call Prompting

Customer service center

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 VDNVector 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

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

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

```
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

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). To learn how to integrate distributed call centers using multi-site Best Service Routing, see Best Service Routing (BSR).

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

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). 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

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.

Table 1: 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)

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:

Table 2: 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.

Table 3: 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
Spanish	Dishwasher	1200	21	20	--

VDN skill preference	Appliance	VDN	First	Second	Third
	Refrigerator	1201	22	20	--
	Washer	1203	41	40	--
	Dryer	1204	42	40	--

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

Table 4: 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	--

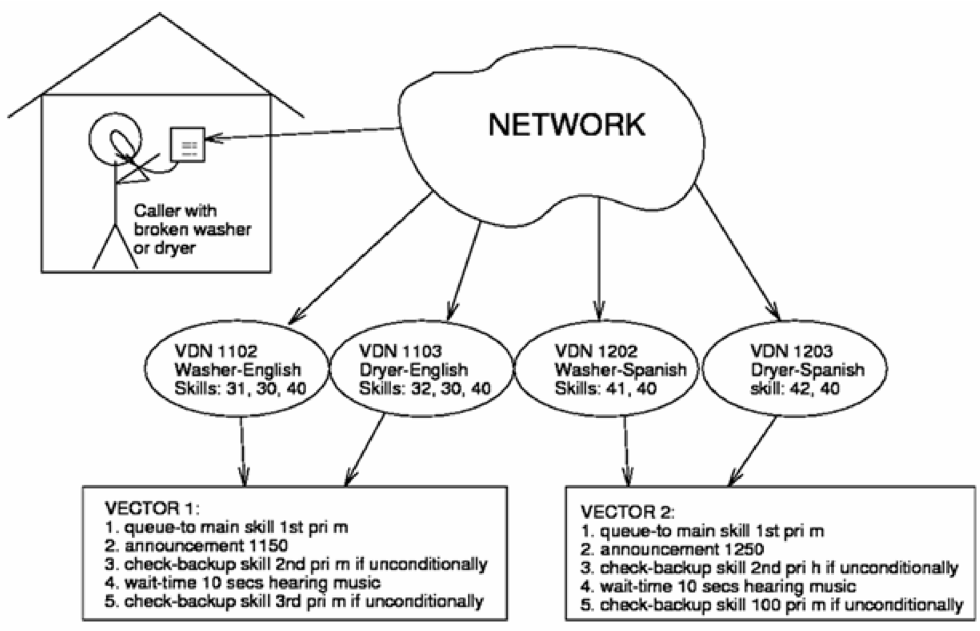
Once skills are assigned to VDNs and to agents, calls are directed to the appropriate vector.

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.

Figure 1: Warranty service call center (part 1) on page 56 and Figure 2: Warranty service call center (part 2) on page 56 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.


NETWORK
Caller with broken washer or dryer
VDN 1102 Washer-English Skills: 31, 30, 40
VDN 1103 Dryer-English Skills: 32, 30, 40
VDN 1202 Washer-Spanish Skills: 41, 40
VDN 1203 Dryer-Spanish skill: 42, 40

```
VECTOR 1:
1. queue-to main skill 1st pri m
2. announcement 1150
3. check-backup skill 2nd pri m if unconditionally
4. wait-time 10 secs hearing music
5. check-backup skill 3rd pri m if unconditionally
```

```
VECTOR 2:
1. queue-to main skill 1st pri m
2. announcement 1250
3. check-backup skill 2nd pri h if unconditionally
4. wait-time 10 secs hearing music
5. check-backup skill 100 pri m if unconditionally
```

Figure 1: 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.

Figure 2: 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.

Notifying callers of queue position

Scenario description

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.

Scenario solution

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

Related topics:

Scenario prerequisites on page 57
Tips for setting up the interflow-qpos conditional on page 58

Scenario 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
```

Resort reservation service (with EAS)

About the resort 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.

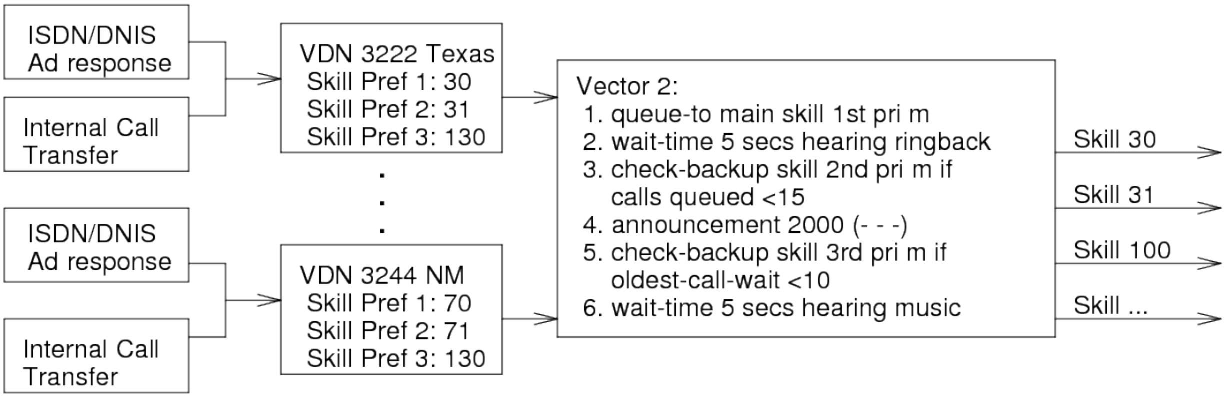
Specific number dialing

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.

Table 5: 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.



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.

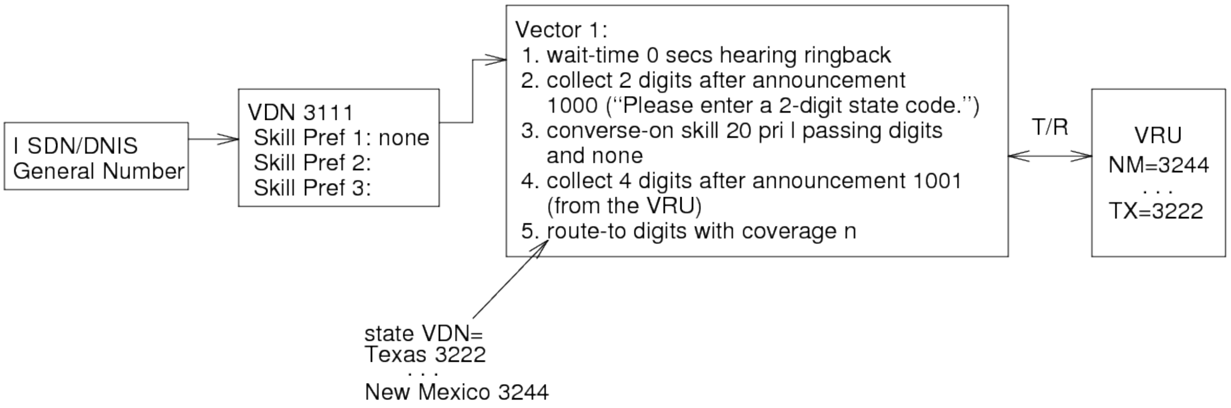

I SDN/DNIS
General Number
VDN 3111
Skill Pref 1: none
Skill Pref 2:
Skill Pref 3:
Vector 1:
1. wait-time 0 secs hearing ringback
2. collect 2 digits after announcement 1000 (“Please enter a 2-digit state code.”)
3. converse-on skill 20 pri l passing digits and none
4. collect 4 digits after announcement 1001 (from the VRU)
5. route-to digits with coverage n
T/R
VRU
NM=3244
. . .
TX=3222
state VDN=
Texas 3222
. . .
New Mexico 3244

Figure 3: 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 83.

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.

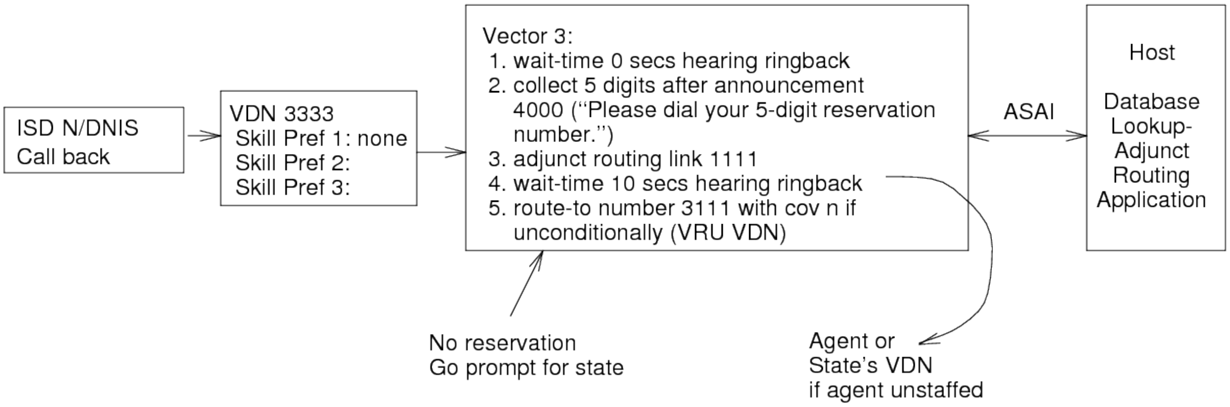

ISD N/DNIS
Call back
VDN 3333
Skill Pref 1: none
Skill Pref 2:
Skill Pref 3:
Vector 3:
1. wait-time 0 secs hearing ringback
2. collect 5 digits after announcement 4000 (“Please dial your 5-digit reservation number.”)
3. adjunct routing link 1111
4. wait-time 10 secs hearing ringback
5. route-to number 3111 with cov n if unconditionally (VRU VDN)
ASAI
Host
Database Lookup-Adjunct Routing Application
No reservation
Go prompt for state
Agent or
State’s VDN
if agent unstaffed

Figure 4: 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 on page 60 section.

Attendant routing

Attendant routing example

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:

Table 6: 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 63 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.

Related topics:

CAS branch on page 65
CAS main on page 66

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

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) in the *Avaya Aura™ Call Center Feature Reference* document.

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 Administering Network Connectivity on Avaya Aura™ 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 Feature Reference*.

Primary Vector for Network Call Redirection example

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

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

Status poll vector for Network Call Redirection example

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 split   2  pri m   adjust-by 0
2. consider split 11    pri m  adjust-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 for Network Call Redirection example

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

The public network 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

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.

Table 7: 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

Primary Vector for BSR using EWT and agent adjustments example

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

In this example, the **`consider`** commands in steps 2, 3, and 4 collect information to compare local split 1 with location 2 and location 3. In each case, an available agent is found and an agent idle time returned. The adjust-by in steps 3 and 4 adjusts the value of the agent idle time as shown in table Table 7: Idle time adjustment calculations on page 72. Step 5 queues the call to the best location found.

Status poll vector for BSR using EWT and agent adjustments example

To collect information from 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 split   2  pri m   adjust-by 0
```

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

```
2. consider split 11    pri m  adjust-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 for BSR using EWT and agent adjustments example

Based on the values derived in table Table 7: Idle time adjustment calculations on page 72, 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 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 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:

        i. 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 ii on page 75.
        - If there is no match, go to iiii on page 76.

        ii. 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 iii on page 75.
        - If there is no match, go to iV on page 76.

        iii. 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 iV on page 76.

    iv. 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.

Related topics:

Emergency and routine service on page 76
Late Caller Treatment on page 79
Messaging option on page 81

Emergency and routine service

Write a vector that does the following:

1. 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.
2. Enables the caller to speak with an agent, if an agent is available, concerning a non emergency matter.

Related topics:

Emergency and routine service suggested solution 1 on page 77
Emergency and routine service suggested solution 2 on page 77

Emergency and routine service 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.

Emergency and routine service suggested solution 2

Note:

This example uses the Call Prompting feature. For more information about Call Prompting, see Call Prompting.

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.

Related topics:

Late Caller Treatment suggested solution on page 79

Late Caller Treatment 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:

1. 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.*
2. Plays 30 seconds of ringback for the caller
3. After the ringback, plays an announcement for the caller that is followed by music

Result

Suggested solution

Messaging option

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

Chapter 5: Basic Call Vectoring

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.

Related topics:

Basic Call Vectoring command set on page 83
General considerations for Basic Call Vectoring on page 85

Basic Call Vectoring command set

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

Description	Command
Treatment steps	
Play an announcement.	**announcement**
Delay with audible feedback of silence, ringback, system music, or alternate audio or music source.	**wait-time**
Play a busy tone and stop vector processing.	**busy**
Disconnect the call.	**disconnect**
Execute a Voice Response Unit (VRU) script.	**converse-on split**
Routing steps	
Queue the call to an ACD split.	**queue-to split**
Queue the call to a backup ACD split.	**check split**
Leave a message.	**messaging split**
Route the call to a number that is programmed in the vector or to a Service Observing Feature Access Code.	**route-to number**
Send a message to an adjunct that requests routing instructions for the call.	**adjunct routing link**

Description	Command
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

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.

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.

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.

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.
- 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 Duplicating Vectors on page 191.

Chapter 6: Variables in Vectors

About VIV

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 (in uppercase A to Z and AA to ZZ) to be used as either conditionals or thresholds or both in many commands. These letters entered in uppercase 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:

Parameter	Description
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 104.
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* , *local persistent* (for collect variable only), or *global*. Local variables use data associated with a call and the value assigned to the variable only apply within the original vector processing for the call (the value is cleared after the call leaves vector processing). Local persistent variables use data associated with a call and apply to one or more vectors that process the call (the last assigned value is retained throughout the life of the call). Global variables are system wide and apply to all vectors in which they are used. For more information, see Definition of local, global, and local persistent variables on page 102.
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 118.

Implementing vector variables

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

1. 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 90 and for more detailed descriptions, see System-assigned vector variable types on page 104.

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

3. In the Variable for Vectors administration table, select any unused variable name between A - Z or AA - ZZ. This variable name is used to represent the variable in vector steps. In the table row for the variable that you have selected, enter the following information in the specified fields:

   a. **Description** - Enter a short description of your variable.

   b. **Type**- Select the variable type.

   c. **scope** -(local, local persistent (for collect variable only), or global),

   d. **length** - Enter length of the digit string.

   e. **start** - Enter digit start position (first position is ‘1’) and

   f. **assignment**- Enter an initial value.

   g. **VAC** - (optional, value variables only).

   Note:

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

4. If you administer a value variable type in the **VAC** field 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.

   b. 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.

   c. In the **Code** field, provide the digits that you want to dial when you access the FAC.

   d. Go back to the Variables for Vectors administration table and enter the VVx number in the VAC column for the value variable that you are administering.

   For more detailed information, see VIV administration on page 119.

5. 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.

6. If required, change the variable assignments.

   Some variables, such as the ani and tod variable types, do not require value reassignments after the variables are implemented in vectors, since values for the variable are always provided by individual callers or the Communication Manager. 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, the value assigned for the variable changes when a caller is prompted by an announcement and enters new digits or changed by a **`set`** command.

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 93.For descriptions of a few basic ways that you can apply variables in your call vectors, see VIV vector examples on page 122.

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	Holds the caller's phone number	*L*	Start digit position and Length	16	Incoming call data
asaiuui	Holds call-specific user data associated with the caller	*L*	Start digit position and Length	16 out of a total of 96	Incoming call or ASAI application data
collect	Holds user defined digits associated with the call for control, routing or	L, P, or G	Start digit position and Length	16	The for parameter of the collected digits command or

Variable type	Description	Scope	Specification	Max digit length	Assigns
	special treatment that can be assigned a value by the Variables for Vectors table, collect digits steps or the set vector command.				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	Provides the count of vector steps executed for the call, including the current step	*L*	None	4	The vector processing step counter
value	Holds a single numerical digit (0-9) for user-defined processing	*G*	None	1	A user-defined value entered using the VAC FAC procedure or assignment in the variables table
vdn	Holds the VDN extension number of the call for processing	*L*	Active or Latest VDN	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

announcement commands with vector variables

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 variables in vector commands

The requirements for using variables in place of specific entries in vector commands 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 or VDN screen for the active VDN for the call..
- The number that the variable expands to during vector processing must be a valid entry for the command parameter.
- The number obtained by the variable during processing must be a valid announcement extension assigned on the Audio/Announcement screen.

See also:

- announcement command for Call Vectoring on page 219
- announcement commands with VDN variables on page 137

collect command with vector variables

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 for collect command with vector variables

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 as the announcement extension, see announcement commands with VDN variables on page 137.

converse-on command with vector variables

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

```
converse-on

split
 [
hunt group
,

1st
```

[2] 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.

```
,
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
]
```

Requirements and considerations for converse-on command with vector variables

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 in the *Avaya Aura™ Call Center Feature Reference* document.
- 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 244
- converse-on command with VDN variables on page 138

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 VDN variables on page 137.

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

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

See also:

- disconnect command for Call Vectoring on page 256
- disconnect command with VDN variables on page 140

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` 1-8000 @ 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 = low m = medium h = high t = top	>>= <>= <<=	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 = low, m = medium, h = hight = top	>>= <>= <<=	A-Z, AA-ZZ
	for skill	hunt group, skills for VDN:1st, 2nd,3rd				
holiday	in table	A-Z, AA-ZZ				
	not-in table					
ii-digits	>,>=, <>, =, <, <=	A-Z, AA-ZZ				
	in table	A-Z, AA-ZZ				

`goto step` 1-99 if or `goto vector` 1-8000 @ step 1-99 if						
	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 = low, m = medium, h = hight = top	>, >=, <., =, <, <=	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 = low, m = medium, h = hight = top	>, >=, <>, =, <, <=	
	split	hunt group				

Requirements and considerations for goto commands with vector variables

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 259
- goto commands with VDN variables on page 140

route-to number command with vector variables

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

`route-to number`	up to 16 digits (0-9) <digits>[3][A-Z, AA-ZZ][4] <digits>*<digits>A <digits>#<digits>A <digits>~p<digits>A <digits>~m<digits>A <digits>~s<digits>A <digits>~w<digits>A <digits>~W<digits>A ~r[5]<digits>A ~r+[5]<digits>A	with cov	y,n	if	digit	>, >=, <>, =<, <=	`0-9, #`
					interflow-qpos	<, =, <=	1-9
					unconditionally		

[3] <digits> notation for 0 or more digits in the range of 0-9.

[4] 2. Indicates that a vector variable [A-Z, AA-ZZ] can be entered here. Also shown by an “A” notation.

[5] ~r invokes Network Call Redirection over the incoming trunk. ~r+ invokes NCR with E.164 numbering notation for incoming SIP trunking when required by the Service Provider.

Requirements and considerations for route-to command with vector variables

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

- A vector variable can be used in the number field as the destination address for the **`route-to`** command alone or in combination with special character designations. When the route-to number step with a variable is executed, the current numerical value, or assignment of the variable, of up to 16 digits is used for the destination along with any entered digits.
- A variable can be used in place of digits with all the possible special characters and digits can be entered ahead of the variable when needed for the application.
- If the vector variable or resultant destination is not defined or is invalid, the route-to step fails, a vector event 38 (variable not defined) is logged, and vector processing continues at the next vector step. The destination number 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 the vector event is logged, and vector processing continues at the next step.

See also:

- route-to command on page 290
- route-to command with VDN variables on page 142

set command with vector variables

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

See also:

- set command on page 305
- set command with VDN variables on page 143
- System-assigned vector variable types on page 104
- User-assigned vector variable types on page 112

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 VDN variables on page 137.

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 319
- wait command with VDN variables on page 144

VIV requirements

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

The **MultiVantage G3 Version** field on theSystem-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

Definition of local, global, and local persistent variables

Variable conditionals can be either local, global, or local persistent 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, local persistent, 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.

Local persistent scope

When a collect type variable's scope is "local persistent," (the P scope) its value is assigned on the basis of call-specific information (the same as "local" scope) and applies in one or more vectors that process the call. Unlike a local collect variable, the value assigned to a local persistent variable persists until a call disconnects. The application continues to use the variable and its most recently-assigned value when the call leaves vector processing but then returns to vector processing, such as via VDN return destination or when the call is transferred to a VDN for further processing by the answering agent.

