Avaya Call Center

Release 3.0

Vector Enhancements for Call Vectoring and Expert Agent Selection Supplement

Release 3.0
August 2005

Notice

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

Documentation disclaimer

Avaya Inc. is not responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. Customer and/or End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation to the extent made by the Customer or End User.

Link disclaimer

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

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available through the following Web site:

http://www.avaya.com/support

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 Web site:

http://www.avaya.com/support

Trademarks

Avaya is a trademark of Avaya Inc.

All non-Avaya trademarks are the property of their respective owners.

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

Preface 7
Purpose 7
Intended users 7
Conventions and terminology 7
Reasons for reissue 8
Availability 8
Related documentation 8
Administration documents 9
Change description 10
Call Center documents 10
Documentation Web sites 10
Support 11

Displaying vector variable information 13
How to view vector variable information 13
Display fields 14
Examples 15

Advanced set command rules and applications 17
Set command syntax and valid entries 17
Arithmetic operations 18
About arithmetic operations 18
Rules and considerations for arithmetic operations 19
Invalid results for arithmetic operations 19
The length parameter in arithmetic operations 20
String operations 20
About string operations 21
Rules and considerations for CATR and CATL operations 21
Rules and considerations for SEL operations 22
Invalid results for string operations 22
Start and length parameters in string operations 23
MOD10 operations 23
About the MOD10 operation 24
Information about the MOD10 algorithm 24
Rules and considerations for MOD10 operations 24
MOD10 results 25
Invalid results for MOD10 operations 26
Start and length parameters in MOD10 operations 26
Set command considerations 26

Dial-ahead digits and the digits buffer 27
DIGITS7 message 27
Allowed assignments 27
Assigning a new value to a collect variable 27
Determining the number of digits 28
Clearing the digits buffer 28

Other considerations 29
Global variables can change during processing 29
Comparing none, # and numeric digits. 29
How comparisons worked before vector variables 30
How comparisons work now 30
Comparisons still not allowed 31

New events 33

Set command calculation examples 35
Arithmetic operation examples 35
ADD examples 35
SUB examples 36
MUL examples 37
DIV examples. 38
String operation examples 38
CATL examples 39
CATR examples 40
SEL examples 41
MOD10 operation examples. 42

Set command Call Center application examples 43
Validating numbers 43
A 19-digit credit card validation 47
Using bilingual announcements 47
Collecting an account number 48
Example 1: Without using the set command 49
Example 2: Using the set command 49
Percentage routing using VDN variables 50
Overview of tasks 50
Diagram of tasks 51
Setting up percentage routing 52

Notifying callers of queue position example 57
Scenario 57
Solution. 57
Prerequisites 57
Setting up the interflow-qpos conditional 57

Command job aid 59
Obtaining switch capacity information 59

Index 69

Preface

This section includes the following topics:

- Purpose on page 7
- Intended users on page 7
- Conventions and terminology on page 7
- Reasons for reissue on page 8
- Availability on page 8
- Related documentation on page 8
- Support on page 11

Purpose

This guide discusses supplemental information about the 3.0 Call Vectoring and Expert Agent Selection (EAS) features of Avaya Call Center. The information in this supplement was not included in the Release 3.0 *Avaya Communication Manager Call Center Software - Call Vectoring and EAS Guide* because of information that became available after this document went to production.

Intended users

This guide is intended primarily for personnel who use Call Vectoring or EAS. You should use this guide as an information source for implementing Call Vectoring or EAS. A knowledge of Automatic Call Distribution (ACD) is assumed.

Conventions and terminology

If you see any of the following safety labels in this document, take careful note of the information presented.

CAUTION:
Caution statements call attention to situations that can result in harm to software, loss of data, or an interruption in service.

WARNING:
Warning statements call attention to situations that can result in harm to hardware or equipment.

DANGER:
Danger statements call attention to situations that can result in harm to personnel.

SECURITY ALERT:
Security alert statements call attention to situations that can increase the potential for unauthorized use of a telecommunications system.

Reasons for reissue

This document was reissued to add the following information:

- New events on page 33
- Notifying callers of queue position example on page 57
- Command job aid on page 59

Availability

Copies of this document are available from the Avaya online support Web site, http://www.avayadocs.com.

Related documentation

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

- Administration documents on page 9

- Change description on page 10
- Call Center documents on page 10
- Documentation Web sites on page 10

Administration documents

The primary audience for these documents consists of communication server administrators who work for external customers and for Avaya's dealers. The satisfaction and needs of our external customers is the primary focus for the documentation.

- ***Administrator Guide for Avaya Communication Manager*** - Provides complete step-by-step procedures for administering the communication server, plus feature descriptions and reference information for administration screens and commands.
- *Avaya Communication Manager ASAI Technical Reference*, 555-230-220 - Provides detailed information regarding the Adjunct/Switch Application Interface (ASAI). Written for application designers responsible for building and programming custom applications and features.
- *Avaya Communication Manager Little Instruction Book for Basic Administration* - Provides step-by-step procedures for performing basic communication server administration tasks. Includes managing phones, managing features, and routing outgoing calls.
- *Avaya Communication Manager Little Instruction Book for Advanced Administration* - Provides step-by-step procedures for adding trunks, adding hunt groups, writing vectors and recording announcements.
- *Avaya Communication Manager Little Instruction Book for Basic Diagnostics* - Provides step-by-step procedures for baselining your system, solving common problems, reading alarms and errors, using features to troubleshoot your system, and contacting Avaya.
- *Feature Description and Implementation for Avaya Communication Manager*, 555-245-205 - Provides feature descriptions and some implementation guidance for Avaya Communication Manager.
- *Hardware Guide for Avaya Communication Manager* - Provides hardware descriptions, system parameters, lists of hardware required to use features, system configurations, and environmental requirements.
- *Overview for Avaya Communication Manager* - Provides a brief description of Avaya communication server features.
- *Reports for Avaya Communication Manager* - Provides detailed descriptions of the measurement, status, security, and recent change history reports available in the system and is intended for administrators who validate traffic reports and evaluate system performance. Includes corrective actions for potential problems.

Change description

For information about the changes made in Avaya Call Center 3.0 and Avaya CMS R13, see:

- *Avaya Call Center 3.0 and Call Management System (CMS) Release 13 Change Description*, 07-300304

Call Center documents

These documents are issued for Avaya Call Center applications. The intended audience is call center administrators.

- *Avaya Communication Manager Call Center Software - Call Vectoring and EAS Guide* - Provides information on how to write, use, and troubleshoot vectors, which are command sequences that process telephone calls in an Automatic Call Distribution (ACD) environment.
- *Avaya Communication Manager Call Center Software - Automatic Call Distribution (ACD) Guide* - Provides feature descriptions and some implementation guidance for call center features.
- *Avaya Communication Manager Call Center Software - Basic Call Management System (BCMS) Operations* - Provides information on the use of the BCMS feature for ACD reporting.

Documentation Web sites

For product documentation for all Avaya products and related documentation, go to http://www.avayadocs.com.

Use the following Web sites to view related support documentation:

- Information about Avaya products and service

  http://www.avaya.com
- Sun hardware documentation

  http://docs.sun.com
- Okidata printer documentation

  http://www.okidata.com
- Informix documentation

  http://www.informix.com

- Tivoli Storage Manager documentation

  http://www.tivoli.com

Support

Contacting Avaya technical support

Avaya provides support telephone numbers for you to report problems or ask questions about your product.