For example, to count the VDN return destination loops, a local collect persistent variable can be defined with a length of 1 digit in the Variables for Vectors table. The value can be incremented every time the vector processing loops through the VDN return destination vector. The vector that the VDN Return goes to can test that value to determine how many loops the call has gone through.

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:

- Variable types that are local to the call or caller include ani, asaiuui, collect (when set to L), stepcnt, vdn, and vdntime.

  Note:

  ASAI data for a call can be modified by a CTI adjunct when a route-to adjunct command is used. For more information, see asaiuui type variable on page 105.

-

About local persistent variables

You should understand the following about local persistent variables:

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

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 referenced, and all calls that are executing those vector steps. When you change the value, the change is instantly propagated to all vector steps in which that variable is referenced. This rule applies to all global variable types that allows input in the**Assignment** field in the Variables for Vectors administration screen or other methods to assign a value. For more information, see Required variable administration entries on page 120.

Note:

Some variable types allow you to use the **`set`** command, the collect digits step, an FAC or the active VDN to change the specified value. When you use any of 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 system-obtained 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 120.

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.

Related topics:

System-assigned definition on page 105
ani type variable on page 105
asaiuui type variable on page 105
dow type variable on page 107
doy type variable on page 107
stepcnt type variable on page 108
tod type variable on page 109
vdn type variable on page 110
vdntime type variable on page 111

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 you know who called, you 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. Another way it can be changed or initially assigned is by using the **`set`** command. See set command on page 305 for details. 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 Manager. .

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 Manager..

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 Manager clock. The values assigned to this variable are obtained from the system clock on the Communication Manager. For information about setting and maintaining the system clock, see *Administering Avaya Aura™ Call Center Features* and the and the *Avaya Aura™ Communication Manager Feature Description*.

The Communication Manager always returns four digits for the tod variable. This includes leading zeros where appropriate. Any comparison to the tod variable is also formatted as four digits. 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 119.

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.

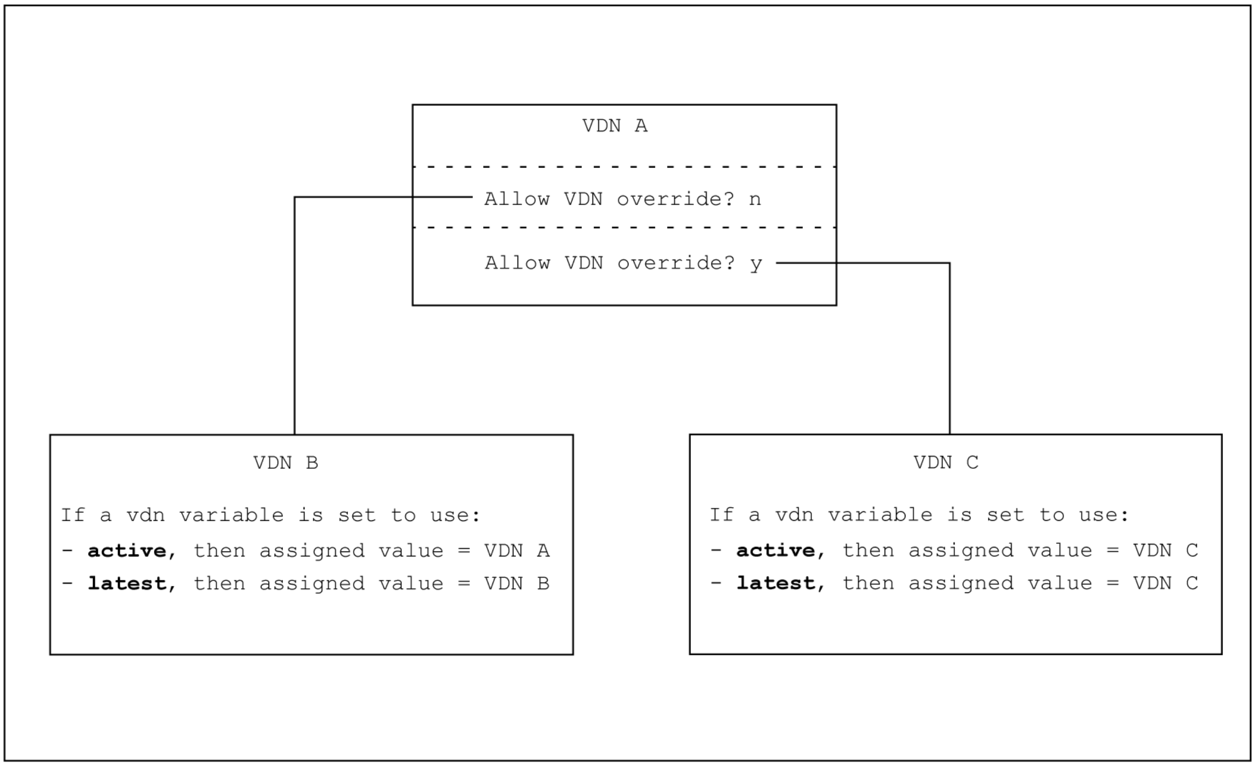

VDN A
Allow VDN override? n
Allow VDN override? y
VDN B
If a vdn variable is set to use:
- **active**, then assigned value = VDN A
- **latest**, then assigned value = VDN B
VDN C
If a vdn variable is set to use:
- **active**, then assigned value = VDN C
- **latest**, then assigned value = VDN C

Figure 5: Interactions between vdn variable assignments and VDN override settings

For more information about VDN Override settings, see VDN Override.

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.

Related topics:

User-assigned definition on page 113
collect type variable on page 113
value type variable on page 117

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 assigned from the system clock, data about the incoming call, or by the processing of the call.

collect type variable

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

Syntax

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

```
collect 2 digits for V
```

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

Note:

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

Other ways to assign a value to a collect type variable is by using the Variables for Vectors administration table (for the globe scope only) or by assignment using the **`set vector`** command. When used with the **`set`** command the collect type variable serves as a general purpose variable for implementing many different kinds of applications. See the set command on page 305 section for details. An example using the administration table to assign a value of 14 to variable V is shown in the following example excerpt from the table:

Variable	Description	Type	Scope (L, P, or G)	Length	Start	Assignment	VAC
V	Local collect variable	Collect	G	2	1	14	NA

Note:

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

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

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

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

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

For a complete description of the collect variable syntax used with the **collect** command, see collect digits command on page 234. For vector examples that show how the collect variable can be used, see Example 1 on page 115.

Scope

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

- If the scope is local, the assigned value is null until a value is provided during processing for the call. The assigned value is retained through all further call processing steps, including any chained vectors and route-to VDN commands, until a new value is assigned during vector processing or initial vector processing for the call is terminated, at which time the value is cleared.
- If the scope is local persistent, the assigned value is null until the value is provided during processing for the call. Unlike a variable with local scope, a collect type variable with local persistent scope persists until the call disconnects. The application can continue to use the variable when the call leaves vector processing and then returns to vector processing, such as via VDN return destination or a subsequent transfer to another VDN by the answering agent. Persistent local collect variables retain the last value assigned during vector processing, and after termination of vector processing, internal transfer to another local vector, RONA/ROIF/ROOF return to vector processing or VDN Return Destination return to vector processing, until the call disconnects.
- If the scope is global, the assigned value is retained as a system-wide variable value until it is reassigned, either by changes made to the Variable for Vectors screen, or by a collect digits/ced/cpd for [A-Z, AA-ZZ] vector step designed for that purpose. For more information about how to set up a VDN and vector to facilitate changing of Global Collect variable values, see the example at collect command with vector variables on page 93 and Example application using time and day variables on page 122.

Additional information about the collect variable

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

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

Example 1

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 122.

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

```
1. wait-time 0 secs hearing ringback
```

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

]
7. wait-time 60 secs hearing music
8. goto step 6 unconditionally
```

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

Example 2

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

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

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

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

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

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

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

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

value type variable

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 119.
- 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 Performing optional FAC administration for value variables on page 121.
- 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 type variables is only global.

Reason to use the value variable

One of the reasons to use the value variable is to change vector processing quickly through a manual operation. Before the value vector variable 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. Now the value vector variable type can be used as the trigger that can be tested in vectoring using a **`goto vector`** command. The trigger then can be set by dialing the Feature Access Code (FAC) assigned to the value type variable and entering in the value that will change the vector processing for the calls to that vector. For example, value variable V can be set to 0 for normal operation and then set to 1 to trigger the disaster operation using the FAC.

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 table.

- 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 set to yes on the Class of Service (COS) screen.
- To reset the assigned value for a value variable to null, access the FAC associated with variable and enter `*` instead of a digit.

Example

The following example shows how 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 126.

VIV interactions and considerations

VIV interactions/ Considerations	Description
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 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.
Variable failure conditions	When the variable conditional that is tested is not defined in the variables for vectors administration table, a **`goto`** test fails, the call does not branch, and processing falls through to the next vector step.
Retention of vector variable values and assignments	The content of a local vector variable exists only while a call is in vector processing. Once a call exits vector processing, the value is cleared. It is important to note that a call that

VIV interactions/ Considerations	Description
	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 for the same call. The value of a vector variable is not directly passed during an adjunct routing route request operation. The adjunct routing route request operation does pass the value of the "digits" buffer using the collected digits Information Element (IE). The vector **set** command can populate the "digits" buffer. 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.

Related topics:

Example Variables for Vectors screen on page 119
Required variable administration entries on page 120
Performing optional FAC administration for value variables on page 121

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 120.

```
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.

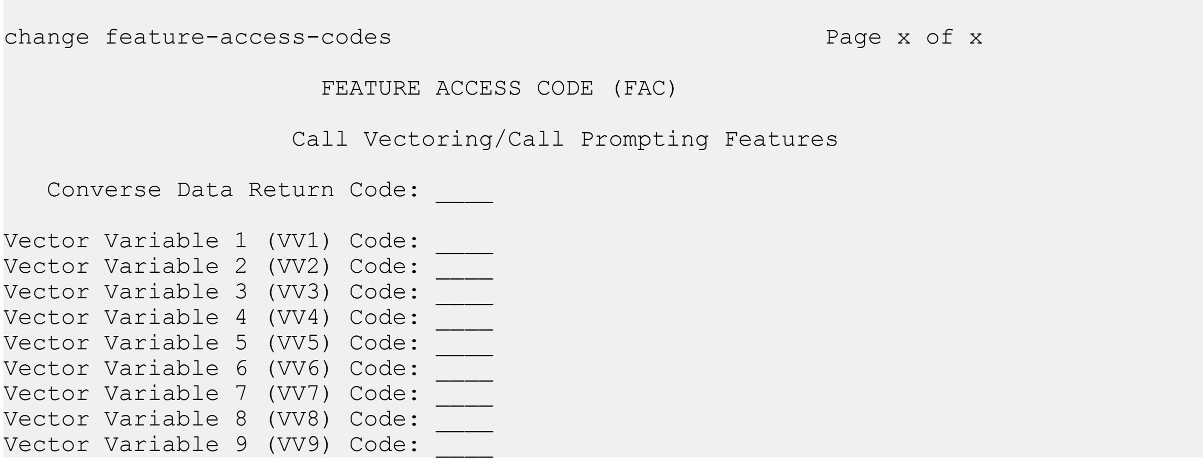
Variable type	Scope	Length	Start	Assignment	VAC (Variable Access Code)
ani	Local only (L)	1 to 16 digits (required)	start position from 1 to 16 (required)	Not applicable	Not applicable
asaiuui	Local only (L)	1 to 16 digits (required)	start position from 1 to 96 (required)	Not applicable	Not applicable
collect	Local, Local Persistent, or Global (L, P, or G, required)	1 to 16 digits (required)	start position from 1 to 16 (required)	Local and Local Persistent- not applicable Global - 1 to 16 digits	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

Variable type	Scope	Length	Start	Assignment	VAC (Variable Access Code)
value	Global only (G)	1	Not applicable	1 digit (0 to 9, optional)[6]	VV*x* (optional)[7]
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

Performing 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:

[6] *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 Performing optional FAC administration for value variables on page 121.*

[7] 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 Performing optional FAC administration for value variables on page 121. 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.

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 120.

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 104.

Related topics:

Example application using time and day variables on page 122
Example application using a value variable on page 126
Example applications using Global Collect variables on page 128
Example applications using vdn type variables on page 129
Example application using a vector variable in other commands on page 131
Example application using a vector variable in the converse-on command on page 131

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 Related topics.

Related topics:

Scenario details on page 123
Administering the variables on page 123
Creating a vector to use the time and day variables on page 124
Creating a vector to reassign call center hours of operation on page 125

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

Variable	Description	Type	Scope	Length	Start	Assignment
L[8]	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

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
```

[8] 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.

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 124, 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 related topics.

Related topics:

Administering a FAC code to use with a value variable on page 126
Administering the value variable on page 127
Using the value variable in multiple vectors on page 127

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 127.

```
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.

Related topics:

Verifying a password and then changing a value on page 128

Adding change confirmation on page 129

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

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

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

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

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

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

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

```
1. wait 0 secs hearing ringback
2. goto step 4 if Y = 4561
[VDN for 800 number callers]
3. announcement 2700  [
Our store is located at 1300 West 120th Avenue.
]
4. queue-to skill 30 pri l
5. wait-time 5 secs hearing ringback
6. announcement 1002
[
All agents are serving other customers, please wait..
]
7. wait-time 60 secs hearing music
8. goto step 6 if unconditionally
```

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

Example application using a vector variable in other commands

A vector variable can be used in the route-to number command to route a call to a destination provided indirectly 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 vector variable in the converse-on command

Including a vector 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 collect, value, tod, doy, dow, and asaiuui. You can use the asaiuui type to forward data provided by a remote site or local ASAI interfaced application or with assignment using the **`set`** command. For this example, variable D forwards numerical account code data of up to 6-digits provided by an ASAI application.

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 in the *Avaya Aura™ Call Center Feature Reference* document 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 353.

Chapter 7: VDN variables

Description of VDN variables

VDN variables provide more opportunities for VDNs to use less vectors by sharing over applications.

You can:

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

Related topics:
Reason to use VDN variables on page 135

Reason to use VDN variables

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
- converse-on command with VDN variables
- disconnect command with VDN variables
- goto commands with VDN variables
- route-to command with VDN variables
- set command with VDN variables
- wait command with VDN variables

Related topics:

announcement commands with VDN variables on page 137
converse-on command with VDN variables on page 138
disconnect command with VDN variables on page 140
goto commands with VDN variables on page 140
route-to command with VDN variables on page 142
set command with VDN variables on page 143
wait command with VDN variables on page 144

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
]
```

Related topics:

[Requirements and considerations for using VDN variables in announcements](#) on page 138

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 for Call Vectoring on page 219
- announcement commands with VDN variables on page 137

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
,

1st
,
2nd
,
3rd
]
pri
 [
l
,
m
,
```

[9] A valid hunt group is a vector-controlled ACD split or skill assigned on a Hunt Group screen.

```
h
,
t
]
passing
 [
data1
]

and
 [
data2
][10]
converse-on

split
 [
hunt group
][9]

pri
 [
l
,
m
,
h
,
t
]
passing
 [
V1-V9
]
and
 [
V1-V9
]
```

See also:

- converse-on command on page 244
- converse-on command with vector variables on page 94

[10] You can use a VDN variable only in data1, only in data2, or in both.

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 137.

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

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

See also:

- disconnect command for Call Vectoring on page 256
- disconnect command with vector variables on page 96

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 @ step1-99 if**						
A-Z, AA-ZZ	>,<, =,<>, >=, <=	V1-V9				
	in table	V1-V9				
	not-in table					
ani	>, >=, <>, =, <, <=	V1-V9				
	in table	V1-V9				
	not-in table					
available-agents	in skill	hunt group,skills for VDN: 1st, 2nd, 3rd	<, >=, <>, =, <, <=	V1-V9		
calls-queued	in skill	hunt group, skills for VDN:1st, 2nd, 3rd	pri	priorities:l = low, m = medium, h	>, >=, <>, =, <, <=	V1-V9

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

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

See also:

- goto step and goto vector commands on page 259
- goto commands with vector variables on page 96

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*][11]	with cov	y, n (y=yes, n=no)	if	digit	>, >=, <>, =<, <=	0-9, #
						interflow-qpos	< **`<, =, <=`**	1-9
						unconditionally		

[11] When the specified number is preceeded by ~r, Network Call Redirection is attempted.

Related topics:

[Requirements and considerations for route-to command with VDN variables](#) on page 143

Requirements and considerations for route-to command with VDN variables

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 290
- route-to command with VDN variables on page 142

set command with VDN variables

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

```
set [
variables[12]
, digits] = [
operand1
] [
operator
] [
operand2
]
```

You can use VDN variables in the following fields:

- **Operand1**
- **Operand2**

[12] *A user-assignable vector variable only.*

See also:

- set command on page 305.
- set command with vector variables.

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 137.

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 319
- wait command with vector variables on page 101

Case studies

Using one vector for different announcements

In this case study, agents working for the Alpha service bureau handle calls for three different companies - ABC 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

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
V3	route-to destination 1	2001	3001	4001
	route-to destination 2	2002	3002	4002
	route-to destination 3	2003	3003	4003

VDN variable	Parameter	VDN 1	VDN 2	VDN 3
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.

Chapter 8: Vector subroutines

Overview of vector subroutines

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

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

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

The maximum simultaneous active subroutine calls allowed are:

- 8000 for S8500 and S8700 platforms
- 400 for S8300 platforms

Related topics:

Reason to use vector subroutines on page 151

Reason to use vector subroutines

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 **`gotovector`** 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.

Related topics:

Incoming call processing vector example on page 153
Checking working hours subroutine vector on page 153

Incoming call processing vector example

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.
]
```

Chapter 9: ANI /II-digits routing and Caller Information Forwarding (CINFO)

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.

Related topics:

CINFO command sets on page 155
ANI routing on page 156
II-digits routing on page 159
Caller Information Forwarding on page 165

CINFO 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).	`goto vector`

Command category	Action taken	Command
	Go to another vector based on ced or cdpd. (CINFO digits).	
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.

Related topics:

ANI basics on page 156

Use of ANI with vector routing tables example on page 158

Use of ANI with vector routing tables on page 158

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 Use of ANI with vector routing tables on page 158.

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.

Use of ANI with vector routing tables 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.

Use of 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

Related topics:

II-digits basics on page 159
II-digits codes on page 160
II-digits routing example on page 165

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

Table 8: 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

II-digits	Description
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
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

II-digits	Description
	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.
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

II-digits	Description
	accompanying digit pair 61 identifies only the originating cellular/wireless PCS system, not the mobile directory placing the call.
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
- CINFO vector example on page 167
- CINFO interactions on page 168

CINFO basics

UEC IE storage

When an ISDN call is received from either the AT&T network or a tandemed PRI call, the Communication Manager 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 Manager does not store the UEC IE. When the collect digits step is reached, the collected digits buffer is still cleared and if a TTR is attached, it is still disconnected. A vector event is generated to indicate that no digits were collected.

If no ced or cdpd are received from the network when a collect digits step is processed, the step is not processed. However, the collected digits buffer is still cleared and if a TTR is attached, it is still disconnected.

Use of wildcards

If an asterisk (*) is included in the collected digits, it is treated as a delete character. Only the digits to the right of the asterisk are collected. If a pound sign (#) is included in the collected digits it is treated as a terminating character. Only the pound sign and the digits to the left of it are collected. If a single pound sign is sent, it is placed in the collected digits buffer.

String length

The number of ced or cdpd to collect cannot be specified in the collect digits step. Although ced and cdpb can each contain as much as 30 digits, only 16 digits can be collected and stored. If there are more than 16 digits, a vector event is generated.

Vector commands that use ced and cdpd

The following vector steps can access CINFO ced and cdpd in the collected digits buffer:

- adjunct routing link (digits passed in an event report as collected digits)
- converse-on...passing digits
- goto...if digits...
- goto...if digits in table...

- route-to digits
- route-to number ... if digit...

Tip:

You can use the CALLR INFO button on the 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 Manager 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 Manager and then made available to **`collect digits`** vector steps. ced and cdpd are both used to determine routing for the call.