For United States support:

1- 800- 242-2121

For international support:

See the 1-800 Support Directory listings on the Avaya Web site.

Escalating a technical support issue

Avaya Global Services Escalation Management provides the means to escalate urgent service issues. For more information, see the Escalation Management listings on the Avaya Web site.

Displaying vector variable information

In previous releases, you could insert or delete vector steps while using the **change vector** form by pressing **F6** and entering an **i** or a **d** followed by the appropriate vector step. Now you can also display information about the vector variable at the bottom of the form. The **change vector** form command line prompt is changed to:

```
Edit (i/d/v):
```

This section includes the following topics:

- How to view vector variable information on page 13
- Display fields on page 14
- Examples on page 15

How to view vector variable information

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

1. While using the **change vector** form, press **esc**+**f**+**6**.

   The command line **Edit (i/d/v)** displays on the **change vector** form command line.

2. Enter a **v**, plus the variable you want to view after **Edit (i/d/v)**.

   Enter an A through Z value.

   Example: **v G**

3. Press **Enter**.

   Result:

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

    Number: 1                Name: ---------------
                                          Meet-me Conf? n          Lock? n
     Basic? y   EAS? y   G3V4 Enhanced? y   ANI/II-Digits? y   ASAI Routing?
y
 Prompting? y   LAI? y  G3V4 Adv Route? y   CINFO? y   BSR? y   Holidays? y
 Variables? y   3.0 Enhanced? y
01 goto step   1           if rolling-asa      for skill 1st      =  999
02 goto step   2           if rolling-asa      for skill 1st      =  999
03 goto vector 2  @step 1  if rolling-asa      for vdn   active   =  999
04 goto vector 3  @step 1  if rolling-asa      for vdn   latest   =  999
05 goto step   3           if time-of-day      is mon 09:00 to fri 17:00
06 set          A      = V2     MUL   V5
07 set          digits = V4     DIV   A
08 set          digits = none   ADD   none
09 set          U      = digits CATL  none
10 set          digits = digits CATR  none
11 set          digits = none   SEL   digits

                       Press 'Esc f 6' for Vector Editing

Var G: value type VALUE G L=1 ASGN=[5] VAC=VV1
```

Display fields

The following line is displayed on the bottom of the **change vector** form after you perform How to view vector variable information on page 13.

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

Display field	Description
The following four fields are always displayed.	
Var *letter*	Displays the A through Z vector variable letter you requested.
description	Displays the name of the variable. For example, `value type`.
type	Displays the vector variable type. For example, `VALUE`.
scope	Displays the defined scope as L (local) or G (global).

Display field	Description
The following items included in the brackets are displayed only if the item is applicable for the vector variable type.	
L=*length*	If Length is allowed for this variable type, this field displays the defined maximum digit length for the variable.
S=*start*	If Start is allowed for this variable type, this field displays the defined start digit position for the variable. This field does not display for the value type variable.
ASGN=*current_value*	If the variable is *not* local and the assignment is determined during call processing, this field displays the current active or latest assignment for the variable. If there is no current value, ASGN=[] is displayed.
VAC=*fac*	If the value type variable is defined, this field displays the Variable Access Code, VV1 through VV9.

Examples

Refer to the values assigned in Table A when looking at the example outputs in Table B.

Table A

Variable	Description	Type	Scope	Length	Start	Assignment	VAC
A	testing for processing time	vdntime	L				
B	digits for ani testing	collect	G	16	1	1234567890123456	
C	ASAI announce definition	asaiuui	L	1	3		
D	test with null value	collect	G	1	4		
E	total executed vector steps	stepcnt	L				
G	value type	value	G	1		5	VV1
T	time of day, military time	tod	G			1708	
V	set to active VDN for call	vdn	L			active	
W	day of week, 1=Sunday	dow	G			3	

Table A							
Variable	**Description**	**Type**	**Scope**	**Length**	**Start**	**Assignment**	**VAC**
X	caller ID	ani	L	16	1		
Y	day of year	doy	G			102	
Z	temporary value	collect	L	4	1		

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

Advanced set command rules and applications

This appendix provides detailed descriptions and examples of the set command for advanced applications.

This section includes the following topics:

- Set command syntax and valid entries on page 17
- Arithmetic operations on page 18
- String operations on page 20
- MOD10 operations on page 23
- Set command considerations on page 26

Set command syntax and valid entries

The basic syntax of the set command is:

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

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

Command	Variables or digits		Operand1	Operator	Operand2
set	*user-assigned*[1] *A-Z* vector variable	**=**	*user-assigned*[1] *A-Z* vector variable	**ADD** **SUB** **MUL** **DIV** **CATL** **CATR** **MOD10** **SEL**	*user-assigned*[1] *A-Z* vector variable
	digits[2]		system-assigned[3] *A-Z* vector variable		system-assigned[3] *A-Z* vector variable
			V1-V5 VDN variable		directly-entered numeric string[4]
			digits		*V1-V5 VDN variable*
			none		**digits**
					none

1. Only global or local collect type vector variables can be assigned using the set command.
2. The collected digits buffer holds up to 16 digits.

3. For example, ani, asaiuui, doy, and so on.
4. Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.

Arithmetic operations

The arithmetic operations are ADD, SUB, MUL, and DIV.

This section includes the following topics:

- About arithmetic operations on page 18
- Rules and considerations for arithmetic operations on page 19
- Invalid results for arithmetic operations on page 19
- The length parameter in arithmetic operations on page 20

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, **digits**	A-Z, V1-V5, **digits**, **none**	A-Z, V1-V5, **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 19) • Less than 0 = # (see Invalid results for arithmetic operations on page 19) • 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 Arithmetic operation examples on page 35.

Invalid results for arithmetic operations

During arithmetic operations, a variable or digits buffer can be assigned the # character. The # character signifies an invalid value, an overflow value, or an underflow value.

Examples: Dividing by 0 or none, results in an overflow value. Subtracting by a negative, results in an underflow value.

The # character is always a processing result. You can test the # character in a **goto** command by making the # character equivalent to the keyword used in the threshold field.

Example 1: **goto step x if A = #**

Example 2: **goto step x if A <> #**

The length parameter in arithmetic operations

The length parameter as defined for vector variables is applied only when the resulting set command integer value is assigned to a vector variable using the following rules:

- The length definition for the assigned variable specifies the maximum number of digits of the resulting set command operation value that can be assigned to the variable.
- The resulting integer digit string from the set command operation must be equal to or less than the length definition and never greater than a 10-digit integer value of 4295967295.
- If the result is greater than 4295967295 or the number of digits is greater than the length definition for the variable, the result is converted to a # character to indicate an overflow value. For example, if the definition for variable A is length = 5 and the result of the arithmetic operation is 1234567, the assignment to variable A is # indicating an invalid result.
- If the resulting string is shorter than the length definition, leading zeros are not added to the digit string to match the variable length definition.

Note:
Length is always defined as 16 digits for assignment to the digits buffer. The rules described in this section are applied for assignment to the digits buffer in the same manner as assignment to a vector variable using length = 16.

String operations

The string operations are CATL, CATR, and SEL.

This section includes the following topics:

- About string operations on page 21
- Rules and considerations for CATR and CATL operations on page 21
- Rules and considerations for SEL operations on page 22
- Invalid results for string operations on page 22
- Start and length parameters in string operations on page 23

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 22.

	Results	**Operand1**	**Operand2**
Field length	6	6	16
Entries allowed	A-Z, **digits**	A-Z, V1-V5, **digits**, **none**	A-Z, V1-V5, **digits**, **none**, numeric values 0-9999999999999999
Variable or digits buffer *contents*	• none, 0-9999999999999999 • Leading zeros allowed (007 = 007)	• none, #, 0-9999999999999999 • Leading zeros allowed (007 = 007)	
Variable or digits buffer *evaluation*	• none is a null string • 0-9999999999999999 are a string of digits • Leading zeros allowed (007 = 007) • No invalid results. See Invalid results for string operations on page 22.	• none = 0 (null string) • # = none • Leading zeros allowed (007 = 007)	
Start and length parameters	See Start and length parameters in string operations on page 23.	N/A	N/A

For more information, see String operation examples on page 38.

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 21.

	Results	Operand1	Operand2
Field length	6	6	6
Entries allowed	A-Z, **digits**	A-Z, V1-V5, **digits**, **none**	A-Z, V1-V5, **digits**, **none**, numeric values 0-999999
Variable or digits buffer *contents*	• none, 0-9999999999999999 • Leading zeros allowed (007 = 007)	• none, #, 0-9999999999999999 • Leading zeros allowed (007 = 007)	
Variable or digits buffer *evaluation*	• none is a null string • 0-9999999999999999 are a string of digits • Leading zeros retained (007 = 007) • No invalid results. See Invalid results for string operations on page 22.	• 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 23.	N/A	N/A

For more information, see String operation examples on page 38.

Invalid results for string operations

There are no invalid results for string operations. The results are either decimal digits or null (contains no digits or is set to **none**).

Start and length parameters in string operations

The start and length parameters defined for vector variables are both applied only when the resulting set command string operation value is assigned to a vector variable using the following rules:

- The start and length definition for the assigned variable specifies the portion of the resulting set command operation digit string that is selected for the assignment.
- The start definition for the assigned variable is always used to select the portion of the result string to use in the assignment. If start is defined as 1, the portion selected includes all of the left-side digits. For example, if variable S has start = 1, and the result string is 123..., the assignment to S is 123 ... When the start position is greater than 1, the left-side digits before the start position are deleted. For example, if S = 2 and the result string is 123..., the assignment to S is 23...
- The resulting string after application of the variable start position definition must be equal to or less than the length definition. If the string length is greater than the length definition for the variable, the right-side digits are deleted so that the total string length is equal to the length definition. For example, if the definition for variable S is start = 2, length = 5 and the result of the string operation is 1234567, the assignment to variable S is 23456.
- If the resulting string is shorter than the length definition, the string will not have leading zeros added to match the variable length definition.

Note:
For an assignment to the **digits** buffer, the start position is always defined as 1 and the length is always defined as 16 digits. These rules are applied for assignment to the **digits** buffer in the same manner as assignment to a vector variable using start = 1 and length = 16.

MOD10 operations

This section includes the following topics:

- About the MOD10 operation on page 24
- Information about the MOD10 algorithm on page 24
- Rules and considerations for MOD10 operations on page 24
- MOD10 results on page 25
- Invalid results for MOD10 operations on page 26
- Start and length parameters in MOD10 operations on page 26

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 28.

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, **digits**	A-Z, V1-V5, **digits**, **none**	A-Z, V1-V5, **digits**, **none**, numeric values 0-9999999999999999
Variable or digits buffer *contents*	#, 0-9	• none, #, 0-9999999999999999 • Leading zeros allowed (007 = 007)	

	Results	Operand1	Operand2
Variable or digits buffer *evaluation*	• If the numeric value of operand2 is greater than the number of digits in operand1, then the results = # • Valid modulus 10 or Luhn results = 0-9. See MOD10 results on page 25.	• 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 26.	N/A	N/A

For more information, see MOD10 operation examples on page 42.

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.

Set command considerations

This section includes the following topics:

- Dial-ahead digits and the digits buffer on page 27
- DIGITS7 message on page 27
- Allowed assignments on page 27
- Assigning a new value to a collect variable on page 27
- Determining the number of digits on page 28
- Clearing the digits buffer on page 28

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

DIGITS7 message

A DIGITS7 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 <Emphasis>Avaya CMS Reports, 585-210-929.

Allowed assignments

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

Assigning a new value to a collect variable

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

Determining the number of digits

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

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

Clearing the digits buffer

Once all necessary processing for the collected digits buffer has completed, this example describes how to use the **set** command to clear the collected digits buffer. This prevents the answering agent from seeing the contents of the digits buffer. When the agent answers the call, the Info: display will be blank.