```
 1. wait-time 2 secs hearing silence
 2. collect ced digits
 3. goto step 7 if digits = 1
 4. goto step 11 if digits = 2
 5. route-to number 0 with cov n if unconditionally
 6. stop
 7. collect cdpd digits
 8. route-to digits with coverage n
 9. route-to number 0 with cov n if unconditionally
10. stop
11. queue-to split 6 pri m
12. wait-time 10 secs hearing ringback
13. announcement 2564
14. wait-time 20 secs hearing music
```

```
15. goto step 13 if unconditionally
16. route-to number 0 with cov n if unconditionally
```

In this vector, step 1 provides a wait-time step 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 Manager.

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 Manager.

Transfer

If a call is transferred from the Communication Manager, CINFO digits are lost. If a call is transferred to an internal extension, CINFO digits are retained.

Important:

If a call is transferred to a VDN, the CINFO digits 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.

Chapter 10: Multi-national Calling Party Number (CPN) prefixes

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 Communication Manager, 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.

Screens and fields used to administer multi-national CPN prefix

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:

1. Public Unknown Numbering Plan
2. Route Pattern
3. Tandem Calling Party Number Conversion
4. Trunk Group
5. Uniform Dial Plan
6. Vector and
7. Vector Routing Table

Result

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.

Chapter 11: Creating and editing call vectors

Methods for entering a vector online

A vector can be entered using several methods:

- Basic screen administration on the system administration terminal or Avaya Site Administration
- Avaya Call Management System (CMS) using the ASCII administration interface
- Avaya Visual Vectors

The following section discusses the basic screen administration method for entering a vector online at the system administration terminal or Avaya Site Administration. For instructions on creating a vector with Visual Vector, see Avaya Visual Vectors User Guide, see Avaya CMS Administrationvector with Visual Vectors, see Avaya Visual Vectors User Guide. There are no instructions available for using the Avaya CMS ASCII administration interface.

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 83 for details on using these commands.
EAS	Expert Agent Selection is enabled. See Expert Agent Selection for information on how the EAS feature works.
G3V4 Enhanced	You can use the G3V4 Enhanced Vector Routing commands and features. See Feature availability in the *Planning an Avaya Aura™ Call Center Implementation* document 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 155 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 for details on using this command.
Prompting	You can use the Call Prompting commands. See Call Prompting for details on using these commands.
LAI	Look-Ahead Interflow is enabled. See Look-Ahead Interflow (LAI) information on how LAI works.
G3V4 Adv Route	You can use the G3V4 Advanced Vector Routing commands. See Advanced Vector Routing - EWT and ASA 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 155 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) 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 in the *Avaya Aura™ Call Center Feature Reference* document 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 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`.

Result

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

Viewing vector variable information

To display vector variable information from the Variables in Vectors table:

While using the **`change vector`** command, press `Esc f 6`.

Result

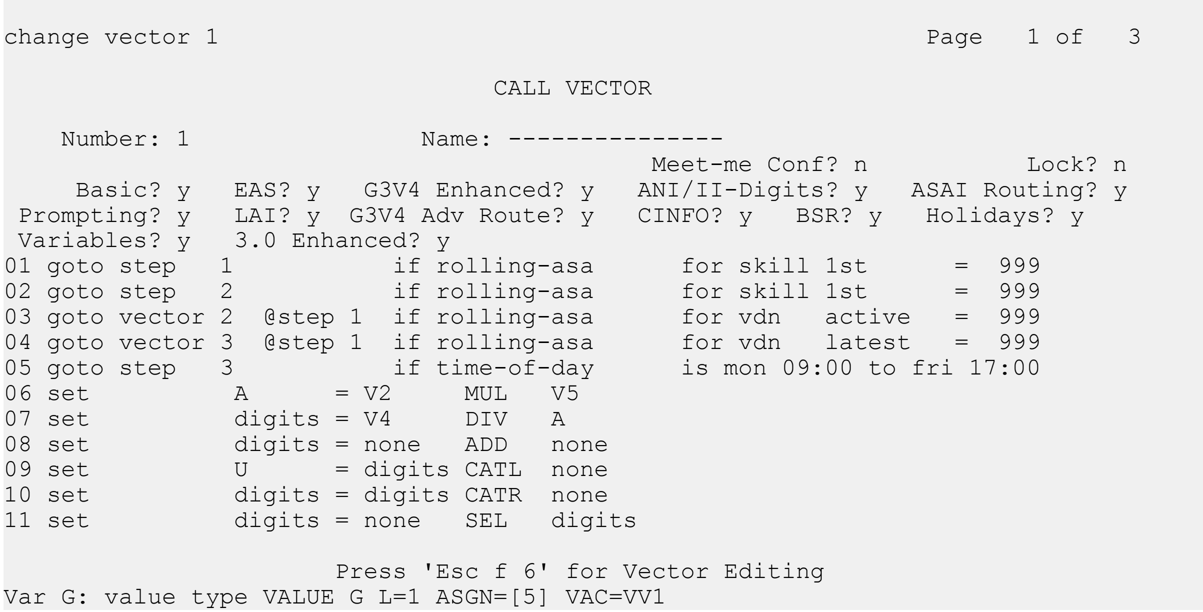
After the Edit (i/d/v/c/u) prompt, enter a “v”, plus the variable you want to view.

Enter an A-Z or AA-ZZ value.

Example: **v G**

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
```

Variable display fields

The following line is displayed on the bottom of the Call Vector screen after you perform Viewing vector variable information on page 178.

Var *letter*: *description type scope [***L=***length* **S=***start* **ASGN=***current_value* **VAC=***fac]*

Name	Description
The following four fields are always displayed.	
Var *letter*	Displays the A through Z vector variable letter you requested.
description	Displays the name of the variable. For example, `value type`.
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	

Name	Description
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.

Variable display examples

Refer to the values assigned in Table A on page 180 when looking at the example outputs in Table B on page 181.

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		

Table A							
Variable	Description	Type	Scope	Length	Start	Assignment	VAC
Y	day of year	doy	G			102	
Z	temporary value	collect	L	4	1		

Table B	
For	Based on the values in Table A, the following text is displayed
`Edit (i/d/v/c/u): v A`	Var A: testing for processing time VDNTIME L
`Edit (i/d/v/c/u): v B`	Var B: digits for ani testing COLLECT G L=16 S=1 ASGN=[1234567890123456]
`Edit (i/d/v/c/u): v C`	Var C: ASAI announce Definition ASAIUUI L L=1 S=3
`Edit (i/d/v/c/u): v D`	Var D: test with null value COLLECT G L=1 S=4 ASGN=[]
`Edit (i/d/v/c/u): v E`	Var E: total executed vector steps STEPCNT L
`Edit (i/d/v/c/u): v G`	Var G: value type VALUE G L=1 ASGN=[5] VAC=VV1
`Edit (i/d/v/c/u): v T`	Var T: time of day, military time TOD G ASGN=[1708]
`Edit (i/d/v/c/u): v V`	Var V: set to active VDN for call VDN L ASGN=ACTIVE
`Edit (i/d/v/c/u): v W`	Var W: day of week, 1=Sunday DOW G L= S= ASGN=[3]
`Edit (i/d/v/c/u): v X`	Var X: caller ID ANI L L=16 S=1
`Edit (i/d/v/c/u): v Y`	Var Y: day of year DOY G ASGN=[102]
`Edit (i/d/v/c/u): v Z`	Var Z: temporary value COLLECT L L=4 S=1

Inserting 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.

Result

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`.

Result

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 183.

Blank steps and comment out steps cannot be commented out. The comment out capability has no limitations other than the number of vector steps. That is, you can comment out 99 steps in all 8000 vectors or 198,000 steps.

To comment out an existing step:

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`.

Result

Note:

The comment out indication functions differently than the **#** command. See # command on page 210 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 processing 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.

Result

Note:

The comment out indication functions differently than the **#** command. See # command on page 210 for details.

Creating and constructing a vector

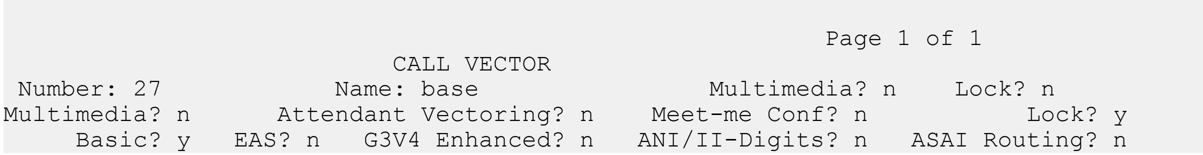
About creating and constructing a vector

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 _______________
```

Related topics:

Agent Availability on page 185
Call Priority levels on page 185

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 _______________
```

Related topics:

Using the wait-time command on page 186

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 319.

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 Avaya Aura™ Communication Manager Feature Description and Implementation, 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.

Related topics:

Conditional branching on page 187

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.

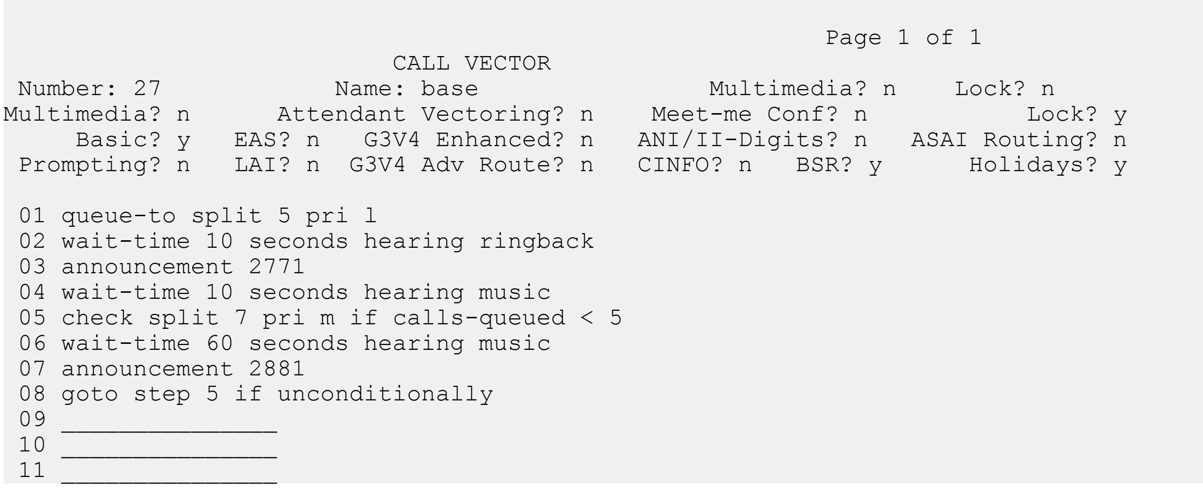
The **goto step** command in vector step 5 allows you to establish an announcement-wait loop that continues until the agent answers the call. Specifically, the command makes an unconditional branch to the **announcement** command in step 3. If the call is not answered by the time that the announcement 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.

Related topics:

Conditions used with the check split command on page 189

Elevating call priority on page 189

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 Feature 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.

Related topics:

Redirecting calls to a backup split on page 190

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 83 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.

Related topics:

Duplicate Vector command on page 191
Duplicate Vector screen field descriptions on page 192
Creating duplicate vectors on page 193

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]
```

`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.

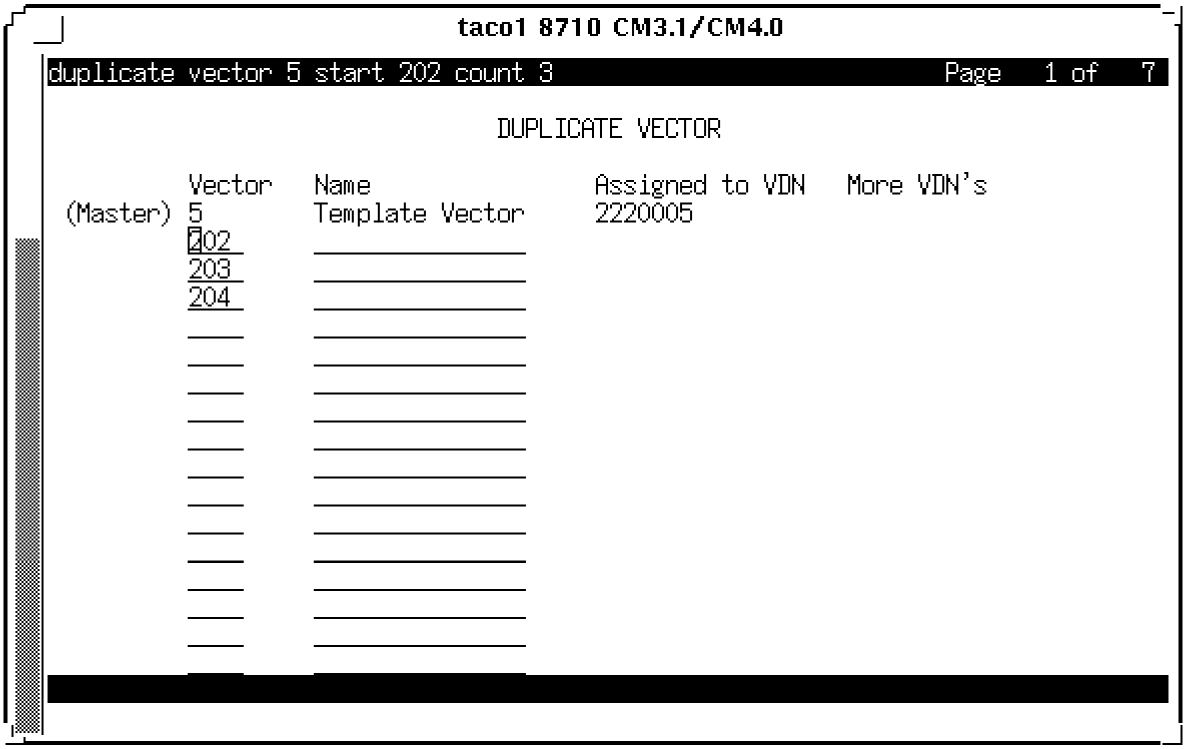
[start nnnn] Specifies the first vector number you want used as a duplicate. This parameter is optional. If you do not specify a start number, the software automatically selects the first available vector after the master vector number. Only one vector is selected.

[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
```



```
                         taco1 8710 CM3.1/CM4.0
duplicate vector 5 start 202 count 3                      Page   1 of   7

                              DUPLICATE VECTOR

          Vector   Name                  Assigned to VDN   More VDN's
 (Master) 5        Template Vector       2220005
          202
          203
          204
```

Duplicate Vector screen field descriptions

The following fields are populated in the Duplicate Vector screen:

Field	Description
Count	Displays the number of duplicates created from the master vector.

Field	Description
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.

Result

The reporting adjunct can access these vectors as normal.

Chapter 12: Vector management and monitoring

Implementation requirements for the Call Vectoring features

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 *Avaya Aura™ Communication Manager Hardware Description and Reference*.

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 *Avaya Aura™ Communication Manager Hardware Description and Reference*.

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	

Screen(s)	Hardware
• 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.

Related topics:

ANI/II-Digits requirements on page 197

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	Requires no hardware in addition to that required for Basic Call Vectoring.

Screens	Hardware
• Trunk Group screens • Vector Routing Tables screens	

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 *Administering Avaya Aura™ Call Center Features*.

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 Considerations for the vectoring features in the *Avaya Aura™ Call Center Feature Reference* document.
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.

Chapter 13: Call Vectoring commands

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 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 Communication Manager 2.0. An Introductory package was also available that was identical to Deluxe without Best Service Routing but with BCMS. The Introductory package is now included with the Communication Manager basic feature offerings along with Call Center Basic.

Most of the features required to fully enable vector commands are included in the basic Communication Manager package. 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. (“Basic” refers to the Call Vectoring (Basic) option)

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
`check split/skill if <condition>`	x			ACD
`check split/skill if rolling-asa`	x			ACD; G3V4 Enhanced; G3V4 Advanced Routing
`check split/skill if expected-wait`	x			ACD; G3V4 Enhanced; G3V4 Advanced Routing
`check best if expected-wait`	x			ACD; G3V4 Enhanced; G3V4 Advanced Routing; BSR
`check split/skill if oldest-call-wait pri`	x			ACD; G3V4 Enhanced
`check split/skill/ best if wait-improved`	x			ACD; G3V4 Advanced Routing; Best Service Routing
`check skill if available-agents all-levels`	x			EAS active
`check skill if available-agents pref-level`	x			EAS active
`check skill if available-agents pref-range`	x			EAS active
`collect digits`		x		
`collect ced/cdpd digits`		x		Vectoring (CINFO)
`consider location`	x			ACD; G3V4 Advanced Routing; Best Service

Command	Basic	Prompting	Attendant	Other Options Required
				Routing; Look-Ahead Interflow[13]
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			
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

[13] Provided with Virtual Routing RTU (right to use).

Command	Basic	Prompting	Attendant	Other Options Required
`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[13]
`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`[14]	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
`reply-best`	x			ACD; G3V4 Advanced Routing; Best Service Routing; Look-Ahead Interflow[13]
`return`	x			Vectoring (3.0 Enhanced)
`route-to number`	x			
`route-to digits with cov y (n)`		x		
`route-to number if digit`		x		

[14] If G3V4 software has not been purchased, these commands require the G3V4 maintenance load.

Command	Basic	Prompting	Attendant	Other Options Required
`route-to number if unconditionally with cov y (n)`[14]	x	x		
`route-to number if digit with cov y (n)`[14]		x		
`route-to number if unconditionally`	x	x		
`route-to number if interflow-qpos`	x			ACD, Look-Ahead Interflow[13]
`set`	x			Vectoring (3.0 Enhanced)
`stop`	x	x		
`wait-time <time>`	x	x	x	
`wait-time <time> hearing <treatment>`	x	x	x	
`wait-time <time> hearing <extn> then <treatment2>`	x	x	x	

Vector command description

The following table provides a brief description of 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 210	Adds comments to vectors.
adjunct routing link command	Requests an adjunct to route a call.
announcement command for Call Vectoring on page 219	Connects a caller to delay recording.
busy command for Call Vectoring on page 226	Connects a caller to busy tone.
check command on page 228	Connects or queues a call on a conditional basis.
collect digits command on page 234	Prompts a caller for digits.

Command	Function
consider command on page 239	Obtains BSR status data from a local split/skill or a remote location
converse-on command on page 244	Delivers a call to a converse split/skill and activates a Voice Response Unit (VRU).
disconnect command for Call Vectoring on page 256	Forces the disconnect of a call with an optional announcement.
goto step and goto vector commands on page 259	Causes an unconditional or a conditional branch to another step in the vector.
messaging command on page 275	Allows a caller to leave a message for callback.
queue-to command on page 280	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 287	Sends BSR status data to the primary vector in a multi-site application.
return command on page 289	Returns vector processing to the step following the `goto` command after a subroutine call has processed.
route-to command on page 290	Connects a call to the destination entered using `collect digits` command, or connects a call to internal or external destination.
set command on page 305	Performs arithmetic and string operations and assigns values to a vector variable or to the Digits buffer during vector processing.
stop command on page 306	Stops further vector processing.
wait-time command on page 319	Initiates feedback to a caller if needed and delays processing of the next step.

# command

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 183 and Removing a comment out indication on page 183 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 8,000 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  13
                                   SYSTEM CAPACITY
                                                                      System
                                                      Used Available  Limit
                                                     -----------------------
                    CALL COVERAGE
                            Coverage Answer Groups:      0    1000      1000
                                    Coverage Paths:      5    9994      9999
                                Call Pickup Groups:      1    4999      5000
                                      Call Records:      -       -     15424

           CALL VECTORING/CALL PROMPTING
                    Total Vector Directory Numbers:     81   19919     20000
```

```
              Meet-me Conference VDNs per system:      1    1799     1800
   Maximum Number of Expanded Meet-me Conf. Ports:      0     300      300
                         Total Vectors Per System:     90    1910     2000
              Meet-me Conference vectors per system:    1     999      999
        BSR Application-Location Pairs Per System:      8    2552     2560
                         Background BSR Poll VDNs:      0       5        5
                     Vector Comment Steps (non-blank):  77    9923    10000
                             Policy Routing Tables:     1    1999     2000
                             Policy Routing Points:     1    5999     6000
```

Operation

You 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

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[15] [A-Z, AA-ZZ] V1-V9

For information about unexpected results, see Troubleshooting vectors.

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.

[15] Link capacity depends on your release and configuration. For more information, see System Capacities Table for Communication Manager on Avaya Media Servers, , 555-245-601.

Related topics:

The adjunct routing link process on page 213
adjunct routing link command feature interactions on page 216
adjunct routing link command interactions with Avaya IQ on page 217
adjunct routing link command interactions with CMS on page 217
adjunct routing link command interactions with BCMS on page 219

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 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.
- 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.

adjunct routing link command 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 in the *Avaya Aura™ Call Center Feature Reference* document.

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.

adjunct routing link command interactions with Avaya IQ

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.

adjunct routing link command interactions with CMS

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	
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.

adjunct routing link command interactions with BCMS

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 for Call Vectoring

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.

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 Avaya Aura™ Communication Manager Feature Description and Implementation.

Related topics:

Basic operation for the announcement command on page 220
General considerations for announcements on page 220
Delay announcements on page 221
Forced announcements on page 222
Information announcements on page 222
Announcement recording tips for high traffic volume applications on page 222
Recording announcements on page 223
Considerations for DTMF Transfer and Connect applications on page 224

Basic operation for the announcement command

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 Avaya Aura™ Communication Manager Feature Description and Implementation.

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 319.

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 *Administering Avaya Aura™ Call Center Features*.
- 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 224.

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 Administering Avaya Aura™ 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 Administering Avaya Aura™ 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).

Related topics:

annoucement command feature interactions on page 225
announcement command interactions with Avaya IQ on page 225
announcement command interactions with CMS/BCMS on page 225

annoucement command 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.

announcement command interactions with Avaya IQ

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.

announcement command interactions with CMS/BCMS

The command is not tracked by CMS or BCMS.

busy command for Call Vectoring

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.

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.

Related topics:

Answer supervision considerations for the busy command on page 227
busy command feature interactions on page 227
busy command interactions with Avaya IQ on page 227
busy command interactions with CMS on page 227
busy command interactions with BCMS on page 227

Answer supervision considerations for the busy command

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

busy command feature interactions

For LAI or BSR, the command is considered a call denial vector command in all cases.

busy command interactions with Avaya IQ

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.

busy command interactions with CMS

Busy command	
Database Item	**Report Heading**
BUSYCALLS/BUSYTIME	Calls Forced Busy Calls Busy/Disc
OTHERCALLS/OTHERTIME	Inbound Other Calls
INTIME	Avg Time In Vector

BUSYTIME, OTHERTIME, and INTIME for splits and vectors are tracked according to when the busy tone starts. BUSYTIME, OTHERTIME and INTIME for VDNs are tracked according to when the trunk idles.

busy command interactions with BCMS

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

check command

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[16], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities:**l** = low **m** = medium**h** = high **t** = top	**if**	**available-agents** > 0-1499[17]	**all-levels**
							pref-level skill level 1[18]
							pref-range skill level 1[18] **to** skill level 2[18]
	skill	hunt group[16], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities:**l** = low **m** = medium**h** = high **t** = top	**if**	**calls-queued** < 1-999[17] **expected-wait** < 1-9999 seconds **oldest-call-wait** > 1-999 seconds **rolling-asa** < 1-999 seconds **staffed-agents** > 0-1499[17] **wait-improved** > 0-9999 seconds **unconditionally**	
	split	hunt group[16]					
	split	hunt group[16]	**pri**	**l** = low **m** = medium**h** = high**t** = top	**if**	**available-agents** > 0-1499[17]	

For information about unexpected results, see Troubleshooting vectors.

[16] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[17] 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.

[18] Skill levels are 1-16 (1 is best, 16 is lowest). Skill Level 2 must be greater than or equal to Skill Level 1.

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.

Related topics:

Check split command on page 230
Check skill for available agents with level preference on page 231
Answer supervision considerations for the check command on page 232
check command feature interactions on page 232
check command interactions with Avaya IQ on page 233
check command interactions with CMS on page 233
check command interactions with BCMS on page 234

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 283.

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:

<table>
<tr><td rowspan="3"><b>check</b></td><td rowspan="3"><b>skill</b></td><td rowspan="3">hunt group, 1st, 2nd, 3rd</td><td rowspan="3">pri</td><td rowspan="3">l = low
m = med
h = high
t = top</td><td rowspan="3">if</td><td rowspan="3">available-agents > 0-1499</td><td colspan="4">all-levels</td></tr>
<tr><td>pref-level</td><td colspan="3">skill level 1</td></tr>
<tr><td>pref-range</td><td>skill level 1</td><td>to</td><td>skill level 2</td></tr>
</table>

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 for the check command

No answer supervision is returned.

check command 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).

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).

check command interactions with Avaya IQ

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.

check command interactions with CMS

Calls answered using the check command are indicated as answered by backup in CMS.

Calls queued using a **`check split/skill`** command are tracked as CALLSOFFERRED and LOWCALLS/MEDCALLS/HIGHCALLS/TOPCALLS.

The presence of the command in a vector enables the calls serviced by the vector to be vector-directed. When such a call is answered by an agent, the call is tracked as ACDCALLS/ ANSTIME, and it is reported as ACD Calls, Split/Skill ACD Calls, and Avg Speed Ans. If the call is also queued to other splits/skills, OUTFLOWCALLS/OUTFLOWTIME is tracked in the first split/skill to which the call queues, and Flow Out is reported (unless the split/skill turns out to be the answering split/skill). DEQUECALLS/DEQUETIME is tracked in the second and third splits/skills if these splits/skills are not the answering split/skill, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time. However, if the second or third split/skill is the answering split/skill, INFLOWCALLS is tracked in the split/skill, and the call is reported as Flow In.

Whenever the call is answered in a split/skill accessed by the **`check split/skill`** command, the BACKUPCALLS data base item is incremented, and the call is reported as Calls Ans in Backup and Calls Handled/Backup. The Calls Ans in Main report item is calculated by using the algorithm ACDCALLS - BACKUPCALLS.

If the call abandons after the command queues the call to a split/skill, ABNCALLS/ABNTIME is tracked for the vector, the VDN, and the first split/skill to which the call is queued. The call is reported as Aban Call and Avg Aban Time. If the call is also queued to other splits/skills, DEQUECALLS/DEQUETIME is tracked in these splits/skills, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time.

BSR status poll calls are not counted as interflows. BSR interflows are now tracked as network interflowed calls (NETCALLS) by the CMS at the receiving switch. The CMS tracks a call's accumulated time-in-VDN as NETINTIME (that is, the NET_TIME value on the CMS at switch C combines the time a call has spent in VDNs at any previous locations, as communicated by information forwarding. The NETINTIME can be added to the time spent in the local switch to provide reports that include the total time the call has spent in the call center network (e.g., total ASA).

For more information on CMS database items and reports, see Avaya CMS Database Items and Calculations and Avaya CMS Supervisor Reports.

check command interactions with BCMS

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

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	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`

For information about unexpected results, see Troubleshooting vectors.

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.

Related topics:

Answer supervision considerations with the collect digits command on page 238
collection digits command feature interactions on page 239
collection digits command interactions with Avaya IQ on page 239
collection digits command interactions with CMS/BCMS on page 239

Answer supervision considerations with the collect digits command

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.

collection digits command feature interactions

For BSR and LAI, the command is considered a call acceptance vector command except for collect ced/cdpd digits which is neutral.

collection digits command interactions with Avaya IQ

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.

collection digits command interactions with CMS/BCMS

Collected digits are passed to the CMS when the **`collect`** step is processed. Digits are not passed to the BCMS.

consider command

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

consider	**location**[19](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 [20], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op		
	split	hunt group[2]				

For information about unexpected results, see Troubleshooting vectors.

Requirements

For switch requirements, see Server requirements in the *Avaya Aura™ Call Center Feature Reference* document.

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 in the *Avaya Aura™ Call Center Feature Reference* document.

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

[19] This item is available only with the Virtual Routing feature.

series) in order to reduce real-time delays for call processing and reduce the incidence of race conditions in multi-site BSR applications.

Related topics:

User adjustments on page 241
Events that clear best data on page 242
Recommendations for the consider command on page 243
Answer supervision considerations for the consider command on page 243
consider command feature interactions on page 243
consider command interactions with Avaya IQ on page 243
consider command interactions with CMS/BCMS on page 244

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.

[20] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

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 on EWT, see Expected Wait Time (EWT) in *Avaya Aura™ Call Center Feature Reference*.

As the steps in a consider series execute, the status data for the best resource found is kept in a buffer. This best data is unaffected by some call processing events and vector commands, while other events and commands initialize (clear) this buffer. The following table shows you what initializes the best data buffer and what does not.

Initialization of BSR best data	
Events and vector commands that clear best data	**Events and vector commands that do not clear best data**
Execution of any queue-to or check command	**`Converse`** command
Vector processing terminates: • **`reply-best`** command executes • agent answers • successful **`route-to`** command • successful **`adjunct routing link`** command • successful **`messaging split/skill`** command • vector disconnect timeout • **`disconnect`** command • **`busy`** command • vector processing reaches last step without call in queue	**`Announcement`** command
	`Collect Digits` command
	Unsuccessful execution of a **`messaging split/skill`** command
	Unsuccessful **`adjunct routing link`** command
	`Goto step/vector` with any conditional
	`Wait` command (with any feedback)
	Unsuccessful **`route-to`** command
	Vector processing reaches last step while call is still in queue
	Execution of a **`consider`** step (this will either replace the current best data with new data or leave the current data untouched)

Recommendations for the consider command

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 for the consider command

All forms of the **`consider`** command are ISDN neutral and do not return answer supervision.

consider command feature interactions

Splits used in consider commands must be vector-controlled.

consider command interactions with Avaya IQ

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.

consider command interactions with CMS/BCMS

BCMS does not log LAI attempts. Therefore, it will not log BSR status polls since they are LAI attempts.

converse-on command

Syntax and valid entries for the converse-on command

converse-on	**skill**	hunt group[21], 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[21]				**none**	**and**	**none**

For information about unexpected results, see Troubleshooting vectors.

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.

[21] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

For details regarding call flows, data passing, collection, and return specifications involving the **`converse-on`** command, see Call flow and specifications for converse - VRI calls in the *Avaya Aura™ Call Center Feature Reference* document.

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 Manager can be set up to pass information in-band to the VRU. In such a case, the **`converse-on`** command can outpulse up to two groups of digits to the VRU. The digits 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 Manager. In many applications, both application selection and data sharing are required. The touch tone outpulsing rate is adjustable. For details, see Call flow and specifications for converse - VRI calls in the *Avaya Aura™ Call Center Feature Reference* document.

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.

Related topics:

Data 1 and Data 2 values administered within the converse-on command on page 246
converse-on split command on page 248
Answer supervision considerations for the converse-on split command on page 251
converse-on split command feature interactions on page 251
converse-on split command interactions with Avaya IQ on page 256
converse-on split command interactions with CMS on page 256
converse-on split command interactions with BCMS on page 256

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) in the *Avaya Aura™ Call Center Feature Reference* document. 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_1is 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.

Related topics:

VRI capabilities on page 248
VRI benefits on page 249
Other VRI considerations on page 249
Using converse-on to outpulse caller information to VRUs on page 249

VRI capabilities

VRI can do the following:

- Execute a VRU script while retaining control of the call in vector processing.

  Note:

  If an agent becomes available to service the call, the line to the VRU is immediately dropped, and the calling party is connected to the available agent.

- Execute a VRU script while the call retains its position in the queue.
- Group VRU ports for multiple applications.
- Use a VRU as a flexible external announcement device.
- Pass data between the switch and a VRU.
- Tandem VRU data through the switch to an Adjunct Switch Application Interface (ASAI) host.

The capabilities listed above are provided by the **converse-on split** command, which is an enhancement to the Basic Call Vectoring customer option. The converse-on split step is integrates a VRU with the Communication Manager.

VRI benefits

Use of VRUs with vector processing 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) in the *Avaya Aura™ Call Center Feature Reference* document.

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.

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 Call flow and specifications for converse - VRI calls in the *Avaya Aura™ Call Center Feature Reference* document.

Answer supervision considerations for the converse-on split command

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.

converse-on split command feature interactions

Interaction	Description
Abandon Call Search	If the converse-on step places a call to a hunt group, and if the incoming call 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.

Interaction	Description
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

Interaction	Description
	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.

Interaction	Description
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

Interaction	Description
	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.

converse-on split command interactions with Avaya IQ

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.

converse-on split command interactions with CMS

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.

converse-on split command interactions with BCMS

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 for Call Vectoring

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.

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.

Related topics:

- Answer supervision considerations for disconnect command on page 258
- disconnect command feature interactions on page 258
- disconnect command interactions with Avaya IQ on page 258
- disconnect command interactions with CMS on page 258
- disconnect command interactions with BCMS on page 259

Answer supervision considerations for disconnect command

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

disconnect command feature interactions

For LAI, the 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.

disconnect command interactions with Avaya IQ

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.

disconnect command interactions with CMS

DISCTIME, OTHERTIME, and INTIME for splits and vectors are tracked according to when the announcement starts. DISCTIME, OTHERTIME and INTIME for VDNs are tracked according to when the trunk idles.

disconnect command	
Database Item	Report Heading
DISCCALLS/DISCTIME	Calls Forced Disc
	Calls Busy/Disc
OTHERCALLS/OTHERTIME	Inbound Other Calls
INTIME	Avg Time In Vector

disconnect command interactions with BCMS

A call that is disconnected using the command is tracked as OTHER in the VDN Report.

goto step and goto vector commands

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 1-8000 [22] @ step1-99 if		
A-Z, AA-ZZ	>,<,=,<>,>=,<=	threshold value or string of digits: 1-16, wildcards (? , +)[23], [A-Z, AA-ZZ], V1-V9
	=,<>	none[24], #[25]

[22] 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.

[23] 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.

[24] 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.

goto step and goto vector				
goto step 1-99 if or goto vector 1-8000 [22] @ step1-99 if				
	in table	1-100[22], [A-Z, AA-ZZ], V1-V9		
	not-in table			
ani	>,>=,<>,=,<,<=	1-16, wildcards (?, +)[24], [A-Z, AA-ZZ], V1-V9		
	=,<>	none[24], #[25]		
	in table	1-100[22], [A-Z, AA-ZZ], V1-V9		
	not-in table			
available-agents	in skill	hunt group[26], skills for VDN: 1st , 2nd , 3rd	>,>=,<>,=,<,<=	0-1499[22]1-1500[22]A-Z, AA-ZZ V1-V9
	in split	hunt group[26]		

goto step and goto vector						
goto step 1-99 if or goto vector 1-8000[27] @step 1-99 if						
calls-queued	in skill	hunt group[28], skills for VDN: 1st , 2nd , 3rd	pri	priorities: l = low m = medium h = high t = top	> >= <> = < <=	0-098[27]1-999 [27]A-Z, AA-ZZ V1-V9
	in split	hunt group[28]				
counted-calls	to vdn	vdn extension , latest , active[29]	>, >=, <>, =, <, <=		*0-998*[27] *1-999*[27] *A-Z, AA-ZZ V1-V9*	
digits	>, >=, <>, =, <, <=	threshold value or string: *1-16, wildcards* (*?, +*)[30] *, [A-Z, AA-ZZ], V1-V9*				
	<>, =	none[31]				

[25] 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.

[26] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[27] 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.

[28] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[29] Active refers to the VDN specified by VDN Override settings. Latest refers to the VDN specified for the current vector.

[30] 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.

[31] 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.

goto step and goto vector						
goto step 1-99 if or goto vector 1-8000[27] @step 1-99 if						
	=	meet-me-access[32]				
	in table	1-100[27], [A-Z, AA-ZZ], V1-V9				
	not-in table					
expected-wait	for best	>, >=, <>, =, <, <=	0-9999 seconds,[A-Z, AA-ZZ], V1-V9			
	for call					
	for split	hunt group[28]	pri	priorities: l = low m = medium h = high t = top	> >= <> = <<=	0-9998 sec 1-9999 sec A-Z, AA-ZZ V1-V9
	for skill	*hunt group*[28], skills for VDN: 1st , 2nd , 3rd				

goto step and goto vector			
goto step1-99 if or goto vector 1-8000[33] @step 1-99 if			
holiday	in table	1-999 , [A-Z, AA-ZZ], V1-V9	
	not-in table		
ii-digits	>, >=, <>, =, <, <=	2-digit string , wildcards (? , +)[34], [A-Z, AA-ZZ], V1-V9	
	<>, =	none[35]	
	in table	1-100[27], [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[36] *1-999*	= , <>	registered

[32] This item is available only with meet-me conference vectors.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

goto step and goto vector			
goto step1-99 if **or** **goto vector 1-8000[33] @step 1-99 if**			
	all		
	any		
meet-me-full[37] (goto step only)			
meet-me-idle[31] (goto step only)			
no match[38]			

goto step and goto vector						
goto step 1-99 if **or** **goto vector 1-8000[39] @step 1-99 if**						
oldest-call-wait	in skill	*hunt group*[40], skills for VDN: 1st , 2nd , 3rd	pri	priorities: l = low m = medium h = high t = top	> , >= , <> , =< , <=	*0-998* sec *1-999* sec *A-Z, AA-ZZ V1-V9*
	in split	hunt group[28]				
port-network	Port network ID[41] *1-999*		= , <>	registered		
	all					
	any					
queue-fail[42]						
rolling-asa	**for skill**	*hunt group*[28], skills for VDN: 1st , 2nd , 3rd		> , >= , <> , =, < , <=	*0-998 sec, 1-999 sec A-Z, AA-ZZ V1-V9*	

[37] This item is available only with meet-me conference vectors.

[38] This item is available only with the Dial by Name feature.

[39] 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.

[40] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[41] 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.

[42] This item is available only with the Attendant Vectoring feature.

goto step and goto vector **goto step 1-99 if** **or** **goto vector 1-8000[39] @step 1-99 if**				
	for split	hunt group[28]		
	for vdn	vdn extension, latest, active[43]		
server	=, <>	main, ess, lsp		
service-hours	in table	1-999, [A-Z, AA-ZZ], V1-V9		
	not-in table			
staffed-agents	**in skill**	*hunt group*[28], skills for VDN: 1st , 2nd , 3rd	>, >=, <>, =, <, <=	*0-1499*[33], *1-1500*[33] *A-Z, AA-ZZ V1-V9*
	in split	hunt group[34]		

goto step and goto vector **goto step*1-99*if** **or** **goto vector *1-8000*[44] @step *1-99* if**								
time-of-day	**is**	mon , tue , wed , thu , fri , sat , sun , all	hour: 00-23	minute: 00-59	**to**	mon , tue , wed , thu , fri , sat , sun , all	hour: 00-23	minute: 00-59
V1-V9	> , < , = ,	threshold value or string of digits: 1-16 , wildcards (? , +), [A-Z, AA-ZZ], V1-V9						

[43] *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

[44] 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.

goto step and goto vector						
goto step*1-99***if** **or** **goto vector** ***1-8000*** [44] **@step** ***1-99*** **if**						
	<> , >= , < , <=					
	= , <>	none[45] , # [46]				
	in table	1-100[36], [A-Z, AA-ZZ], V1-V9				
	not-in table					
wait-improved for	best	> , >= , <> , = , < , <=				*0-9998* sec *1-9999* sec *A-Z, AA-ZZ V1-V9*
	skill	*hunt group*[47], skills for VDN: 1st , 2nd , 3rd	**pri**	priorities: **l = low m = medium h = high t = top**	> , >= , <>, =< , <=	
	split	hunt group[5]				
unconditionally						

For information about unexpected results, see Troubleshooting vectors.

Requirements

For more information about options required to enable the **goto** commands, see Communication Manager options required to enable vector commands on page 205 .

Related topics:

goto step and goto vector commands operation on page 265

Time adjustments using goto conditionals on page 268

[45] 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.

[46] 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.

[47] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

goto step and goto vector commands operation

Basic operation

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

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

- The average speed of answer for the indicated split/skill or VDN meets the constraints defined by the comparator and threshold value.
- The number of available agents in the indicated split/skill meets the constraints defined by the comparator and the threshold value.
- The number of queued calls in the indicated split/skill and at the specified priority level (or higher) meets the constraints defined by the comparator and the threshold value.
- The number of active calls in the indicated VDN meets the constraints defined by the comparator and the threshold value.
- The expected wait time at the specified priority level for the indicated split/skill, or for the call meets the constraints defined by the comparator and the threshold value.
- The oldest call-waiting in the indicated split/skill at the specified priority level (or higher) has been waiting for a period of time within the constraints defined by the comparator and the threshold value, which is expressed in seconds.
- The number of staffed agents in the indicated split/skill meets the constraints defined by the comparator and the threshold value.
- Digits collected using the **`collect digits`** command match the criteria defined by the comparator for the specified digit string. Or, the digits are found or not found, depending upon the option chosen, in the specified Vector Routing Table. The # digit can be tested against as a single digit.
- The ani digits match the criteria defined by the comparator for the specified digit string. Or, the ani digits are found or not found, depending upon the option chosen, in the specified Vector Routing Table.
- The II-digits match the criteria defined by the comparator for the specified digit string. Or, the II-digits are found or not found, depending upon the option chosen, in the specified Vector Routing Table.
- Time-of-day criteria are met.

  Note:

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

- The Expected Wait Time (EWT) for the call is decreased by a period of time within the constraints defined by the comparator and the threshold value, which is expressed in seconds. The improvement in EWT is defined by calculating the difference between the call's current EWT and its EWT were it to be queued to the resource specified in the command.

- The call's position in the interflow-eligible portion of the queue meets the condition defined by the comparator and the threshold value (representing queue position counting backward from 1, which is the head of the eligible queue).
- For Attendant Vectoring, there is no way to check ahead of time to see if a call can queue, and there is no way to check if, after the fact, a call queued successfully. The **`queue-fail`** command allows you to provide additional routing if a call to an attendant vector fails. You can redirect the call to another step or to another vector if the call cannot be queued.

General considerations

When a **`goto`** command is used in a vector step to connect to a different VDN, the following events occur:

1. Vector processing continues at the first step in the branched-to vector.
2. Call (if queued) remains in queue.
3. Wait treatment (if any) is continued.
4. Processing then continues in the receiving vector at step 1.

Unconditional branching

Unconditional branching passes control from the current vector step to a preceding vector step, a subsequent vector step, or to another vector. Unconditional branching is implemented when a **`goto step`** or **`goto vector`** command is associated with an unconditionally parameter.

The following example shows a vector that uses an unconditional branching step:

Unconditional branching example

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

In the example shown above, the unconditional branch statement in step 7 establishes a loop between steps 5 through 7. Vector processing within the loop terminates when:

- An agent answers the call
- The system recognizes that the caller abandoned the call

Conditional branching

Conditional branching passes control from the current vector step to a preceding vector step, a subsequent vector step, or to another vector. Conditional branching is enabled by a **`goto step`** or **`goto vector`** command when a conditional statement is associated with the command.

The list of condition statements that can be assigned, which depends on the features enabled in your Communication Manager installation, is summarized in the following table.

Condition statement [48]	Basic Call Vectoring	Advanced Vector Routing[49]	ANI and II-Digits Routing[50]
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		

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

[48] For information about the comparators that can be used with these condition statements, see goto step and goto vector commands on page 259. 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.

[49] For more information about this feature, see Advanced Vector Routing - EWT and ASA.

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 in the *Avaya Aura™ Call Center Feature Reference* document.
`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

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 <> #

[50] For more information about this feature, see ANI /II-digits routing and Caller Information Forwarding (CINFO) on page 155.

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

Description of conditionals

You can use any of three registered and unregistered vector conditionals with the **goto step** or **goto vector** commands to set up alternate routing or treatment of calls. These three conditionals test which type of server is processing the vector. These conditionals also test the registration status of media gateways and port networks connected with that server. The three conditionals are as follows:

- media-gateway - monitors the H.248 Media Gateway registration status
- port-network - monitors the port network gateway registration status
- server - monitors the type of server currently processing the vector step for the call

These conditionals allow alternate routing or treatment of calls based on the current status of the server processing a call, such as:

- The H.248 Media or Port Network Gateway is not registered with the Media Server processing the call
- A backup server is processing the call in *survivable* mode due to a failure of IP connectivity.

Reason to use the media gateway, port network, and server vector conditionals

These conditionals allow you to monitor the Communication Manager when it is running in a *survivable* configuration. Based on that knowledge, you can use alternative call handling or resources. For example, you can use different announcements, Interactive Voice Response systems (IVRs), or different skills to provide the best possible call handling with the available resources.

Syntax of gateway conditionals

The 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 Communication Manager server currently processing the vector step for the call.
= registered	Returns true if the specified gateway is registered with the server.
<> registered	Returns true if the specified gateway is not registered with the server processing the vector step.

When gateways are not equipped

If the specified gateway number or gateway type is not administered, the test fails and logs a vector event. Vector processing continues at the next step in the vector.

Syntax of server conditionals

The 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. ...
```