```
1. collect 9 digits after announcement 4501 for none
2. ...
3. ... [steps 2, 3, and 4 process the collected account number]
4. ...
5. set digits = none CATR none [removes all digits in the collected    digits buffer]
6. queue-to skill 1st pri l
```

Other considerations

This section includes the following topics:

- Global variables can change during processing on page 29
- Comparing none, # and numeric digits on page 29

Global variables can change during processing

The collect global vector variable value is susceptible to being unintentionally changed and read by 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 global variables to local variables to secure a snapshot of the global values that are used for subsequent vector processing.

Also, if you are modifying the value of a global variable, it is important to complete the manipulation of the global variable within seven steps. This is due to vector operation that temporarily suspends vector processing for 0.2 seconds after processing seven 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 *Avaya Communication Manager Call Center Software - Call Vectoring and EAS Guide*, page 622.

Comparing none, # and numeric digits

This section includes the following topics:

- How comparisons worked before vector variables on page 30
- How comparisons work now on page 30
- Comparisons still not allowed on page 31

How comparisons worked before vector variables

Prior to the introduction of Communication Manager 3.0, goto comparison tests using the keywords **none** or **#** as a threshold value were supported for only the **=** or **<>** comparators. For example:

```
goto step 5 if digits = none
goto step 5 if digits <> #
```

You could not enter any other comparators with these keywords.

How comparisons work now

With vector variables and VDN variables, goto test comparisons against or containing the keywords **none** or **#** are allowed with all comparators including <, >, <=, or >=. These keywords can be compared against digit strings. When Communication Manager tests these comparisons, the keywords and digits have weighted values ordered as follows:

none < # < 0 < 1 to 9 < 00…

All comparisons are basically string comparisons, not numeric comparisons. A string comparison of 0 is less than 00, and not equal as in a numeric comparison.

With the introduction of Communication Manager 3.0, it is now possible to do less than or greater than comparisons with variables which can have a value of **none** (empty string) or **#** (invalid result or a single # digit was collected) using the ordering rules above. For example:

```
goto step 5 if digits = A
goto step 5 if digits <> A
goto step 5 if digits < A
goto step 5 if digits > A
goto step 5 if digits <= A
goto step 5 if digits => A
```

Using these properties, you can determine if a caller has entered a digit between 1 to 9 as follows:

1. collect 1 digit after hearing announcement x for A
2. goto step 1 if A <= 0 [will branch to step 1 if A has a value of none, # or 0]
3. [this step reached if A contains a digit between 1 to 9]

Comparisons still not allowed

You cannot use a comparison of the digits buffer that contains **none** or **#** against a specific numeric value that is not a variable. The goto test will always fail and fall through to the next step. For example:

```
2. goto step 1 if digits <= 0 [will branch to step 1 only if digits contains a 0]
3. …  [this step reached if digits contains none, # or a digit between 1 to 9]
```

You cannot *directly* enter **none** or **#** as a threshold value with comparators other than = or <>.

Set command calculation examples

This chapter provides **set** command examples.

This section includes the following topics:

- Arithmetic operation examples on page 35
- String operation examples on page 38
- MOD10 operation examples on page 42

Arithmetic operation examples

This section provides examples for the ADD, SUB, MUL and DIV arithmetic operators.

This section includes the following topics:

- ADD examples on page 35
- SUB examples on page 36
- MUL examples on page 37
- DIV examples on page 38

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 (#).

Command	Result
set A = A ADD B A = 000123 B = #	123 + # = # The # character in a variable is seen as invalid (#).
set A = A ADD A A = none (or null string)	0 + 0 = 0 none is interpreted as 0
set B = A ADD 456 Variable definitions: • A = 1234 • B = collect type, length = 3, start = 2 (start is ignored)	1234 + 456 = 1690 = # This is an overflow since the result was greater than three digits.
set B = A ADD 456 Variable definitions: • A = 1234 • B = collect type, length = 5, start = 2 (start is ignored)	1234 + 456 = 1690 No leading zeros are included in the result.

SUB examples

The following table provides examples for using the SUB operator. The SUB operator subtracts operand2 from operand1.

Command	Result
set A = A SUB 1 A = 000123	123 - 1 = 122 This operation is valuable for count-down variables. If an operand has preceeding zeros and is subtracted from or by another operand, the leading zeros are ignored. The resulting value does not retain the leading zeros.
set A = A SUB 0456 A = 000123	123 - 456 = -333 = # The smallest result allowed is 0. Negative values are invalid (#).
set A = A SUB 770 A = 777	777 - 770 = 7 The result is 7, not 007.
set A = A SUB 10 A = 4294967296	# - 10 = # If an operand contains a value greater than 4294967295, the result is invalid (#).

Command	Result
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.
set A = A DIV 2 A = 5	5 / 2 = 2.5 = 2 The result is rounded down to the nearest whole integer. Results do not contain any decimal values.
set A = A DIV 1000 A = 000123	123 / 1000 = 0.123 = 0 The result is rounded down to the nearest whole integer.
set A = A DIV A A = 9999999999	# / # = # This operation is an operand overflow because the operand values exceed 4294967295.
set A = A DIV # A = 000123	000123 / # = # A # character in an operand makes the operation invalid (#).
set B = A DIV 100 Variable definitions: • A = 12345678 • B: collect type, length = 4, start = 3 (start is ignored)	12345678 / 100 = 123456.78 = 123456 = # The result is rounded down to the nearest integer, but it is an invalid result (#) because it is greater than four digits.
set B = A DIV 100 Variable definitions: • A = 12345678 • B: collect type, length = 6, start = 3 (start is ignored)	12345678 / 100 = 123456.78 = 123456 The result is rounded down to the nearest integer. This result is valid because the result is six digits in length.

String operation examples

This section provides examples for the CATL, CATR, and SEL string operators.

This section includes the following topics:

- CATL examples on page 39
- CATR examples on page 40
- SEL examples on page 41

CATL examples

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

Command	Result
set A = B CATL 0123 B = 56789	56789 CATL 0123 = 012356789
set A = A CATL A A = #	# CATL # = none
set A = B CATL 0123 Variable definition: • A = collect type, length = 4, start = 3 • B = 56789	56789 CATL 0123 = 012356789 = 2356 Four digits were selected starting at position 3 in the resulting digit string.
set digits = A CATL 0123 digits = length=16, start=1 A = 123456789012345	123456789012345 CATL 0123 = 0123123456789012345 = 0123123456789012 The first 16 digits are selected and stored in the digits buffer.
set A = A CATL A A = none (null string or empty)	none CATL none = none

CATR examples

The following table provides examples for using the CATR operator. The CATR operator concatenates, or appends, the operand2 digit string to the right end of operand1.

Command	Result
set A = B CATR 0123 B = 56789	56789 CATR 0123 = 567890123
set A = A CATR A A = #	# CATR # = none
set A = B CATR 0123 Variable definition: • A = collect type, length = 4, start = 3 • B = 56789	56789 CATR 0123 = 567890123 = 7890 Four digits were selected starting at position 3 in the resulting digit string.
set digits = A CATR 0123 digits: length = 16, start = 1 A = 123456789012345	123456789012345 CATR 0123 = 01234567890123450123 = 1234567890123450 NOTE: 1234567890123450 is stored in the digits buffer.
set A = A CATR A A = none (null string or empty)	none CATR none = none

SEL examples

The following table provides examples for using the SEL operator. The SEL operator selects the right-most number of digits from operand1. The number of digits selected from operand1 is specified by operand2.

Command	Result
set A = B SEL 12 Variable definitions: • A = collect type, length = 10, start = 3 • B = 1234567890123	1234567890123 SEL 12 = 234567890123 = 4567890123 Ten digits were selected starting at position 3.
set A = B SEL 5 The number of digits specified in operand2 is more than the number of digits in operand1. B = 1234	1234 SEL 5 = 01234 The result contains all of the digits in operand1 with leading zeros as necessary.
set A = B SEL 1 Operand1 contains no digits (#) or is set to zero. B = #	# SEL 1 = 0 The result is null padded with a zero. The SEL 1 adds a leading zero.
set A = B SEL C Operand2 contains no digits or is set to zero. B = 1234 C = #	1234 SEL # = 1234 SEL 0 = none The result is none.
set A = B SEL 3 B = 120005	120005 SEL 3 = 005 The zeros are retained in the result.
set A = B SEL C B = 1234567890123456 C = 99	1234567890123456 SEL 99 = 1234567890123456 Operand2 contains a number greater than 16, therefore a maximum of 16 digits is selected from operand1.
set A = B SEL 16 A = collect type, length = 10, start = 3 B = 1234567890123	1234567890123 SEL 16 = 0123456789 Selects 10 digits starting at position 3.

MOD10 operation examples

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

Command	Result
set B = A MOD10 13 A = #	# MOD10 13 = # The operation is invalid if either operand contains a # character or **none**.
set B = digits MOD10 A A = 20	B = # The operation is invalid because operand2 contains a number greater than 16.
set B = digits MOD10 A A = none	B = # The operation is invalid.
set A = digits MOD10 5 digits = 123456	123456 MOD10 5 = # The operation is invalid because operand1 contains a digit string that has more digits than what is specified in operand2.
set A = digits MOD10 5 digits = 1234	1234 MOD10 5 = # The operation is invalid because operand1 contains a digit string that has fewer digits than what is specified in operand2.
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.

Set command Call Center application examples

This section includes the following topics:

- Validating numbers on page 43
- A 19-digit credit card validation on page 47
- Using bilingual announcements on page 47
- Collecting an account number on page 48
- Percentage routing using VDN variables on page 50

Validating numbers

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

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

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

3. Validate the first four digits of each card number prefix.

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

Company	Card number identifier	Card length
Diner's Club/Carte Blanche	300xxxxxxxxxx 305xxxxxxxxxx 36xxxxxxxxxxx 38xxxxxxxxxxx	14
American Express	34xxxxxxxxxxxx 37xxxxxxxxxxxx	15
VISA	4xxxxxxxxxxx 4xxxxxxxxxxxxxx	13, 16
MasterCard	51xxxxxxxxxxxxx 55xxxxxxxxxxxxx	16
Discover	6011xxxxxxxxxxx	16

4. Perform a Luhn or Modulus 10 check on the whole number.

   See Information about the MOD10 algorithm on page 24

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 25.

Using bilingual announcements

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

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

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

```
1.collect 1 digit after hearing announcement 3000 for A
     a. goto step 1 if A > 2
     b. route to operator if A <= 0
2.set B = A CATR 001 [B = A001, A = 1 or 2]
3.announcement B
4.set B = A CATR 002 [B = A002, A = 1 or 2]
5.collect 16 digits after hearing announcement B for none
6.queue to skill
7.set B = A CATR 003 [B = A003, A = 1 or 2]
8.wait .. hearing B then music
9.goto step 8 if unconditionally
```

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

Collecting an account number

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

Example 1: Without using the set command

The following vector shows how to collect an account number. Agents can see the account number if the call is answered by the available agent after steps 1 to 3 are executed. Agents cannot see the account number after the customer is prompted at step 4. This is because the **collect** in step 4 overwrites the digits buffer that is sent to the agent's display.

```
 1. collect 9 digits after announcement 4501 [announcement 4501 asks
    customers for their account number. Nine digits are stored in the digits
    buffer after customers enter their account number]
 2. queue-to skill 1st  pri h  [Queue the call]
 3. wait-time 30 secs hearing music [call waits, digits buffer = variable A =
    9 digits]
 4. collect 1 digits after announcement 4502 [Press 1 if customer wants to
    leave a message. The digits buffer now contains the response, overwriting
    previous collected digits.]
 5. goto step 8 if digits = 1 [Check if the caller wants to leave a message]
 6. goto step 3 if unconditionally [The caller does not want to leave a
    message, so go back to wait.]
 7. stop
 8. messaging skill 2nd for extension active [Caller leaves a message]
 9. treatment if messaging skill out of service
```

Example 2: Using the set command

The following vector shows how to use the **set** command to solve the problem presented in Example 1. The vector in this example overwrites the digits buffer with the initially-stored account number in step 1. While a call is waiting to be answered, the digits buffer is refreshed with the account number.

```
 1. collect 9 digits after announcement 4501 for A [announcement 4501 asks
    customers for their account number. Nine digits are stored in the digits
    buffer and the A variable after customers enter their account number.]
 2. queue-to skill 1st  pri h  [Queue the call]
 3. wait-time 30 secs hearing music [call waits, digits buffer = varaible A =
    9 digits]
 4. collect 1 digits after announcement 4502 for B [Press 1 if customer wants
    to leave a message. The digits buffer and variable B now contain the
    response.]
 5. set digits = A SEL 9 [reset the digits buffer to contain the original 9
    digits collected in step 1]
 6. goto step 9 if B = 1 [Check if the caller wants to leave a message]
 7. goto step 3 if unconditionally [The caller does not want to leave a
    message, so go back to wait.]
 8. stop
 9. messaging skill 2nd for extension active [Caller leaves a message]
10. treatment if messaging skill out of service
```

Percentage routing using VDN variables

This chapter provides an example of how to use percentage routing with the new VDN variables.

The XYZ Company wants to:

- Route 25% of calls to the ABC Company in India
- Route 25% of calls to the DEF Company in China
- Route 50% of calls to the XYZ Company's local call center through XYZ Company's Avaya Communication Manager system

They can use Percent Allocation to allocate call types into three groups of call handlers - one for India, one for China, and one for the local call center.

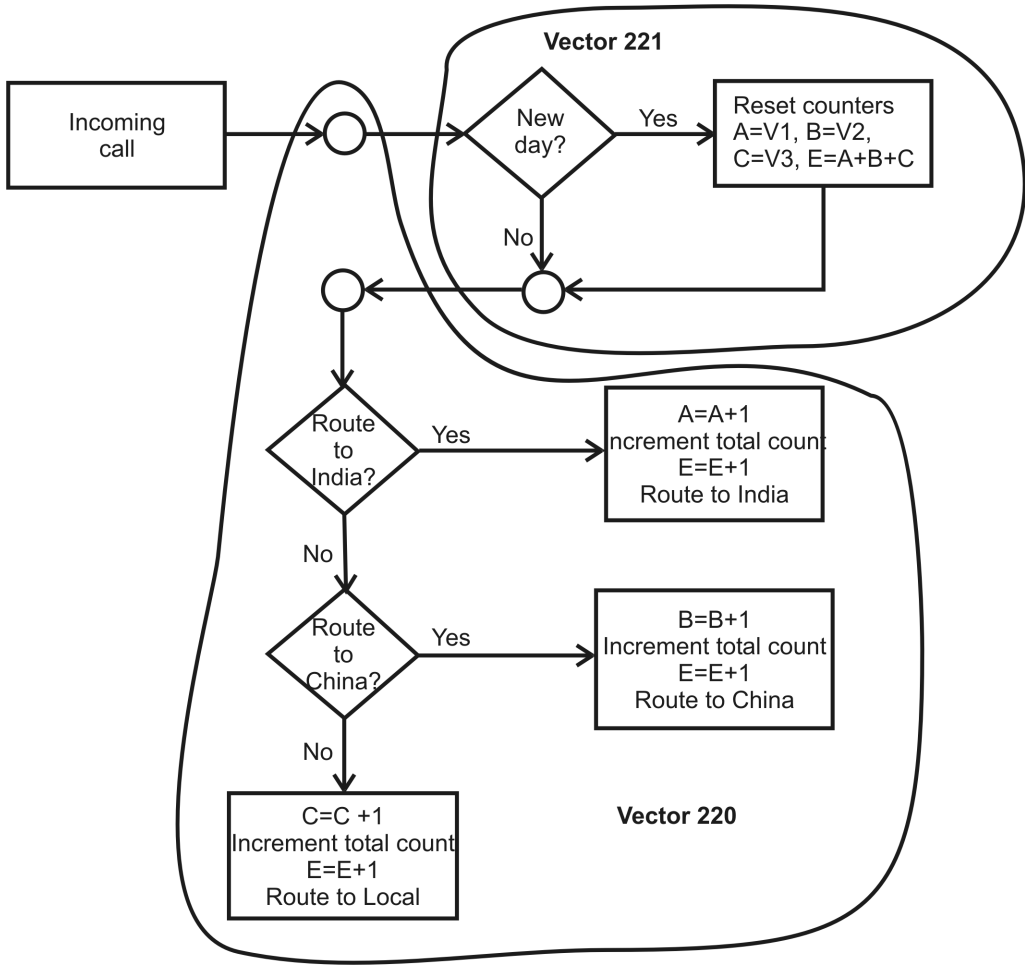
Overview of tasks

In this example, XYZ Company will use the **`set`** command and VDN variables to do the following tasks:

- Setup the Variables for Vectors and VDN variable definitions
- Use vector 220 as the primary vector used to calculate percentages and route calls accordingly
- Use vector 221 as a subroutine vector to initialize the routing for the initial call that is performed every day
- Use VDN 2220 as the VDN that calls vector 220 with assigned VDN variable values associated with Percentage Routing. In this example, the following types of calls are routed to VDN 2220:
  - Toll-free number calls
  - Direct-inward-dialing calls
  - Calls screened by Interactive Voice Response (IVR)
  - Calls transferred by a local agent

Diagram of tasks

The following flow chart provides an overview of this example.



Legend:

- New day - True when the dow variable is different from the stored dow value of the first call.
- Route to India - True when the percentage of calls to India is less than the percentage value stored in the VDN variable V1=25%. Variable A (initialized to V1) is the count of calls routed to India. This value is incremented before the call is routed. The total count of variable E is also incremented.
- Route to China - True when the percentage of calls to China is less than the percentage value stored in the VDN variable V2=25%. Variable B (initialized to V2) is the count of calls routed to China. This value is incremented before the call is routed. The total count of variable E is also incremented.

- Percentage of calls
  - India = (A/A+B+C) * 100
  - China = (B/A+B+C) * 100
  - Local = (C/A+B+C) * 100

Setting up percentage routing

To set up percentage routing:

1. Define the Variables for Vectors using the Variables for Vectors form.

   Example:

```
change variables                                                    Page 1 of x

                                Variables for Vectors

 Var  Description                 Type    Scope Length Start Assignment       VAC
 A    India Routed Calls          collect   G     16     1
 B    China Routed Calls          collect   G     16     1
 C    Locally Retained Calls      collect   G     16     1
 D    Calculated Percentage       collect   L     16     1
 E    Total Routed Calls Today    collect   G     16     1
 F    Day-of-Week for First Call  collect   G     16     1
 G    Day-of-Week (1=Sun..etc.)     dow     G     16     1        2
```

2. Set up the VDN you want used as the called VDN. This VDN calls the vector with the assigned VDN variable values associated with Percentage Routing. In the following example, the VDN is 2220 and the vector number is 220.

   Example:

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

                               Extension: 2220
                                    Name: Percent Routing
                          Vector Number: 220


                     Allow VDN Override? n
                                   COR: 1
                                    TN: 1
                              Measured: external


        VDN of Origin Annc. Extension:
                            1st Skill:
                            2nd Skill:
                            3rd Skill:
```

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

                    Return Destination:
                VDN Timed ACW Interval:
                       BSR Application:15
          BSR Available Agent Strategy: 1st -found

                   Observe on Agent Answer?:n

               Display VDN for Route-to DAC?:n
VDN Override for ISDN Trunk ASAI Messages?:n

                       BSR Local Treatment?:n
```

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

   Example:

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

                                  VDN VARIABLES

                          Var          Description          Assignment

                          V1            India                   25
                          V2            China                   25
                          V3            Local                   50
                          V4
                          V5

                         VDN Time Zone Offset + 00:00
```

4. Use the Call Vector form to set up a primary vector that calculates percentages and routes calls accordingly. This is the main vector for processing calls placed to VDN 2220.

   Example:

```
change vector 220                                          Page 1 of 3
CALL VECTOR

             Number: 220         Name: Percent Route
Multimedia? n                                                    Lock? n
     Basic? y  EAS? y   G3V4 Enhanced? y    ANI/II-Digits? y     ASAI Routing? y
 Prompting? y  LAI? y   G3V4 Adv Route? y   CINFO? y   BSR? y        Holidays? y
 Variables? y  3.0 Enhanced? y
01 goto vector 221 @ step 1 if unconditionally
02 set D = A MUL 100 [D = calculated %, A = India routed calls]
03 set D = D DIV E [D = calculated %, E = total routed calls today]
04 goto step 11 if D < V1 [D = calculated %, V1 = India with an assignment of 25]
05 set D = B MUL 100 [D = calculated %, B = China routed calls]
06 set D = D DIV E [D = calculated %, E = total routed calls today]
07 goto step 15 if D < V2 [D = calculated %, V2 = China with an assignement of 25]
08 set D = C MUL 100 [D = calculated %, C = local routed calls]
09 set D = D DIV E [D = calculated %, E = total routed calls today]
10 goto step = 19 if unconditionally
11 set A = A ADD 1 [A = India routed calls]
12 set E = E ADD 1 [E = total routed calls today]
13 route-to number 97051001 with cov n if unconditionally
14 stop
15 set B = B ADD 1 [B = China routed calls]
16 set E = E ADD 1 [E = total routed calls today]
17 route-to number 97051002 with cov n if unconditionally
18 stop
19 set C = C ADD 1 [C = local routed calls]
20 set E = E ADD 1 [E = total routed calls today]
21 route-to number 97051003 with cov n if unconditionally
```

5. Set up a subroutine vector to initialize the routing every day for the first call. This subroutine is called by step 1 in vector 220.

   Example:

```
change vector 221                                          Page 1 of 3
CALL VECTOR

             Number: 221         Name: Reset New Day
Multimedia? n                                                    Lock? n
     Basic? y  EAS? y   G3V4 Enhanced? y    ANI/II-Digits? y     ASAI Routing? y
 Prompting? y  LAI? y   G3V4 Adv Route? y   CINFO? y   BSR? y        Holidays? y
 Variables? y  3.0 Enhanced? y
01 goto vector step 8 if F = G [F = day of week for first call, G = day of week[1]]
02 set F = none ADD G [F = day of week for first call, G = day of week[1]]
03 set A = none ADD V1 [A = India routed calls, V1 = India with an assignment of
25]
04 set B = none ADD V2 [B = China routed calls, V2 = China with an assignment of
25]
05 set C = none ADD V3 [C = local routed calls, V3 = local with an assignment of
50]
06 set D = A ADD B [D = total routed calls today, A = India routed calls, B =
China routed calls]
07 set E = D ADD C [E = total routed calls today, C = local routed calls]
08 return
```

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

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

Notifying callers of queue position example

This example explains how to notify callers of their position in queue without using a VRU or an IVR system.

Scenario

The XYZ call center has the following requirements:

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

Solution

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

Prerequisites

Before using the interflow-qpos conditional, consider the following prerequisites:

- Virtual Routing (LAI) must be active.
- Set the **Interflow-Qpos EWT Threshold** field on the feature related system parameter form 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.