goto step and goto vector command 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.

goto step and goto vector command interactions with Avaya IQ

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.

goto step and goto vector command interactions with CMS/BCMS

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	Vector Flow Out	
GOTOCALLS/ GOTOTIME		
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

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*[51] **1st** (VDN skill) **2nd** (VDN skill) **3rd** (VDN skill)	**for extension**	*extension no.* **latest** **active** [2] *A-Z, AA-ZZ* *V1-V9*

[51] 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.

[52] Active refers to the VDN specified by VDN Override settings. Latest refers to the VDN specified for the current vector.

	split	hunt group[1]		

For information about unexpected results, see Troubleshooting vectors.

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.

Related topics:

Using a messaging step in a vector on page 277
Leaving a recorded message on page 277
Answer supervision considerations for the messaging command on page 278
messaging command feature interactions on page 278
messaging command interactions with Avaya IQ on page 279

messaging command interactions with CMS on page 279
messaging command interactions with BCMS on page 280

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 for the messaging command

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

messaging command feature interactions

Interaction	Description
Messaging-system hunt group	The command can use a messaging-system hunt group 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	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

Interaction	Description
mailbox extension	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 277

messaging command interactions with Avaya IQ

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.

messaging command interactions with CMS

When a queued call successfully goes to the messaging split, OUTFLOWCALLS/ OUTFLOWTIME (1st split/skill) and DEQUECALLS/DEQUETIME (2nd/3rd splits [skills]) are tracked in the split/skill tables. 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.

messaging command interactions with BCMS

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

queue-to command

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`**[53]			
	`attendant`[53]	extension no.		
	`best`			

	`hunt-group`[53]	group number[54]	**`pri`**	`priorities:`
				`l = l`

[53] This item is available with only the Attendant Vectoring feature.

[54] A valid group number is a vector-controlled hunt group of any type (ACD, UCD, and so on).

	skill	hunt group[55], VDN skills (**1st, 2nd, 3rd**)		ow **m = m** edium **h = h** igh **t = t** op
	split	hunt group[55]		

For information about unexpected results, see Troubleshooting vectors.

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.

[55] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

- 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 242.

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.

Related topics:

queue-to split command on page 283

Answer supervision considerations for the queue-to command on page 285

queue-to command feature interactions on page 285

queue-to command interactions with Avaya IQ on page 286

queue-to command interactions with CMS on page 286

queue-to command interactions with BCMS on page 287

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.

Related topics:

queue-to split command general considerations on page 283
Multiple split queuing on page 283
Option with the VDN as the coverage point on page 284

queue-to split command 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 for the queue-to command

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

queue-to command 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.

queue-to command interactions with Avaya IQ

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.

queue-to command interactions with CMS

Calls queued using a **`queue-to split/skill`** command are tracked as CALLSOFFERRED and LOWCALLS/MEDCALLS/HIGHCALLS/TOPCALLS.

Split/skill calls are reported in the standard reports according to the final disposition of the call.

The presence of the command in a vector enables the calls that are serviced by the vector to be vector-directed. When such a call is answered by an agent, the call is tracked as ACDCALLS/ANSTIME, and it is reported as ACD Calls, Split/skill ACD Calls, and Avg Speed Ans. If the call is also queued to other splits/skills, OUTFLOWCALLS/OUTFLOWTIME is tracked in the first split/skill to which the call queues, and Flow Out is reported (unless the split/skill turns out to be the answering split/skill). DEQUECALLS/DEQUETIME is tracked in the second and third splits/skills if these splits/skills are not the answering split/skill, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time. However, if the second or third split/skill is the answering split/skill, INFLOWCALLS is tracked in the split/skill, and the call is reported as Flow In.

If the call abandons after the command queues the call to a split/skill, ABNCALLS/ABNTIME is tracked for the vector, the VDN, and the first split/skill to which the call is queued. The call is reported as Aban Call and Avg Aban Time. If the call is also queued to other splits/skills, DEQUECALLS/DEQUETIME is tracked in these splits/skills, and the call is reported as Dequeued Calls and Dequeued Avg Queue Time.

BSR status poll calls are not counted as interflows. BSR interflows are now tracked as network interflowed calls (NETCALLS) by the CMS at the receiving switch. The CMS tracks a call's accumulated time-in-VDN as NETINTIME (that is, the NET_TIME value on the CMS at switch C combines the time a call has spent in VDNs at any previous locations, as communicated by information forwarding. The NETINTIME can be added to the time spent in the local switch to provide reports that include the total time the call has spent in the call center network (e.g., total ASA).

For more information on the database items and reports, see Avaya Avaya CMS Database Items and Calculations, and Avaya CMS Supervisor Reports.

queue-to command interactions with BCMS

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

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.

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 242.

Related topics:

Answer supervision considerations for the reply-best command on page 288
reply-best command interactions with Avaya IQ on page 288
reply-best command interactions with CMS/BCMS on page 289

Answer supervision considerations for the reply-best command

The **reply-best** step does not return answer supervision.

reply-best command interactions with Avaya IQ

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.

reply-best command interactions with CMS/BCMS

Operation of the **`reply-best`** command is not reported or tracked by the CMS or by the BCMS.

return command

This section includes the following topics:

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.

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.

Related topics:

When return destination information is not stored on page 290

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

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[56]	digits	with coverage	y, n					
	meetme[57]							
	number[58]	up to 16 digits (0-9) <digits>[59][A-Z, AA-ZZ, V1-V9][60]<digits>*<digits>A <digits>#<digits>A <digits>~p<digits>A <digits>~m<digits>A <digits>~s<digits>A <digits>~w<digits>A	with cov	y, n	if	digit	>, >=, <>, =<, <=	*0-9#* [64]
						interflow-qpos	<= <=	1-9

[56] 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.

[57] This item is available only with meet-me conference vectors.

[58] A destination for the route-to is entered in the number field. This field can contain an administration limit of a maximum of 16 decimal digits or combination of characters and numbers that total 16. Special notations (e.g., ~p) with a ~ followed by a character are counted as two digits towards the 16. The number field supports some feature activations using Feature Access Codes (FACs) alone or followed by digits including Service Observing, Remote Logout of Agent, remote access extension, attendant access number, Forced Logout/Aux and other destination numbers that can be dialed with a phone. The number field also supports vector variables (A-Z, AA-ZZ) and VDN variables (V1-V9) whose value in decimal digits is defined elsewhere before the route-to number command is to be executed.

[59] The notation <digits> means that 1 or more digits in the range of 0 to 9 can be inserted when necessary for the application. The total of digits and characters must be within the 16 digit positions total.

[60] Either a vector variable (A-Z, AA-ZZ) or VDN variable (V1-V9) can be entered at the end of any entry (digits or special character) or entered in place of <digits>; this is shown with an "A" in the other examples. Each variable whether a single character or double character counts as two digits towards the maximum of digits in the number field. The variable can be preceded by digits as long as the total is within the 16 digit/character position limit. The variable must always be the last entry and can not be followed by a digit. Use of a variable allows having a route-to number destination address of more than 16 digits since a variable can be assigned up 16 digits during processing and will be combined with the entry in the number field.

		<digits>~W<digits>A ~r[61]~r+[62]*[63]#[63]				unconditionally
	name1 [65]	with coverage	y, n			
	name2 [65]					
	name3 [654]					

For information about unexpected results, see Troubleshooting vectors.

Operation

The **route-to digits** 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 number** command attempts to route a call to the destination specified by the administered digit string.

Related topics:

Conditional route-to statements on page 293
Destinations for the route-to command on page 293
Command completion and failures on page 293
About the number field on page 294
Abbreviated Dialing special characters on page 295
Using the route-to command for NCR on page 295
Coverage parameter on page 297
Route-to number command on page 297
Answer supervision considerations for the route-to command on page 300
route-to command feature interactions on page 300
route-to command interactions with Avaya IQ on page 302
route-to command interactions with CMS on page 302
route-to command interactions with BCMS on page 304

[61] When the specified number is preceded by ~r, Network Call Redirection (NCR) invocation is attempted back over the trunk group to the network Service Provider. The ~r sequence is counted as two digit positions toward the 16 total. The + character is a special indication for E.164 numbering required by some network Service Providers for NCR invocation over SIP trunking. The "+" character is counted as two digit positions towards the 16 total. The ~r or ~r+ entries must be in the initial digit/character positions of the number field.

[62] By prefixing a VDN number with ~r* or ~r# in the route-to number command, you can access a Feature Access Code or remote phone number over Network Call Redirection.

[63] By prefixing a VDN variable with * or #, you can access a Feature Access Code using the route-to number command. Using the * prefix you can also access a remote number for which you need to dial "*9." For this, you need to set up and call a VDN that includes 9 followed by the phone number. For example, route-to number *V1 if cov unconditionally command can route to an external number *913032451234 if V1 is set up as "913032451234." .

[64] 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.

[65] This item is available only with the Dial by Name feature.

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 Hunt Group/Queue
- Attendant access number
- Remote extension (UDP/DCS)
- External number, such as a TAC or AAR/ARS FAC followed by a public or private network number (for example, 7-digit ETN, 10-digit DDD, and so on.)
- Remote Access Extension.
- Service Observing FAC
- Another Avaya switch via Network Call Redirection

  For more information, see Using the route-to command for NCR on page 295.
- Remote Logout of Agent FAC
- Forced Agent Logout/Aux Work By Location/Skill FACs

Note:

The active VDN’s Class of Restriction (COR) settings are used for calling permissions when routing via the `route-to` command. The route-to digits command can be used to implement an automated attendant function.

Command completion and failures

The `route-to digits` command fails if no digits are collected. Vector processing continues at the next vector step.

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

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

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

A route-to step in a vector is treated as 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 that can be entered in the number field is 16.

You can use the following variables in the number field alone or in combination with special characters and preceding digits (see Syntax and valid entries on page 291 for details):

- A - Z or AA – ZZ vector variable defined in the Variables for Vectors table
- V1 - V9 VDN variable assigned to the active VDN for the call

Each variable whether a single character or double character counts as two digits towards the maximum of digits in the number field.

Using the route-to command for NCR

You can use variables with the ~r special character in the **`route-to`** number command to activate Network Call Redirection (NCR). The digits following the invocation character are the

Service Provider network redirect to phone number address without inclusion of an internal access code. Use the any of the following formats:

- ~r<*number up to 14 digits*> - This option allows you to enter a specific number. For example, ~r13035552345. This format can contain only up to 14 digits for the redirection address because ~r takes up 2 digit spaces of the 16-digit number field.
- ~r*[A-Z or AA-ZZ]* - This option allows you to enter a vector variable as the redirect to address. For example, ~rA. The variable can have a value up to 16 digits during processing since it is not counted towards the number field 16 digit limit during administration. The variable letter(s) can have preceding digits following the ~r. For example ~r123AB. The variable character(s) always count as two digits towards the 16 digit number field limit.
- ~r*[V1-V9]* - This option allows you to enter a VDN variable. For example ~rV1. This variable can also have a value up to 16 digits and can have preceding digits following the ~r. The variable always counts as two digits towards the 16 digit number field limit.
- ~r+<*number up to 12 digits*> - This option allows you to enter a specific number for the special case where a network Service Provider requires the + character to indicate E. 164 numbering for NCR invocation over a SIP trunking interface with the network. For example, ~r+305558754. This format can contain only up to 10 digits for the redirection address because ~r and + both take up 2 digit spaces each for a total of 4 spaces of the 16-digit number field.
- ~r+*[A-Z or AA-ZZ]* - This option allows you to enter a vector variable as the E.164 numbered redirection address. For example, ~r+A. The variable can have a value up to 16 digits during processing since it is not counted towards the number field 16 digit limit during administration. The variable letter(s) can have preceding digits following the ~r+. For example ~r+123AB. The variable character(s) always count as two digits towards the 16 digit number field limit.
- ~r+*[V1-V9]* - This option allows you to enter a VDN variable as the E.164 numbered redirection address. For example ~r+V1. This variable can also have a value up to 16 digits and can have preceding digits following the ~r+. The variable always counts as two digits towards the 16 digit number field limit.

For examples, see Using route-to number ~r vector step to activate NCR in the *Avaya Aura™ Call Center Feature Reference* document.

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 371.

Route-to number command

The **route-to number** command is used to route calls to a vector-programmed number.

Related topics:

About interflow routing on page 297
General considerations for interflow routing on page 298
Service Observing routing on page 299

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

- Feature Access Code
- Remote phone number for which you need to dial *9 as prefix

General considerations for interflow routing

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:
    - Avaya Aura™ Communication Manager Feature Description and Implementation
    - *Avaya Aura™ Call Center Feature Reference*
    - *Administering Avaya Aura™ Call Center Features*
- For more information about the route-to number A to Z and AA to ZZ variables, see Variables in Vectors.

Answer supervision considerations for the route-to command

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

route-to command feature interactions

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 299.

Note:

The chapter *Operation details for The route-to command* gives a detailed description of the feature interactions for the `route-to` number with and without coverage command.

route-to command interactions with Avaya IQ

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.

route-to command interactions with CMS

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

Routed to trunk		
Database item	**Report heading**	**Notes**
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	
	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.

route-to command interactions with BCMS

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

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[66] type A-Z or AA-ZZ vector variable	=	user-assigned[66]type A-Z or AA-ZZ vector variable	**ADD** **SUB** **MUL** **DIV** **CATL** **CATR** **MOD10** **SEL**	user-assigned[66] A-Z or AA-ZZ vector variable
	asaiuui A-Z or AA-ZZ vector variable				
	digits[67]		system-assigned[68]A-Z or		system-assigned[68] A-Z

[66] Only global or local collect type vector variables can be assigned using the set command.

[67] The collected digits buffer holds up to 16 digits.

[68] For example, ani, asaiuui, doy, and so on.

Command	Variables or digits	Operand1	Operator	Operand2
		AA-ZZ vector variable		or AA-ZZ vector variable
		V1-V9 VDN variable		directly-entered numeric string[69]
		digits		V1-V9 VDN variable
		none		**digits**
				none

For information about unexpected results, see Troubleshooting vectors.

stop command

Purpose

The **stop** command halts the processing of any subsequent vector steps.

Syntax

```
stop
```

For information about unexpected results, see Troubleshooting vectors.

Requirements

No special requirements.

Operation

A vector stops processing when:

- A vector step includes a **stop** command
- The last step vector step is processed
- 10,000 vector steps have been processed

The **stop** command halts the processing of any subsequent vector steps. After the **stop** command is processed, any calls that are already queued remain queued, and any wait treatment is continued. Wait treatments include silence, ringback, system music, or alternate audio or music source.

[69] Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.

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.

Related topics:

Answer supervision considerations for the stop command on page 307
stop command feature interactions on page 307
stop command interactions with Avaya IQ on page 307
stop command interactions with CMS on page 308
stop command interactions with BCMS on page 308

Answer supervision considerations for the stop command

The command has no effect on answer supervision.

stop command 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.

stop command interactions with Avaya IQ

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.

stop command interactions with CMS

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.

stop command interactions with BCMS

None.

wait-time command 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.