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 1 > 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 1
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
```

Command 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.

Obtaining switch capacity information

Most vector commands require one or more input values for the command, as well as for various parameters, such as an announcement extension number, a time interval, a maximum queue size, and so forth. When the minimum and maximum ranges for command parameter values are identical for all Avaya switch platforms, the limiting ranges are specified in the job aid. Alternately, when the minimum and maximum ranges for a parameter value are not the same among Avaya switch platforms, the upper limit of a value range is indicated by the term *switch max*.

To determine the maximum values you can use in Call Vectoring commands, see *System Capacities Table for Avaya Communication Manager on Avaya Media Servers*. You can find the latest capacity tables from the Avaya support Website at:

http://www.avayadocs.com

adjunct routing link	*1-16* - CTI Link ID

announcement	*extension no., A-Z, V1-V5*

busy

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

1. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
2. The maximum limit is less on some platforms. Use the help key for your switch administration software to determine the applicable limit for your system.

collect	**ced**	**for**	**none**, *A-Z*		
	cdpd				
	1-16	**digits after announcement**	*extension no.*, **none**, *A-Z*, *V1-V5*	**for**	**none**, *A-Z*

consider	**location**[1]	*1-255* (multi-site BSR only)			**adjust by**	*0-100* percent
	skill	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op		
	split	*hunt group*[2]				

1. This item is available only with the Virtual Routing feature.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

converse-on	**skill**	*hunt group*[1], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op	**passing**	*6-digit string*, *****, **#**, **ani**, **vdn**, **digits**, **qpos**, **wait**, *A-Z*, *V1-V5*	**and**	*6-digit string*, *****, **#**, **ani**, **vdn**, **none**, **digits**, **qpos**, **wait**, *A-Z*, *V1-V5*
	split	*hunt group*[1]				**none**	**and**	**none**

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

disconnect	**after announcement**	*extension no.*, **none**, *A-Z, V1-V5*

goto step and goto vector				
goto step ***1-32*** **if** **or** **goto vector** ***1-999***[1] **@step** ***1-32*** **if**				
A-Z	**>**,**<**,**=**,**<>**,**>=**,**<=**	threshold value or string of digits: *1-16, wildcards (?, +)*[2]*, A-Z, V1-V5*		
	=,**<>**	**none**[3], **#**[4]		
	in table	*1-100*[1]*, A-Z, V1-V5*		
	not-in table			
ani	**>**,**>=**,**<>**,**=**,**<**,**<=**	*1-16, wildcards (?, +)*[3]*, A-Z, V1-V5*		
	=,**<>**	**none**[3], **#**[4]		
	in table	*1-100*[1]*, A-Z, V1-V5*		
	not-in table			
available-agents	**in skill**	*hunt group*[5], skills for VDN: **1st**, **2nd**, **3rd**	**>**,**>=**,**<>**,**=**,**<**,**<=**	*0-1499*[1], *1-1500*[1], *A-Z*, *V1-V5*
	in split	*hunt group*[5]		

1. The maximum limit is less on some platforms. Use the help key for your switch administration software to determine the applicable limit for your system.
2. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (**+**) matches any or no characters at the specified position.
3. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
4. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
5. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

<table>
<tr><th colspan="7">goto step and goto vector (cont.)</th></tr>
<tr><td colspan="7">goto step 1-32 if
or
goto vector 1-999[1] @step 1-32 if</td></tr>
<tr><td rowspan="2">calls-queued</td><td>in skill</td><td><i>hunt group</i>[2],
skills for VDN:
1st, 2nd, 3rd</td><td rowspan="2">pri</td><td rowspan="2">priorities:
l = low
m = medium
h = high
t = top</td><td rowspan="2">>,>=,<>
,=,
<,<=</td><td rowspan="2"><i>0-098</i>[1],
<i>1-999</i>[1],
<i>A-Z, V1-V5</i></td></tr>
<tr><td>in split</td><td>hunt group[2]</td></tr>
<tr><td>counted-calls</td><td>to vdn</td><td><i>vdn extension</i>,
latest, active[3]</td><td colspan="3">>,>=,<>,=,
<,<=</td><td><i>0-998</i>[1]
<i>1-999</i>[1]
<i>A-Z</i>
<i>V1-V5</i></td></tr>
<tr><td rowspan="5">digits</td><td>>,>=,<>,=,<,<=</td><td colspan="5">threshold value or string:
<i>1-16, wildcards (?, +)</i>[4]<i>, A-Z, V1-V5</i></td></tr>
<tr><td><>,=</td><td colspan="5">none[5]</td></tr>
<tr><td>=</td><td colspan="5">meet-me-access[6]</td></tr>
<tr><td>in table</td><td colspan="5" rowspan="2">1-100[1], A-Z, V1-V5</td></tr>
<tr><td>not-in table</td></tr>
<tr><td rowspan="4">expected-wait</td><td>for best</td><td rowspan="2">>,>=,<>,=,<,<=</td><td colspan="4" rowspan="2"><i>0-9999</i> seconds, <i>A-Z, V1-V5</i></td></tr>
<tr><td>for call</td></tr>
<tr><td>for split</td><td><i>hunt group</i>[2]</td><td rowspan="2">pri</td><td rowspan="2">priorities:
l = low
m = medium
h = high
t = top</td><td rowspan="2">>,>=,<>
,=,<,<=</td><td rowspan="2"><i>0-9998</i> sec
<i>1-9999</i> sec
<i>A-Z</i>
<i>V1-V5</i></td></tr>
<tr><td>for skill</td><td>hunt group[2],
skills for VDN:
1st, 2nd, 3rd</td></tr>
</table>

1. The maximum limit is less on some platforms. Use the help key for your switch administration software to determine the applicable limit for your system.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
3. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.
4. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (**+**) matches any or no characters at the specified position.
5. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
6. This item is available only with meet-me conference vectors.

goto step and goto vector (*cont.*)			
goto step *1-32* if **or** **goto vector *1-999*[1] @step *1-32* if**			
holiday	**in table**	*1-99, A-Z, V1-V5*	
	not-in table		
ii-digits	**>,>=,<>,=,<, <=**	*2-digit string*, wildcards (*?, +*)[2], *A-Z, V1-V5*	
	<>,=	**none**[3]	
	in table	*1-100*[1], *A-Z, V1-V5*	
	not-in table		
interflow-gpos	**>,>=,<>,=,<, <=**	*1-9, A-Z, V1-V5*	
media-gateway	H.248 gateway ID[4] *1-999*	**=, <>**	**registered**
	all		
	any		
meet-me-full[5] (**goto step** only)			
meet-me-idle[5] (**goto step** only)			
no match[6]			

1. The maximum limit is less on some platforms. Use the help key for your switch administration software to determine the applicable limit for your system.
2. The question mark (?) is a wild card that matches any digit (0-9) at the specified position. The plus sign (**+**) matches any or no characters at the specified position.
3. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
4. The maximum number of port networks and media-gateways supported varies with the server platform. For example, the S8710 server supports up to 64 port networks and 250 media gateways. Check capacity tables for supported limits.
5. This item is available only with meet-me conference vectors.
6. This item is available only with the Dial by Name feature.

goto step and goto vector (*cont.*)						
goto step *1-32* **if** **or** **goto vector** *1-999*[1] **@step** *1-32* **if**						
oldest-call-wait	**in skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**	**pri**	priorities: **l** = **l**ow **m** = **m**edium **h** = **h**igh **t** = **t**op	**>**,**>=**,**<>**, **=<**,**<=**	*0-998* sec, *1-999* sec, *A-Z, V1-V5*
	in split	*hunt group*[2]				
port-network	Port network ID[3] *1-999*		**=**, **<>**	**registered**		
	all					
	any					
queue-fail[4]						
rolling-asa	**for skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**	**>**,**>=**,**<>**,**=**, **<**,**<=**		*0-998 sec'*, *1-999 sec,* *A-Z,* *V1-V5*	
	for split	*hunt group*[2]				
	for vdn	*vdn extension*, **latest**, **active**[5]				
server	**=**, **<>**	**main**, **ess**, **lsp**				
staffed-agents	**in skill**	*hunt group*[2], skills for VDN: **1st**, **2nd**, **3rd**	**>**,**>=**,**<>**,**=**, **<**,**<=**		*0-1499*[1], *1-1500*[1] *A-Z,* *V1-V5*	
	in split	*hunt group*[2]				

1. The maximum limit is less on some platforms. Use the help key for your switch administration software to determine the applicable limit for your system.
2. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.
3. The maximum number of port networks and media-gateways supported varies with the server platform. For example, the S8710 server supports up to 64 port networks and 250 media gateways. Check capacity tables for supported limits.
4. This item is available only with the Attendant Vectoring feature.
5. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

<table>
<tr><td colspan="9">goto step and goto vector (cont.)</td></tr>
<tr><td colspan="9">goto step 1-32 if
or
goto vector 1-999[1] @step 1-32 if</td></tr>
<tr><td>time-of-day</td><td>is</td><td>mon, tue, wed, thu, fri, sat, sun, all</td><td>hour:
00-23</td><td>minute:
00-59</td><td>to</td><td>mon, tue, wed, thu, fri, sat, sun, all</td><td>hour:
00-23</td><td>minute:
00-59</td></tr>
<tr><td rowspan="4">V1-V5</td><td>>, <, =,<>, >=, <, <=</td><td colspan="7">threshold value or string of digits: <i>1-16</i>, wildcards (<i>?</i>, <i>+</i>), <i>A-Z</i>, <i>V1-V5</i></td></tr>
<tr><td>=, <></td><td colspan="7">none[2], #[3]</td></tr>
<tr><td>in table</td><td colspan="7" rowspan="2">1-100[1], A-Z, V1-V5</td></tr>
<tr><td>not-in table</td></tr>
<tr><td rowspan="2">wait-improved for</td><td>best</td><td colspan="4">>, >=, <>, =, <, <=</td><td colspan="2" rowspan="2"><i>0-9998</i> sec,
<i>1-9999</i> sec,
<i>A-Z</i>,
<i>V1-V5</i></td></tr>
<tr><td>skill</td><td><i>hunt group</i>[4], skills for VDN:
1st, 2nd, 3rd</td><td>pri</td><td>priorities:
l = low
m = medium
h = high
t = top</td><td>>,>=,
<>,=<,
<=</td></tr>
<tr><td>split</td><td>hunt group[5]</td></tr>
<tr><td colspan="9">unconditionally</td></tr>
</table>

1. The maximum limit is less on some platforms. Use the help key for your switch administration software to determine the applicable limit for your system.
2. Use the word **none** in the threshold field to test for an empty digits string. Only the = or the <> comparators are valid in this case.
3. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
4. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

messaging	**skill**	*hunt group*[1]	**for extension**	*extension no.*
		1st (VDN skill)		**latest**[2]
		2nd (VDN skill)		**active**[2]
		3rd (VDN skill)		
	split	*hunt group*[1]		

1. A valid hunt group is an ACD split or skill or a non-ACD hunt group assigned for AUDIX, remote AUDIX, MSA, or QSIG MWI on the hunt group.
2. *Active* refers to the VDN specified by VDN Override settings. *Latest* refers to the VDN specified for the current vector.

queue-to	**attd-group**[1]			
	attendant[1]	*extension no.*		
	best			
	hunt-group[1]	*group number*[2]	**pri**	priorities: **l = l**ow **m = m**edium **h = h**igh **t = t**op
	skill	*hunt group*[3], VDN skills (**1st**, **2nd**, **3rd**)		
	split	*hunt group*[3]		

1. This item is available with only the Attendant Vectoring feature.
2. A valid group number is a vector-controlled hunt group of any type (ACD, UCD, and so on).
3. A valid hunt group is a vector-controlled ACD split or skill assigned on a hunt group form.

reply-best

return

route-to	**digits**	**with coverage**	**y**, **n** y = yes n = no					
	meetme[1]							
	number	*0-9*, *****, **#**, **~p**, **~m**, **~s**, **~w**, **~W**, *A-Z*,*V1-V5*, **~r**<*up to 14-digit number*>[2], **~r[***V1-V5***]**[2], **~r[***A-Z***]**[2]	**with cov**	**y**, **n** y = yes n = no	**if**	**digit**	**>**, **>=**, **<>**, **=<**, **<=**	*0-9*, **#**[3]
						interflow-qpos	**<**, **=**, **<=**	*1-9*
						unconditionally		
	name1[4]	**with coverage**	**y**, **n** y = yes n = no					
	name2[4]							
	name3[4]							

1. This item is available only with meet-me conference vectors.
2. When the specified number is preceeded by ~r, Network Call Redirection is attempted.
3. The # character is used in the threshold field to match a single # digit entered by the caller or an ASAI adjunct in the dial-ahead buffer. In this case, only the = or <> comparators are valid.
4. This item is available only with the Dial by Name feature.

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

Command	Variables or digits		Operand1	Operator	Operand2
set	*user-assigned*[1] *A-Z* vector variable	**=**	*user-assigned*[1] *A-Z* vector variable	**ADD** **SUB** **MUL** **DIV** **CATL** **CATR** **MOD10** **SEL**	*user-assigned*[1] *A-Z* vector variable
	digits[2]		system-assigned[3] *A-Z* vector variable		system-assigned[3] *A-Z* vector variable
			V1-V5 VDN variable		directly-entered numeric string[4]
			digits		*V1-V5 VDN variable*
			none		**digits**
					none

1. Only global or local collect type vector variables can be assigned using the set command.
2. The collected digits buffer holds up to 16 digits.
3. For example, ani, asaiuui, doy, and so on.
4. Limited to 4294967295 with ADD, SUB, MUL, or DIV. For all other operators, the limit is 16 digits.

stop

wait-time	*0-999*	**secs**	**hearing**	**music**, **ringback**, **silence**, **i-silent**		
	0-480[1]	**mins**		*audio source ext.*[2]*, A-Z, V1-V5*	**then**	**music** **ringback** **silence** **continue**[3]
	0-8[1]	**hrs**				

1. This option is not available for vector administration done through Avaya Call Management System or Visual Vectors.
2. This consists of a valid announcement or music source extension that is defined on the announcement audio sources form.
3. The continue treatment is valid only with Multiple Audio/Music Sources. It indicates that the caller continues to hear the alternate audio or music source using an announcement until another vector command takes effect.

Index

Symbols

character 19, 20, 24, 25
comparisons 29

A

account number collection example 48
algorithms 24
arithmetic operations
  examples 35
  invalid results 19
  rules 18
  start and length 20

B

bilingual announcements example 47

C

change vector form 14
comparisons using none, # and numeric digits 29
credit card numbers 44

D

determining the number of digits example 28
dial-ahead digits and the digits buffer 27
digits buffer clearing example 28

G

global variable change 29

H

helplines 11

I

invalid character 19, 20, 24, 25

L

LUHN-10 algorithm 24

M

MOD10 algorithm 24
MOD10 operations
  examples 42
  invalid results 26
  rules 23
  start and length 26

N

none comparisons 29
numeric digits comparisons 29

P

percentage routing using VDN variables example . . . 50

S

SEL operations 22
string operations
  examples 38
  rules 20
  start and length 23

V

validating numbers by matching segments example . . 47
vector command
  parameters 59
  syntax 59
Vector Directory Number form
  screen–add/change 53, 54
vector variable information, viewing 13
viewing vector variable information 13