Related topics:

Multiple audio/music sources on page 308
Answer supervision considerations for the wait-time command on page 310
wait-time command feature interactions on page 310
wait-time command interactions with Avaya IQ on page 310
wait-time command interactions with CMS/BCMS on page 310

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 *Administering Avaya Aura™ Call Center Features*, and the Announcements/Audio Sources screen reference in Administering Avaya Aura™ 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 <extension> segment of the vector. In this case, if **<treatment2>** is specified as `continue`, then the caller hears silence.

Answer supervision considerations for the wait-time command

If the **`music`** or **`audio source`** treatment is included in the command, answer supervision is triggered. If the command is encountered and answer supervision was sent previously, the caller hears the treatment specified in the current command. If, for a CO trunk user, the command with **`silence`**, **`ringback`**, or **`i-silent`** treatment is encountered prior to answer supervision, the caller continues to hear ringback from the CO.

wait-time command feature interactions

Feature interaction	Description
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.

wait-time command interactions with Avaya IQ

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.

wait-time command interactions with CMS/BCMS

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.

Variable, digits buffer, and asaiuui

Variable

You can enter user-assigned A to Z and AA to ZZ collect vector or the asaiuui variable types in the **Variable** field. The collect vector variable can be either local or global.

For more information, see User-assigned vector variable types on page 112.

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 317.

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 312 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 15785670

459
```

- The ASAI UUI string associated with the call can be up to 96 digits.
- The ASAI UUI string is the user data portion of ASAI UUI IE or ASAI UUI portion of the shared UUI, which contains a total of 99 bytes. 2 bytes for the header (op code and the data length) plus a protocol discriminator byte and a maximum of 96 bytes of actual user data. The data length byte value includes the protocol discriminator byte plus the actual user data bytes.
- The protocol discriminator is unusually set 0x00 (user specified) or to 0x04 (indicating IA5 characters) by the adjunct and the setting is retained by Communication Manager. If the Call Vectoring set command initially creates the ASAI UUI data, the protocol discriminator is set to 0x04 by Communication Manager.
- The set command can only change the actual data bytes following the protocol discriminator. The assignment affects the set of bytes defined by the start and length parameters for the vector variable and is limited to the decimal digits (0-9) subset of the ASCII (IA5) character set.
- The ASAI user data sent to the reporting adjunct (Avaya CMS/IQ) 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 315.

Example: Assigning a # character to the fifth digit position using divide by 0.

```
Define A as asaiuui type with start=5, length=1
ASAI UUI for caller = 1234567890

set A = none DIV 0

ASAI UUI for caller = 1234#67890
```

If the result of a set string operation, such as CATL, CATR, or SEL is greater than 16 digits, you can use the asaiuui variable definition to truncate the right end of the string to the required number of digits.

Example: Truncating result to 16 digits.

```
Define A as asaiuui type with start=1, length=16‟
digits”
 (from the collect step) = 1234567890

set A = digits CATR 1234567890

ASAI UUI for caller = 1234567890123456
```

- UUI can be transported betweenCommunication Manager systems using what is called “Service Provider” format or “Shared” format. The Service Provider format only carries a single data element usually containing ASAI/CTI user data associated with the call. The Service Provider UUI IE contains an op code (0x7E), a data length byte and a protocol discriminator byte. This is then followed by up to 96 data bytes. The Shared format is a multi-data element format that can contain UCID, ASAI user data, collected digits, VDN name, etc. associated with the call. Each data element is designated by a unique op code defined by the Avaya shared UUI specification. The ASAI user portion of the shared UUI consists of the op code (0xC8) and the data length byte followed by the user data.
- With Service Provider format the protocol discriminator is unusually set to 0x00 (indicating that the following data is in a user specified format) or to 0x04 (indicating IA5 ASCII characters) by the adjunct and the setting is retained by Communication Manager.

- The protocol discriminator for the shared UUI format is usually set to 0x00 (user specified), however 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 Communication Manager, otherwise the protocol discriminator setting is not changed. In the case of Shared format the UUI IE protocol discriminator is set to 0x04 regardless of what else is carried by the shared UUI.
- The set command can only change the actual data bytes and 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 312.
- 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.

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 112.

- 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 104.

- V1 to V9 VDN variables

  For more information, see VDN variables on page 28.

- 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 112.

- 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 104.

- V1 to V9 VDN variables

  For more information, see VDN variables on page 28.

- **`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.

Set command considerations

Dial-ahead digits and the digits buffer

The digits buffer in the `set` command does not include dial-ahead-digits, nor does the digits buffer overwrite any current dial-ahead digits unless there is a subsequent collect step.

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 or asaiuui type vector variable, or to the Digits buffer. If a **set** command attempts to assign a value to a system-assignable vector variable or any other unsupported variable type during vector processing, the **set** command fails and a new assignment not allowed vector event is logged. Vector processing continues at the next step in the vector.

Assigning a new value to a collect variable

If a **set** command assigns a new value to a collect user-assignable vector variable, this new value applies to all subsequent references to that variable in vectors and displays in the Variables for Vector table in the Assignment field.

Determining the number of digits

To determine if the number of digits in variable A is 6 digits, use the following example.

```
1. set B = A MOD10 6
2. goto step 8 if B = # [if it branches to 8, A does not have 6 digits]
3. ...[else A does have 6 digits]
```

Clearing the digits buffer

Once all necessary processing for the collected digits buffer has completed, this example describes how to use the **set** command to clear the collected digits buffer. This prevents the answering agent from 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 1
```

wait-time command

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`[70]	**`mins`**		*audio source ext.*[71] *A-Z, AA-ZZ V1-V9*	**`then`**	**`music`**
						`ringback`
						`silence`
						`continue`[72]

	`0-8`[70]	**`hrs`**				

For information about unexpected results, see Troubleshooting vectors.

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

wait-time command basic operation

The specified feedback is given to the caller, and vector processing waits the specified time before going on to the next step. If the time specified is `0`, feedback is provided without any

[70] This option is not available for vector administration done through Avaya Call Management System or Visual Vectors.

[71] This consists of a valid announcement or music source extension that is defined on the announcement audio sources form.

[72] 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.

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 in *Avaya Aura™ Call Center Feature Reference*.

Multiple audio or music sources on delay

You can specify an alternative audio or music source for a vector wait-time step. This alternative source can be any extension number that is administered on the Announcements/Audio Sources screen. For instructions for entering an audio or music source on this screen, see Administering Avaya Aura™ 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 Avaya Aura™ Communication Manager Feature Description and Implementation.

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.

Chapter 14: Considerations for the vectoring features

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).

Note:

If EAS is optioned, skill replaces split.

This section includes the following topics:

- Displaying VDN names for vector-initiated DACs on page 325
- Call transfer to VDNs on page 331
- VDN Return Destination

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:

- Display VDN for Route-to DAC feature operation on page 326
- Display VDN for Route-to DAC prerequisites on page 327
- Administering the Display VDN for Route-To DAC feature on page 328
- Methods for creating vectors that use the Display VDN for Route-to DAC feature on page 329
- Interactions with other Communication Manager features on page 330

Related topics:

Display VDN for Route-to DAC feature operation on page 326
Display VDN for Route-to DAC prerequisites on page 327
Administering the Display VDN for Route-To DAC feature on page 328
Methods for creating vectors that use the Display VDN for Route-to DAC feature on page 329

Display VDN for Route-to DAC feature operation

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 withcov 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.

Related topics:

Station display formats on page 327

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.

Display VDN for Route-to DAC 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 328.

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. The **Expert Agent Selection?** field must be set to `y`.
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`**.

Methods for 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.

Related topics:

Using collect digits and route-to digits commands on page 329
Using route-to number commands on page 329
Using adjunct routing link commands on page 330

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:

Interaction	Description
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**

Interaction	Description
	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.

Call transfer 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

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 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**. For an example of limiting the number of returns to the Return Destination, see Example 2 under collect type variable in *Programming Call Vectors in Avaya Aura™ Call Center*. 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
```

Chapter 15: Troubleshooting vectors

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

Call vectoring command success/failure criteria	
	calls receive 45 seconds of ringback.)
check split	
Fails if any of the following are true: • Vector conditional is false. • Split's queue is full. • Split is not vector-controlled. • Call is already queued at the specified priority to the specified split. • Call is already queued to three different splits.	Continue vector processing with the next sequential step.
Otherwise:	
Succeeds, and the call is terminated to an agent.	Exit vector processing, and pass control to call processing.
Succeeds, and the call is queued or requeued in the specified split at the specified priority.	Continue vector processing with the next sequential step.
collect-digits	
Fails if any of the following are true:	
Call originates from an outside caller who is not using a touch-tone telephone.	Call Prompting timer takes effect, command times out, and vector processing continues at the next vector step.
No 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.
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.	Continue vector processing with the next sequential step.

Call vectoring command success/failure criteria	
• Status Poll VDN number is invalid. • Status Poll fails because all trunks are busy.	
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.
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.	Continue vector processing with the next sequential step.
Succeeds if the step condition is met.	goto step - continue vector processing with the destination step goto vector - continue vector processing with the first nonblank step of the destination vector.

Call vectoring command success/failure criteria	
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.	Continue vector processing with the next sequential step.
Otherwise, succeeds.	Terminate vector processing.
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.	Continue vector processing with the next sequential step.
Otherwise:	
Succeeds, and the call is terminated to an agent.	Exit vector processing, and pass control to call processing.
Succeeds, and the call is queued or requeued in the specified split at the specified priority.	Continue vector processing with the next sequential step.
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.

Call vectoring command success/failure criteria	
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.
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.

Unexpected feature operations	
Customer observations	**Causes**
	Also, use the interflow-qpos conditional (see How enhanced LAI works).
Remote agent receiving phantom calls when vectoring uses qpos conditional.	Interflow-qpos threshold may be set too low.
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 in the *Avaya Aura™ Call Center Feature Reference*.

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	
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.
`busy`	
Ringback heard instead of busy tone.	Unanswered CO trunk.
`check`	

Unexpected command operations	
Customer observation	**Cause**
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.
	Announcement restore in progress.
	Dial ahead digit exists.
Collect step and announcement skipped.	TTR not in system.
	Link to PN that has TTR is down.
	TTR queue full.

Unexpected command operations	
Customer observation	**Cause**
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.
	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.

Unexpected command operations	
Customer observation	**Cause**
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`[73]	
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.
	All ports busied out.
	Announcement restore in progress.
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.

[73] Refer to the Converse command debugging on page 349 section later in this section for more details on converse-on command debugging

Unexpected command operations	
Customer observation	**Cause**
	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).
Delay before messaging-system answers.	All messaging-system ports are busy, but there is space in the queue.
Busy tone.	Queue for the messaging-system voice ports is full.
Step skipped.	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`[74]	
Step skipped (that is, default treatment)	Invalid local extension.
	No trunks available.

[74] Complete operation details for the route to commands are presented in _ on page 0 .

Unexpected command operations	
Customer observation	**Cause**
	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 379.
`stop`	
Call dropped	Call not queued when vector processing stops.
`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.

Converse command debugging

The following table is intended to help your troubleshooting efforts with the `converse-on` command.

Note:

See Call flow and specifications for converse - VRI calls in the *Avaya Aura*™ *Call Center Feature Reference* document for details on the call flow for converse-VRI calls.

Table 9: 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
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.

Symptom	Cause	Analysis
	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.
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.

Symptom	Cause	Analysis
	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.

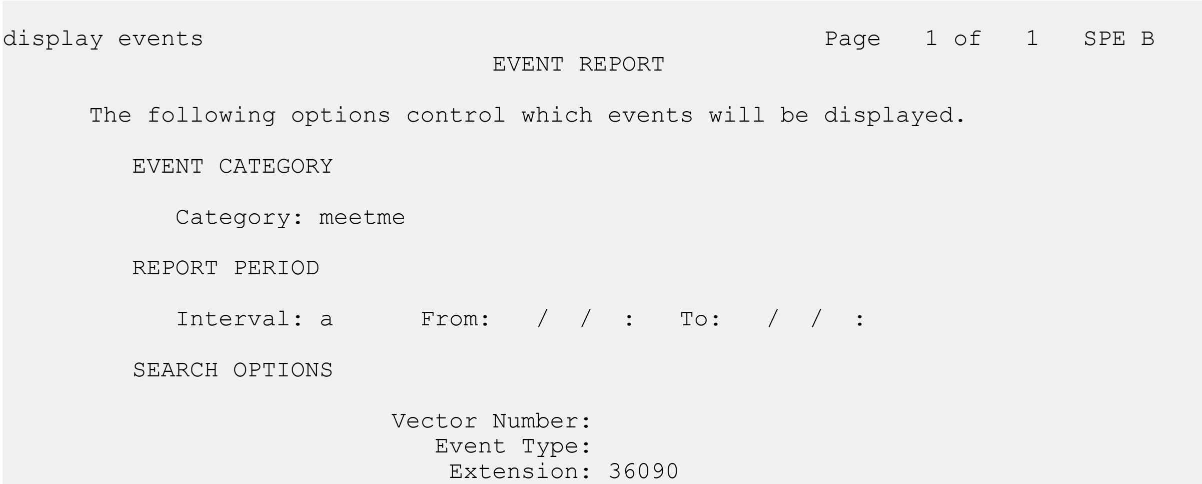
By displaying events, you can diagnose and correct each problem, as indicated by its corresponding event number, and eliminate the need for a technician to make on-site visits to do the same.

Related topics:

Displaying events criteria on page 351
Displaying an events report on page 352

Displaying 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.

Displaying an 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 353.
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.
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)

Event type	Event description	Explanation
		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.
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.

Event type	Event description	Explanation
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.
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 #.

Event type	Event description	Explanation
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 the chapter Operation details for the route-to command.
51	No digits to route-to	The route-to digits step was unable to route the call because the previous collect digits step failed to collect any digits. This could result from an error in vector programming (for example, a route-to digits step appears without a preceding collect digits step). More often, however, this results because the caller was unable to enter the required digits (that is, the caller was using a rotary telephone), or because the caller was not provided with enough information to do so (as can be the case for auto-attendant applications).
52	No available trunks	A **`route-to`** command was unable to reach the specified off-switch destination due to a lack of available trunks.
53	Route-to step failed	The step was unable to seize a trunk because of a hardware problem or glare.
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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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.
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

Event type	Event description	Explanation
		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).
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 in the *Avaya Aura™ Call Center Feature Reference* document.
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

Event type	Event description	Explanation
295	No room for collected dgt	UUI Feature Priorities page of the Trunk screen. For more information, see Information Forwarding in the *Avaya Aura™ Call Center Feature Reference* document.
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 in the *Avaya Aura™ Call Center Feature Reference* document.
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.
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 341 as well as Information Forwarding and Advanced multi-site routing in the *Avaya Aura™ Call Center Feature Reference* document.
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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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.
326	No ETSI ECT linkID	The PSTN switch has returned a FACILITY message to the local Communication Manager that includes the following reject component: LinkIDNotAssignedByNetwork. In this case, the local Communication Manager 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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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.
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.

Event type	Event description	Explanation
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 115.

Chapter 16: 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.

Table 10: Switch route-to command operation

Condition		cov = n Any Step	cov = y Any Step[75]
Invalid Destination[76]		Goes to next step, else stop	Goes to next step, else stop
VDN *Extension*[77]			
	Vector Assigned	Goes to new vector	Goes to new vector
	Vector Has No Steps	Stop[78]	Stop[78]

[75] 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 [77] 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 in the *Administering Avaya Aura™ Call Center Features* document.). 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.

[76] 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.

[77] 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).

Condition	cov = n Any Step	cov = y Any Step[75]
Station Extension Idle (all appearances idle)		
CF-ALL Active or CF-DA Applies	Forwards if possible, else next step, else stop[78]	Forwards if possible, else coverage, else busy
Coverage		
DA Applies	Rings idle app.	Coverage on DA
All Applies	Goes to next step, else stop[78]	Coverage
SAC Applies	Rings idle appearance	Coverage
None of Above Applies	Rings idle appearance	Call delivered and is allowed to cover
Station Extension Active (with idle 2-way app)		
CF-ALL Active	Forwards if possible, else next step, else stop[78]	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[78]	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[78]	Queues if possible, else coverage, else busy
CF-ALL Active or CF-DA Applies	Forwards if possible, else next step, else stop[78]	Forwards if possible, else coverage, else busy
Call Waiting to Analog Sta Would Apply	Goes to next step, else stop[78]	Call waits
Coverage		

[78] 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.

Condition		cov = n Any Step	cov = y Any Step[75]
	Ext Act Applies	Goes to next step, else stop[78]	Coverage
	Ext Bsy Applies	Goes to next step, else stop[78]	Coverage
	All Applies	Goes to next step, else stop[78]	Coverage
	SAC Applies	Goes to next step, else stop[78]	Coverage
	None of Above Applies (or hunt, fwd, or cov destination is unavailable)	Goes to next step, else stop[78]	Busy tone given
Extension with Incompatible COR		Goes to next step, else stop.	Goes to next step, else stop.
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[78]	Coverage, else busy
	No Idle App on Any Member	Goes to next step, else stop[78]	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[78]	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[78]	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[78]	Routes to local attendant
	Trunk Available	Call delivered	Call delivered
	Trunk Not Available	Goes to next step, else stop[75]	Queues if possible, else reorder

Condition	cov = n Any Step	cov = y Any Step[75]
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[78]	Reorder tone given
Queuing Assigned	Goes to next step, else stop[78]	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[78]	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[78]	Queues if possible, else busy tone
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[78]	Reorder tone given
Look Ahead Interflow (LAI) (feature active & routes over ISDN-PRI facility)[79]		
B-Channel Not Available	Goes to next step, else stop[78]	Queues if possible, else reorder

Condition	cov = n Any Step	cov = y Any Step[75]
B-Channel Available and Receiving Switch:		
Accepts	Interflow succeeds[80]	Call cut-through
Rejects	Goes to next step, else stop[78]	Call gets busy/disconnect
Receiving Switch w LAI Acting as Tandem Sees from Remote Receiving Switch:		
Call Accepted	Interflow succeeds[80]	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

[79] 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.

[80] 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).

Chapter 17: Advanced set command rules and applications

Arithmetic operations

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 379) • Less than 0 = # (see Invalid results for arithmetic operations on page 379) • 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 387
- SUB examples on page 388
- MUL examples on page 388
- DIV examples on page 389

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

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 381.

	Results	Operand1	Operand2
Field length	6	6	16
Entries allowed	[A-Z, AA-ZZ] digits	[A-Z, AA-ZZ] V1-V9 digits	[A-Z, AA-ZZ] V1-V9 digits

	Results	Operand1	Operand2
		none	none numeric values 0-999999999999999 99
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 382.	• none = 0 (null string) • # = none • Leading zeros allowed (007 = 007)	
Start and length parameters	See Start and length parameters in string operations on page 382.	N/A	N/A

For more information, see:

- CATL examples on page 390
- CATR examples on page 391

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 380.

	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

	Results	Operand1	Operand2
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 382.	• 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 382.	N/A	N/A

See SEL examples on page 391.

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

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 318.

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)	
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 385.	• 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 385.	N/A	N/A

See MOD10 operation examples on page 392.

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.

Chapter 18: Set command examples

Calculation examples

Arithmetic operation examples

ADD examples

The following table provides examples for using the ADD operator. The ADD operator adds operand1 and operand2.

Command	Result
set A = A ADD 456 A = 000123	123 + 456 = 579 This operation is useful for counters or loop control variables.The result does not retain the zeros.
set A = A ADD none A = 9999999999	9999999999 + 0 = # + 0 = # If an operand contains a value greater than 4294967295, the result is null (#).

set A = A ADD B
A = 000123
B = #
	123 + # = #
	The # character in a variable is seen as invalid (#).

set A = A ADD A A = none (or null string)	0 + 0 = 0 none is interpreted as 0
set B = A ADD 456 Variable definitions: • A = 1234 • B = collect type, length = 3, start = 2 (start is ignored)	1234 + 456 = 1690 = # This is an overflow since the result was greater than three digits.
set B = A ADD 456	1234 + 456 = 1690 No leading zeros are included in the result.

Command	Result
Variable definitions: • A = 1234 • B = collect type, length = 5, start = 2 (start is ignored)	

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 (#).
set A = A MUL 5 **A = #**	# x 5 = # The # character in a variable makes the result invalid (#).
set B = A MUL 10 Variable definitions: • A = 000123 • B: collect type, length = 3, start = 2 (start is ignored)	123 x 10 = 1230 = # The leading zeros are ignored in the operation. The result is greater than three digits, causing an overflow condition.

DIV examples

The following table provides examples for using the DIV operator. The DIV operator divides operand1 by operand2.

Command	Result
set A = A DIV 10 A = 000123	120 / 10 = 12 Leading zeros are ignored and the results are not padded with leading zeros.
setA = A DIV 2 A = 5	5 / 2 = 2.5 = 2 The result is rounded down to the nearest whole integer. Results do not contain any decimal values.
set A = A DIV 1000 A = 000123	123 / 1000 = 0.123 = 0 The result is rounded down to the nearest whole integer.
setA = A DIV A A = 9999999999	# / # = # This operation is an operand overflow because the operand values exceed 4294967295.
set A = A DIV # A = 000123	000123 / # = # A # character in an operand makes the operation invalid (#).
set B = A DIV 100 Variable definitions: • A = 12345678 • B: collect type, length = 4, start	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.

Command	Result
= 3 (start is ignored)	
set B = A DIV 100 Variable definitions: • A = 12345678 • B: collect type, length = 6, start = 3 (start is ignored)	12345678 / 100 = 123456.78 = 123456 The result is rounded down to the nearest integer. This result is valid because the result is six digits in length.

String operation examples

CATL examples

The following table provides examples for using the CATL operator. The CATL operator concatenates the operand2 digit string to the left end of operand1.

Command	Result
set A = B CATL 0123 **B = 56789**	56789 CATL 0123 = 012356789
set A = A CATL A **A = #**	# CATL # = none
set A = B CATL 0123 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 = 0123456789012345 0123 = 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.

Command	Result
set A = B SEL 1 Operand1 contains no digits (#) or is set to zero. B = #	# SEL 1 = 0 The result is null padded with a zero. The SEL 1 adds a leading zero.

set A = B SEL C
Operand2 contains no digits or is set to zero.
B = 1234
C = #
	1234 SEL # = 1234 SEL 0 = none
	The result is none.

set A = B SEL 3 B = 120005	120005 SEL 3 = 005 The zeros are retained in the result.

set A = B SEL C
B = 1234567890123456
C = 99
	1234567890123456 SEL 99 = 1234567890123456
	Operand2 contains a number greater than 16, therefore a maximum of 16 digits is selected from operand1.

set A = B SEL 16
A = collect type, length = 10, start = 3
B = 1234567890123
	1234567890123 SEL 16 = 0123456789
	Selects 10 digits starting at position 3.

MOD10 operation examples

The following table provides examples for using the MOD10 operator. The MOD10 operator validates account numbers, membership numbers, credit card numbers, and checks string lengths using the Modulus 10 algorithm.

Command	Result
set B = A MOD10 13 **A = #**	# MOD10 13 = # The operation is invalid if either operand contains a “#” character or “none”.
set B = digits MOD10 A **A = 20**	B = # The operation is invalid because operand2 contains a number greater than 16.
set B = digits MOD10 A **A = none**	B = # The operation is invalid.
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	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
`digits = 1234`	

`set B = A MOD10 13`
`A = 1234567890128`
	1234567890128 MOD10 13 = 0 (zero)
	A 0 result means that the tested digits match the specified number of digits and passes the Modulus 10 algorithm.

`set B = A MOD10 13`
`A = 1234567890123`
	1234567890123 MOD10 13 = 5
	A 1-9 result means that the tested digits match the specified number of digits but fails the Modulus 10 algorithm.

Application examples

Validating numbers example

The XYZ company wants a write a vector subroutine that validates a credit card number before a query is sent to the appropriate credit card company for their validation.

The XYZ company created a subroutine that does the following tasks:

1. Prompt the user for the type of credit card used for the purchase.
   - 1 - diners club
   - 2 - american express
   - 3 - visa
   - 4 - master card
   - 5 - discover
2. If a valid card type (1-5) is entered, prompt for the credit card number.
3. Validate the first four digits of each card number prefix.

   The following table provides a list of card number identifiers and card number lengths for the major credit card companies.

Company	Card number identifier	Card length
Diner’s Club/Carte Blanche	300xxxxxxxxxxx 305xxxxxxxxxxx 36xxxxxxxxxxxx	14

Company	Card number identifier	Card length
	38xxxxxxxxxxxx	
American Express	34xxxxxxxxxxxxx 37xxxxxxxxxxxxx	15
VISA	4xxxxxxxxxxxx 4xxxxxxxxxxxxxxx	13, 16
MasterCard	51xxxxxxxxxxxxxx 55xxxxxxxxxxxxxx	16
Discover	6011xxxxxxxxxxxx	16

4. Perform a Luhn or Modulus 10 check on the whole number.
5. If the validation fails validation, indicate that the card number entered is invalid and prompt again for the credit card type.
6. If the card validates, return and let variable A contain the type of card, and the digits buffer contain the validated credit card number.

Examples

```
                            VARIABLES FOR VECTORS

Var Description                  Type    Scope Length Start Assignment       VAC

A   credit card type             collect L     1      1
B   4 digit prefix of cred card collect L     4      1
C   modulus 10 result            collect L     1      1
```

```
                                  CALL VECTOR

    Number: 3                Name: credit card chk
                                                Meet-me Conf? n           Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? n   ASAI Routing? y
 Prompting? y   LAI? n  G3V4 Adv Route? y   CINFO? n   BSR? y   Holidays? y
 Variables? y   3.0 Enhanced? y
01 collect      1    digits after announcement 4501     for A
  (Prompt for credit card type, 1-diners, 2-american, 3-visa, 4-mc, 5-discover)
02 collect      16   digits after announcement 4502     for B
  (Prompt for credit card number followed by #)
03 goto step   7           if B               in     table A
  (check if credit card prefix matches appropriate card type)
04 announcement 4503        (“
Bad number,Try again”
)
05 goto step   1           if unconditionally
06
07 goto step   13          if A                =      1 (Diners Club)
08 goto step   17          if A                =      2 (American Express)
09 goto step   20          if A                =      3 (Visa)
10 goto step   22          if A                >=     4 (Master Card, Discover)
11 goto step   1           if unconditionally  (unknown)
12
13 set          C      = digits MOD10 14      (Diners Club, check)
14 goto step   1           if C                <>     0 (not valid, try again)
15 return              (OKAY! VALID! RETURN!
                                digits buffer contains
                                valid creditcard number)
```

```
16
17 set          C      = digits MOD10 15      (American Express)
18 goto step   14          if unconditionally
19
20 set          C      = digits MOD10 13      (Visa check 13 digits)
21 goto step   15          if C                =      0 (OKAY! VALID! RETURN!)
22 set          C      = digits MOD10 16      (VISA, MASTER CARD,
                                       DISCOVER chk 16 digits)
23 goto step   14          if unconditionally
```

```
add vrt 1                                                       Page   1 of   3

                              VECTOR ROUTING TABLE

      Number: 1       Name: Diners Club         Sort? n

           1: 300?
           2: 301?
           3: 302?
           4: 303?
           5: 304?
           6: 305?
           7: 36??
           8: 38??
--------------------------------------------------------------------------------
add vrt 2                                                       Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 2       Name: American Express    Sort? n
           1: 34??
           2: 37??
--------------------------------------------------------------------------------
add vrt 3                                                       Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 3       Name: VISA                Sort? n
           1: 4???
--------------------------------------------------------------------------------
add vrt 4                                                       Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 4       Name: MASTER CARD         Sort? n
           1: 51??
           2: 52??
           3: 53??
           4: 54??
           5: 55??
--------------------------------------------------------------------------------
add vrt 5                                                       Page   1 of   3
                              VECTOR ROUTING TABLE

      Number: 5       Name: MASTER CARD         Sort? n
           1: 6011
----------------------------------------------------------------------------
```

A 19-digit credit card validation

This example determines the validity of a 19-digit credit card number. 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 385.

Using bilingual announcement example

You can program a vector to support bilingual or multilingual announcements. The following table describes the example vectors and announcements.

Announcement	Description
3000	Announcement in English and Spanish asking users to choose between English or Spanish prompts.
1001	Greeting in English [*Welcome ...*]
1002	[*Please enter your ID*]
1003	[*Please wait*]
2001	Greeting in Spanish [*Bienvenida...*]
2002	[*Incorpore su identificación*]
2003	[*Por favor espera*]

Notice that the announcements are administered so that extensions starting with 1 are in English and extensions starting with 2 are in Spanish.

```
1. collect 1 digit after hearing announcement 3000 for A
     a. goto step 1 if A > 2
     b. route to operator if A <= 0
2. set B = A CATR 001 [B = A001, A = 1 or 2]
3. announcement B
4. set B = A CATR 002 [B = A002, A = 1 or 2]
5. collect 16 digits after hearing announcement B for none
6. queue to skill
7. set B = A CATR 003 [B = A003, A = 1 or 2]
```

```
8. wait .. hearing B then music
9. goto step 8 if unconditionally
```

Because you can append at the beginning of the string or at the end, you can place a language digit at the beginning or at the end of the string. This example places the language digit at the beginning of the string.

Collecting an account number examples

The first example describes a vector designed to collect an account number without using the **set** command. The second example describes how to use the **set** command to solve the problem.

Related topics:

Example 1: Without using the set command on page 397
Example 2: Using the set command on page 397

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 examples

This chapter provides examples of how to use VDN variables to implement percentage routing application. Starting with Communication 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. See below for an illustration of how a complicated application can be implemented using variables and vectoring. Also, see the much simpler Setting up percentage routing using Policy Routing Table on page 403. For the details on the Percentage-Allocation Routing feature see Call Vectoring features in *Avaya Aura™ Call Center Feature Reference*.

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 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.

This topic includes the following two examples:

- Example 1: Percentage routing using VDN variables on page 399
- Example 2: Percentage routing using PRT on page 403

Example 1: Percentage routing using VDN variables

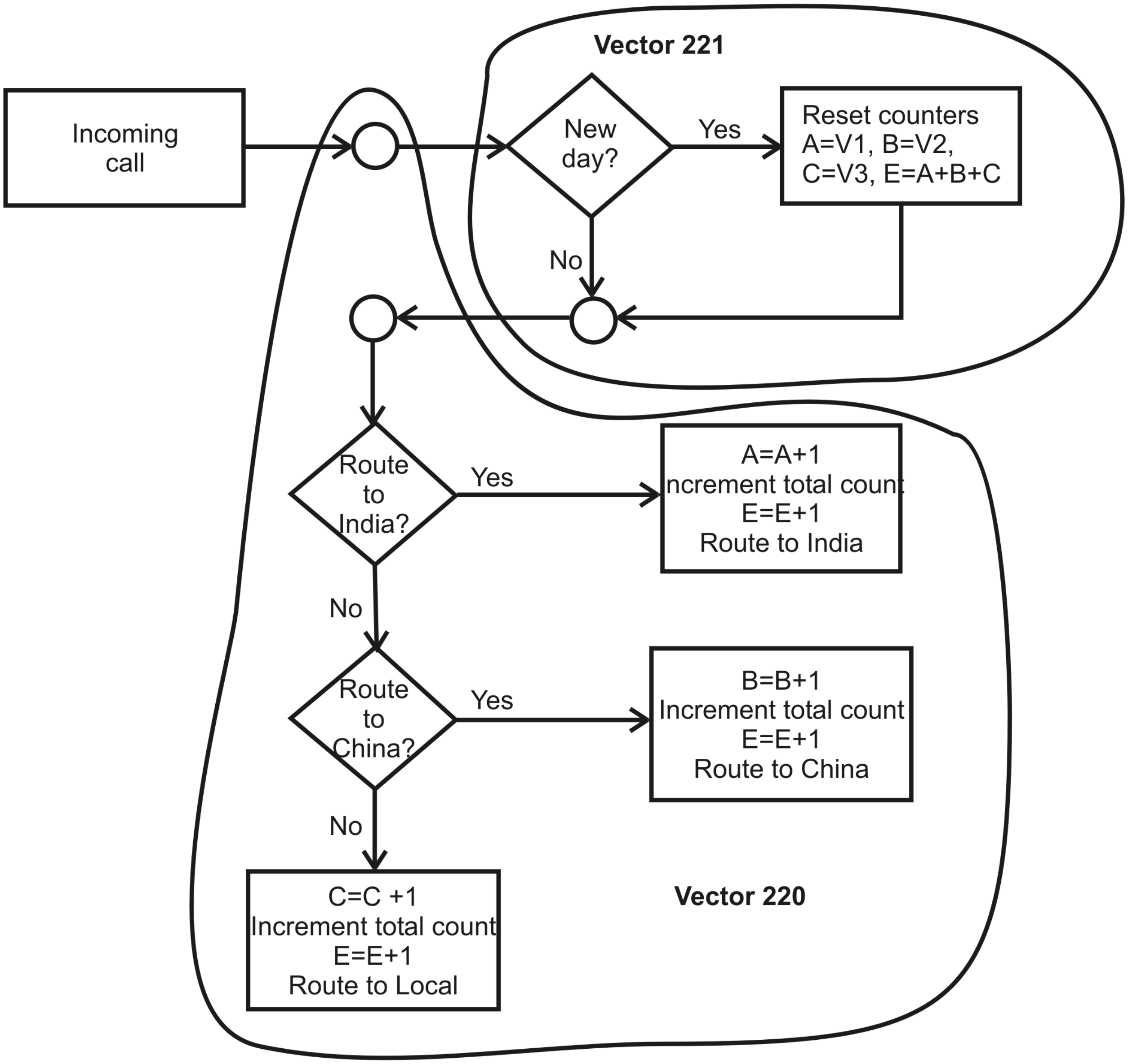
Overview of tasks in percentage routing using VDN variables example

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 in percentage routing using VDN variables example

The following flowchart provides an overview of this example.


Vector 221
Incoming call
New day?
Yes
Reset counters A=V1, B=V2, C=V3, E=A+B+C
No
Route to India?
Yes
A=A+1 Increment total count E=E+1 Route to India
No
Route to China?
Yes
B=B+1 Increment total count E=E+1 Route to China
No
C=C +1 Increment total count E=E+1 Route to Local
Vector 220

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 example

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
                           Destination: Vector Number        220

                  Meet-me Conferencing? n
                    Allow VDN Override? n
                                   COR: 1
                                   TN*: 1
                              Measured: external

               Service Objective (sec): 20
        VDN of Origin Annc. Extension*:
                            1st Skill*:
                            2nd Skill*:
                            3rd Skill*:
```

```
* Follows VDN Override Rules
```

```
change vdn 2220                                       Page   2 of   3
                            VECTOR DIRECTORY NUMBER

                                 AUDIX Name:
                        Return Destination*:
                    VDN Timed ACW Interval*:




                    Observe on Agent Answer? n

  Send VDN as Called Ringing Name Over QSIG? n

              Display VDN for Route-To DAC*? n
            VDN Override for ASAI Messages*: no

                   Allow Conference to VDN*? n
          Reporting for PC Predictive Calls? n
           Pass Prefixed CPN to VDN/Vector*? system
* Follows VDN Override Rules
```

3. Define a VDN variable for each country where calls are routed.

   Example:

```
change vdn 2220                                       Page   3 of   3
                            VECTOR DIRECTORY NUMBER

                                VDN VARIABLES*

                    Var  Description       Assignment
                    V1   India             25
                    V2   China             25
                    V3   Local             50
                    V4
                    V5
                    V6
                    V7
                    V8
                    V9





                   VDN Time-Zone Offset*: + 00:00
                  Daylight Savings Rule*: system



* Follows VDN Override Rules
```

4. Use the Call Vector screen to set up a primary vector that calculates percentages and routes calls accordingly.

   This is the main vector for processing calls placed to VDN 2220.

   Example:

```
change vector 220                                           Page   1 of   6
                                  CALL VECTOR

   Number: 220            Name: Percent Route       Background BSR Poll? n
```

```
                                          Meet-me Conf? n          Lock? n
    Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing?
n
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? n   Holidays? y
 Variables? y   3.0 Enhanced? Y
01 goto vector 221 @ step 1 if unconditionally
02 set D = A MUL 100 [D = calculated %, A = India routed calls]
03 etc. use steps 01-21 from the existing vector on page 402-403
```

5. Set up a subroutine vector to initialize the routing every day for the first call.

   This subroutine is called by step 1 in vector 220.

   Example:

```
change vector 221                                           Page   1 of   6
                                  CALL VECTOR

   Number: 221            Name: Reset New Day       Background BSR Poll? n
                                          Meet-me Conf? n          Lock? n
    Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing?
n
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? n   Holidays? y
 Variables? y   3.0 Enhanced? Y
01 goto vector step 8 if F = G [F = day of week for first call, G = day of
week]
02 etc. use steps 01-08 of the existing vector on page 403
```

Note:

1 = Sun, 2 = Mon, 3 = Tues, 4 = Wed, 5 = Thurs, 6 = Fri, 7 = Sat

6. Run a **list trace vdn** command to verify that each variable is updating correctly.

Example 2: Percentage routing using PRT

Overview of tasks in percentage routing using Policy Routing Table example

Overview of tasks for percentage routing using Policy Routing Table :

- Define VDN 2220 that incoming calls route to assigning the destination to a PRT table.
- Define the PRT table to route calls based on the desired percentages.
- Define the three VDNs that will route the calls to appropriate destination using a VDN variable V1 for route to destination phone number.
  - VDN: 2221 routes to China via 97051001 assigned to V1
  - VDN2: 2222 routes to India via 97051002 assigned to V1
  - VDN3: 2223 route to the local call center via 97051003 assigned to V1
- Define routing vector 220.

Setting up percentage routing using Policy Routing Table

The XYZ Company wants to (same allocation as in Setting up percentage routing example):

- 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 Communication Manager system

Following are the steps XYZ Company has to complete to set up percentage routing using the Policy Routing tables (PRT):

1. Define VDN 2220 for call handling:

   Example:

```
change vdn 2220                                       Page   1 of   3
                            VECTOR DIRECTORY NUMBER

                             Extension: 2220
                                 Name*: Percent Routing using PRT
                           Destination: Policy Routing Table   8

                  Meet-me Conferencing? n
                    Allow VDN Override? n
                                   COR: 1
                                   TN*: 1
                              Measured: external

               Service Objective (sec):
        VDN of Origin Annc. Extension*:
                            1st Skill*:
                            2nd Skill*:
                            3rd Skill*:



* Follows VDN Override Rules
```

2. Define the Policy Route Table to allocate percentages of routing to destinations.

```
change policy-routing-table 8                               Page   1 of   1
                              POLICY ROUTING TABLE
 Number: 2     Name: policy 8          Type: percentage   Period: 100-count

                                                  Target     Actual Call
 Index Route-to VDN  VDN NAME                     %      %      Counts
   1   2221     ABC in India                            25     0.0*     0
   2   2222     DEF in China                           25     0.0*     0
   3   2223     XYZ local center                     50     0.0*     0
   4
   5
   6
   7
   8
   9
  10
  11
  12
  13
  14
  15
                                          Totals 100                0
```

3. Define the VDNs (2221, 2222 and 2223) on the VDN screen each routing to the Destination set to Vector Number 220 as shown in Setting up percentage routing example. (See Vector Directory Number screen on page 401.)
4. As required by the application, set Page 2 of the VDN screen.
5. Set up the route destination numbers on page 3 for the VDN variable V1 as the following (For an example of a VDN page 3 screen, See Step 3 on page 402 of Setting up percentage routing example):
   a. VDN 2221: assign 97051001 to V1 (routes to China)
   b. VDN 2222: assign 97051002 to V1(routes to India)
   c. VDN 2223: assign 97051003 to V1 (routes to the local call center)
6. Define the routing vector 220.

```
change vector 220                                            Page   1 of   6
                                       CALL VECTOR

    Number: 220             Name: Percent Route        Background BSR Poll? n
                                               Meet-me Conf? n           Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing?
n
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? n   Holidays? y
 Variables? y   3.0 Enhanced? Y
01 route-to number V1 with cov n if unconditionally
```

Assigning ASAI UUI values

Pass In-VDN Time to an Adjunct example

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 Communication Manager 4.0 or later this application can be accomplished using the “vdntime” type vector variable to provide the accumulated “in-VDN” time in seconds, and the set command assignment to a “asaiuui” type vector variable to populate the ASAI UUI for the call with amount of time the call has waited.

In this example, define vector variable “A” as a “asaiuui” type with start 1, length 4 and scope local. Define vector variable “V” as a “vdntime” type, predefined with scope local and length 4.

The following is an example vector for this application:

```
01 wait 0 secs hearing ringback
02 set A = V SEL 4
03 adjunct routing link 1
04 wait 10 secs hearing ringback
```

Step 2 assigns the accumulated 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 example

By using the set command assignment to a vector variable defined as the “asaiuui” type, an adjunct-provided account number can be combined with the call in-VDN wait time to be displayed to an agent. The account number is provided by the adjunct in the first 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 Communication Manager 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.

Chapter 19: Improving performance

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

Audible feedback examples

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 222.

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.

Table 11: 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 examples

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).

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.

Table 12: 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 examples

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.

Table 13: 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

After business hours example

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

The following example queues calls to a hunt group regardless of the time of the call. When 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 interflow example

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.

Chapter 20: Call Vectoring Job aids

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 Communication Manager on Avaya Media Servers. You can find the latest capacity tables from the Avaya support Website at:

http://www.avaya.com/support

For detailed information about these commands, see Call Vectoring commands.

#	[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[81] *[A-Z, AA-ZZ] V1-V9*

`announcement`	*extension no. [A-Z , AA-ZZ] V1-V9*

busy

[81] Link capacity depends on your release and configuration. For more information, see System Capacities Table for Communication Manager on Avaya Media Servers .

Check	**best**	**if**	**expected wait**	< 1-9999 seconds, > 0-9999 seconds			
			unconditionally				
			wait improved	< 1-9999 seconds, > 0-9999 seconds			
	skill	hunt group[82], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**if**	**available-agents** > 0-1499[83]	**all-levels**
							pref-level skill level 1[84]
							pref-range skill level 1[84] **to** skill level 2[84]
	skill	hunt group[82], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**if**	**calls-queued** < 1-999[83] **expected-wait** < 1-9999 seconds **oldest-call-wait** > 1-999 seconds **rolling-asa** < 1-999 seconds **staffed-agents** > 0-1499[83] **wait-improved** > 0-9999 seconds **unconditionally**	
	split	hunt group[82]					
	split	hunt group[82]	**pri**	**l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**if**	**available-agents** > 0-1499[83]	

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**[85] (multi-	*1-255 A-Z, AA-ZZ V1-V9*	**adjust by**	*0-100* percent *A-Z, AA-ZZ* *V1-V9*

[82] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[83] 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.

[84] Skill levels are 1-16 (1 is best, 16 is lowest). Skill Level 2 must be greater than or equal to Skill Level 1.

[85] This item is available only with the Virtual Routing feature.

	site BSR only)					
	skill	*hunt group*[86], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op		
	split	hunt group[86]				

converse-on	**skill**	*hunt group*[87], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l = low** **m = medium** **h = high** **t = top**	**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[87]				**none**	**and**	**none**

disconnect	after announcement	*extension no.* **none** *A-Z, AA-ZZ V1-V9*

goto step and goto vector		
goto step ***1-99*** **if** **or** **goto vector** ***1-8000***[88] **@step** ***1-99*** **if**		
A-Z, AA-ZZ	>,<,=,<>, >=,<=	threshold value or string of digits: *1-16, wildcards* (*?* , +)[89] , *[A-Z, AA-ZZ], V1-V9*
	=,<>	none[90], #[91]
	in table	1-100[88], [A-Z, AA-ZZ], V1-V9
	not-in table	

[86] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[87] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[88] 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.

[89] 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.

[90] 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.

[91] 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.

goto step and goto vector				
goto step ***1-99*** **if** **or** **goto vector** ***1-8000***[88] **@step** ***1-99*****if**				
ani	>,>=,<>,=,<,<=	1-16, wildcards (?, +)[90], [A-Z, AA-ZZ], V1-V9		
	=,<>	none[90], #[91]		
	in **table**	1-100[88], [A-Z, AA-ZZ], V1-V9		
	not-in **table**			
available - agents	**in skill**	*hunt group*[92] , skills for VDN: 1st , 2nd , 3rd	>,>=,<>, =, <,<=	*0-1499*[88] *1-1500*[88] *A-Z, AA-ZZ V1-V9*
	in split	hunt group[92]		

goto step and goto vector						
goto step ***1-99*** **if** **or** **goto vector** ***1-8000***[93] **@step** ***1-99*** **if**						
calls-queued	**in skill**	hunt group[94], skills for VDN: 1st, 2nd, 3rd	**pri**	priorities: **l =** low **m =** medium **h =** high **t =** top	> >= <> = < <=	*0-098*[93] *1-999*[93] *A-Z, AA-ZZ V1-V9*
	in split	hunt group[94]				
counted-calls	**to vdn**	vdn extension, latest, active[95]	>,>=,<>,=,<,<=		*0-998*[93] *1-999*[93] *A-Z, AA-ZZ V1-V9*	
digits	>,>=,<>,=,<,<=	threshold value or string: *1-16, wildcards* (*?, +*)[96] , *[A-Z, AA-ZZ], V1-V9*				

[92] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[93] 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.

[94] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[95] *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

[96] 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.

goto step and goto vector						
goto step ***1-99*** **if** **or** **goto vector** ***1-8000***[93] **@step** ***1-99*** **if**						
	<>,=	none[97]				
	=	meet-me-access[98]				
	in table	1-100[93], [A-Z, AA-ZZ], V1-V9				
	not-in table					
expected-wait	**for** best	>,>=,<>,=,<,<=	0-9999 seconds, [A-Z, AA-ZZ], V1-V9			
	for call					
	for split	hunt group[94]	**pri**	priorities: **l = low** **m = medium** **h = high** **t = top**	> >= <> = < <=	*0-9998* sec *1-9999* sec *A-Z, AA-ZZ V1-V9*
	for skill	*hunt group*[94], skills for VDN: 1st, 2nd, 3rd				

goto step and goto vector		
goto step ***1-99*if** **or** **goto vector** ***1-8000***[99] **@step** ***1-99*if**		
holiday	in table	1-99, [A-Z, AA-ZZ], V1-V9
	not-in table	
ii-digits	>, >=, <>, =, <, <=	2-digit string, wildcards (?, +)[100], [A-Z, AA-ZZ], V1-V9
	<>, =	none[101]

[97] 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.

[98] This item is available only with meet-me conference vectors.

[99] 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.

[100] 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.

[101] 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.

goto step and goto vector			
goto step ***1-99*****if** **or** **goto vector** ***1-8000***[99] **@step** ***1-99*****if**			
	in table	1-100[99], [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[102] *1-999*	=, <>	registered
	all		
	any		
meet-me-full[103] (goto step only)			
meet-me-idle[103] (goto step only)			
no match[104]			

goto step and goto vector						
goto step ***1-99*****if** **or** **goto vector** ***1-8000***[105] **@step** ***1-99*****if**						
oldest-call-wait	**in skill**	*hunt group*[106], skills for VDN: 1st , 2nd , 3rd	**pri**	priorities: **l** = low **m** = medium **h** = high **t** = top	>,>=,<>,=<,<=	*0-998* sec *1-999* sec *A-Z, AA-ZZ V1-V9*
	in split	hunt group[106]				

[102] 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.

[103] This item is available only with meet-me conference vectors.

[104] This item is available only with the Dial by Name feature.

[105] 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.

[106] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

goto step and goto vector				
goto step *1-99*if or goto vector *1-8000*[105] @step *1-99*if				
port-network	Port network ID[107] *1-999*		=, <>	registered
	all			
	any			
queue-fail[108]				
rolling-asa	**for skill**	*hunt group*[106], skills for VDN: 1st, 2nd, 3rd	>,>=,<>,=, <,<=	*0-998 sec, 1-999 sec A-Z, AA-ZZ V1-V9*
	for split	hunt group[106]		
	for vdn	vdn extension, latest, active[109]		
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*[106], skills for VDN: 1st , 2nd , 3rd	>,>=,<>,=, <,<=	*0-1499*[105], *1-1500*[105] *A-Z, AA-ZZ V1-V9*
	in split	hunt group[106]		

[107] 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.

[108] This item is available only with the Attendant Vectoring feature.

[109] *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

goto step and goto vector								
goto step ***1-99*****if** **or** **goto vector** ***1-8000***[110] **@step** ***1-99*****if**								
time-of-day	**is**	mon, tue, wed, thu, fri, sat, sun, all	hour: 00-23	minute: 00-59	**to**	mon, tue, wed, thu, fri, sat, sun, all	hour: 00-23	minute: 00-59
V1-V9	>, <, =,<>, >=, <, <=	threshold value or string of digits: 1-16, wildcards (?, +), [A-Z, AA-ZZ], V1-V9						
	=,<>	none[111], #[112]						
	in table	1-100[109], [A-Z, AA-ZZ], V1-V9						
	not-in table							
wait-improved for	best	>, >=, <>, =, <, <=				*0-9998* sec *1-9999* sec *A-Z, AA-ZZ* *V1-V9*		
	skill	*hunt group*[113], skills for VDN: 1st , 2nd , 3rd	**pri**	priorities: **l** = low **m** = medium **h** = high **t** = top	>,>=, <>,= <,<=			
	split	hunt group[5]						
unconditionally								

messaging	**skill**	*hunt group*[114]**1st** (VDN skill)**2nd**(VDN skill)**3rd** (VDN skill)	**for extension**	*extension no.* **latestactive**[115] *A-Z, AA-ZZ V1-V9*

[110] 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.

[111] 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.

[112] 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.

[113] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[114] 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.

[115] *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

split	hunt group[114]

queue-to	**attd-group**[116]			
	attendant[116]	extension no.		
	best			
	hunt-group[116]	group number[117]	**pri**	priorities: **l** = **low** **m** = **medium** **h** = **high** **t** = **top**
	skill	hunt group[118], VDN skills (**1st**, **2nd**, **3rd**)		
	split	hunt group[118]		

reply-best

return

route-to[119]	digits	with coverage	y, n					
	meetme[120]							
	number[121]	up to 16 digits (0-9) <digits>[A-Z, AA-ZZ, V1-V9]<digits>*<digits>A[122][123]<digits>#<digits>A	with cov	y, n	if	digit	> >= <> =< <=	*0-9#* [125]

[116] This item is available with only the Attendant Vectoring feature.

[117] A valid group number is a vector-controlled hunt group of any type (ACD, UCD, and so on).

[118] A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

[119] 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.

[120] This item is available only with meet-me conference vectors.

[121] A destination for the route-to is entered in the number field. This field can contain an administration limit of a maximum of 16 decimal digits or combination of characters and numbers that total 16. Special notations (for example, ~p) with a ~ followed by a character are counted as two digits towards the 16. The number field supports some feature activations using Feature Access Codes (FACs) alone or followed by digits including Service Observing, Remote Logout of Agent, remote access extension, attendant access number, Forced Logout/Aux and other destination numbers that can be dialed with a phone. The number field also supports vector variables (A-Z, AA-ZZ) and VDN variables (V1-V9) whose value in decimal digits is defined elsewhere before the route-to number command is to be executed.

[122] The notation means that 1 or more digits in the range of 0 to 9 can be inserted when necessary for the application. The total of digits and characters must be within the 16 digit positions total.

[123] Either a vector variable (A-Z, AA-ZZ) or VDN variable (V1-V9) can be entered at the end of any entry (digits or special character) or entered in place of ; this is shown with an “A” in the other examples. Each variable whether a single character or double character counts as two digits towards the maximum of digits in the number field. The variable can be preceded by digits as long as the total is within the 16 digit/character position limit. The variable must always be the last entry and can not be followed by a digit. Use of a variable allows having a route-to number destination address of more than 16 digits since a variable can be assigned up 16 digits during processing and will be combined with the entry in the number field.

		<digits>~p<digits>A <digits>~mA<digits>A <digits>~s<digits>A <digits>~w<digits>A <digits>~W<digits>A ~r~r+[124]				interflow-qpos	< = <=	1-9
						unconditionally		
	name1 [126]	with coverage	y, n					
	name2 [126]							
	name3 [126][125]							

`set [`*vector variable*`, Digits] = [`*operand1*`] [`*operator*`] [`*operand2*`]`

Command	Variables or digits		Operand1	Operator	Operand2
set	user-assigned[127] type A-Z or AA-ZZ vector variable	=	user-assigned[127]type A-Z or AA-ZZ vector variable	**ADD** **SUB** **MUL** **DIV** **CATL** **CATR** **MOD10** **SEL**	user-assigned[127] A-Z or AA-ZZ vector variable
	asaiuui A-Z or AA-ZZ vector variable				
	digits[128]		system-assigned[129] A-Z or AA-ZZ vector variable		system-assigned[129] A-Z or AA-ZZ vector variable
			V1-V9 VDN variable		directly-entered numeric string[130]
			digits		V1-V9 VDN variable

[124] When the specified number is preceded by ~r, Network Call Redirection (NCR) invocation is attempted back over the trunk group to the network Service Provider. The ~r sequence is counted as two digit positions toward the 16 total. The + character is a special indication for E.164 numbering required by some network Service Providers for NCR invocation over SIP trunking. The "+" character is counted as two digit positions towards the 16 total. The ~r or ~r+ entries must be in the initial digit/character positions of the number field.

[125] 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.

[126] This item is available only with the Dial by Name feature.

[127] Only global or local collect type vector variables can be assigned using the set command.

[128] The collected digits buffer holds up to 16 digits.

[129] For example, ani, asaiuui, doy, and so on.

[130] Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.

Command	Variables or digits		Operand1	Operator	Operand2
			none		**digits**
					none

```
stop
```

wait-time	0-999	**secs**	**hearing**	**music, ringback, silence, i-silent**		
	0-480[131]	**mins**		*audio source ext.*[132] *A-Z, AA-ZZ V1-V9*	**then**	**music** **ringback** **silence** **continue**[133]
	0-8[131]	**hrs**				

Vector variables job aid

For detailed information about variable types, see Variables in Vectors.

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, P, or G	Start digit position and Length	16	The for parameter of the collected digits command or assignment in the variables table

[131] This option is not available for vector administration done through Avaya Call Management System or Visual Vectors.

[132] This consists of a valid announcement or music source extension that is defined on the announcement audio sources form.

[133] 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.

Variable type	Description	Scope	Specification	Max digit length	Assigns
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

Chapter 21: Glossary

AAR See Automatic Alternate Routing (AAR) on page 429.

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) on page 430.

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) on page 434. 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) on page 429.

ACD agent See agent on page 428.

ACD work mode See work mode on page 441.

ACW See After Call Work (ACW) mode on page 428.

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 on page 428.

Adjunct Routing A means of evaluating calls before the calls are processed by requesting information from an adjunct. The Communication Manager 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) on page 429 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) on page 434 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 on page 430.

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 on page 429, manual-in work mode on page 436, and aux-work mode on page 430.

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) on page 430.

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 on page 431.

application An adjunct that requests and receives ASAI services or capabilities. One or more applications can reside on a single adjunct. However, the Communication Manager 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 Manager and to interflow calls to the Communication Manager.

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) on page 430.

ASAI See Adjunct-Switch Application Interface (ASAI) on page 428.

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 on page 429.

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 on page 429.

Audio Information Exchange (AUDIX™) An Avaya messaging system on page 437. AUDIX™ has been replaced by Message Manager.

AUDIX™ See Audio Information Exchange (AUDIX™) on page 429.

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 on page 430, manual-in work mode on page 436, and After Call Work (ACW) mode on page 428.

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) on page 429 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 on page 429, manual-in work mode on page 436, and After Call Work (ACW) mode on page 428.

available agent strategy A strategy that determines how Best Service Routing (BSR) on page 431 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) on page 428.

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 Manager that monitors the operations of an Automatic Call Distribution (ACD) on page 429 application. BCMS collects data related to the calls on the Communication Manager and organizes the data into reports that help manage ACD facilities and personnel.

BCC See Bearer Capability Class (BCC) on page 431.

BCMS See Basic Call Management System (BCMS) on page 430.

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) on page 431.

Best Service Routing (BSR) An Avaya Communication Manager feature based on call vectoring that routes Automatic Call Distribution (ACD) on page 429 calls to the split, skill, or contact center best able to service each call. BSR can be used on a single Communication Manager, or it can be used to integrate resources across a network of Communication Managers.

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 on page 430.

call appearance

a. 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.

b. 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) on page 429 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 on page 429 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.

Avaya IQ
Avaya IQ is the internal development name for Avaya's next generation reporting platform.

CDR
See Call Detail Recording (CDR) on page 431.

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

a. A circuit-switched call.

b. A communications path for transmitting voice and data.

c. 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.

d. A DS0 on a T1 or E1 facility not specifically associated with a logical circuit-switched call; analogous to a single trunk.

circuit

a. An arrangement of electrical elements through which electric current flows.

b. 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) on page 432.

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) on page 432.

CO
See Central Office (CO) on page 432.

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 on page 429.

COR See Class of Restriction (COR) on page 432.

COS See Class of Service (COS) on page 432.

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 on page 431.

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) on page 434.

EAD-MIA See Expert Agent Distribution-Most Idle Agent (EAD-MIA) on page 434.

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) on page 434.

ETN See Electronic Tandem Network (ETN) on page 434,

EWT See Expected Wait Time (EWT) on page 434.

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) on page 434, Uniform Call Distribution-Least Occupied Agent (UCD-LOA) on page 440, and Uniform Call Distribution-Most Idle Agent (UCD-MIA) on page 440.

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) on page 434, Uniform Call Distribution-Least Occupied Agent (UCD-LOA) on page 440, and Uniform Call Distribution-Most Idle Agent (UCD-MIA) on page 440.

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 on page 439.

intercept tone An tone that indicates a dialing error or denial of the service requested.

interflow An Automatic Call Distribution (ACD) on page 429 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) on page 430 measurement that is made by the system.

intraflow An Automatic Call Distribution (ACD) on page 429 term that refers to the ability for calls to redirect to other splits on the same Communication Manager 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 on page 428.

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) on page 430 feature.

manual-in work mode A mode in which an agent is ready to process another call manually. Also see, auto-in work mode on page 429, aux-work mode on page 430, and After Call Work (ACW) mode on page 428.

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) on page 437.

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 on page 440.

NSF See Network Specific Facility (NSF) on page 437.

occurrence See appearance on page 428.

pickup group A group of individuals authorized to answer any call directed to an extension within the group.

PMS See Property Management System (PMS) on page 438.

poll See status poll on page 439.

poll suppression An automatic feature of Best Service Routing (BSR) on page 431 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 on page 439.

polling, intelligent See intelligent polling on page 435.

PPN See Processor Port Network (PPN) on page 438.

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 on page 438) 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) on page 439.

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 Manager and the Basic Call Management System (BCMS) on page 430.

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 on page 429, manual-in work mode on page 436, After Call Work (ACW) mode on page 428, or aux-work mode on page 430.

Station Message Detail Recording (SMDR) An obsolete term now called Call Detail Recording (CDR) on page 431.

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) on page 431 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 on page 437.

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) on page 440.

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) on page 434, Expert Agent Distribution-Most Idle Agent (EAD-MIA) on page 434, and Uniform Call Distribution-Most Idle Agent (UCD-MIA) on page 440.

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) on page 440.

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 on page 430. To become available to receive ACD calls, the agent enters auto-in work mode on page 429 or manual-in work mode on page 436. To do work associated with a completed ACD call, an agent enters After Call Work (ACW) mode on page 428.

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) on page 430 feature monitors work states and uses this information to provide BCMS reports.

Index