

Temporary Variables

This section is equivalent to the experimental space defined by the Structure of Management Information (SMI). It contains variables that are useful to have, but are beyond the ability of Cisco to control and maintain. Support for these variables can change with each Cisco IOS software release.

Temporary Variables Available

The temporary variables section includes the following groups of variables:

- AppleTalk Group
- Chassis Group
- DECnet Group
- Novell Group
- Virtual Integrated Network Service (VINES) Group

The variables in this group have been deprecated and replaced with the Cisco VINES (cv) group, found in the ciscoMgmt tree.

- Xerox Network Systems (XNS) Group
- Public SNMP Traps Supported by Cisco
- SNMP Traps Defined by Cisco
- Variables Supported in RFC 1285

AppleTalk Group

Variables in this group can be used with all Cisco products running the AppleTalk protocol. These variables provide such information as total number of input and output packets, number of packets with errors, and number of packets with Address Resolution Protocol (ARP) requests and replies.

atArprobe

Indicates the total number of input ARP probe packets.

Syntax: Integer

Access: Read-only

atArreply

Indicates the total number of AppleTalk ARP reply packets output.

Syntax: Integer

Access: Read-only

atArreq

Indicates the total number of input AppleTalk ARP request packets.

Syntax: Integer

Access: Read-only

atAtp

Indicates the total number of AppleTalk ATP packets received.

Syntax: Integer

Access: Read-only

atBcastin

Indicates the total number of AppleTalk input broadcast packets.

Syntax: Integer

Access: Read-only

atBcastout

Indicates the total number of AppleTalk output broadcast packets.

Syntax: Integer

Access: Read-only

atChksum

Indicates the total number of AppleTalk input packets with checksum errors.

Syntax: Integer

Access: Read-only

atDdpbad

Indicates the total number of illegal-sized AppleTalk Datagram Delivery Protocol (DDP) packets received.

Syntax: Integer

Access: Read-only

atDdplong

Indicates the total number of long DDP packets received.

Syntax: Integer

Access: Read-only

atDdpshort

Indicates the total number of short DDP packets received.

Syntax: Integer

Access: Read-only

atEcho

Indicates the total number of AppleTalk echo packets received.

Syntax: Integer

Access: Read-only

atEchoill

Indicates the total number of illegal AppleTalk echo packets received.

Syntax: Integer

Access: Read-only

atForward

Indicates the total number of AppleTalk packets forwarded.

Syntax: Integer

Access: Read-only

atHopcnt

Indicates the total number of AppleTalk input packets that have exceeded the maximum hop count.

Syntax: Integer

Access: Read-only

atInmult

Indicates the total number of AppleTalk input packets with multicast addresses.

Syntax: Integer

Access: Read-only

atInput

Indicates the total number of input AppleTalk packets.

Syntax: Integer

Access: Read-only

atLocal

Indicates the total number of AppleTalk input packets for this host.

Syntax: Integer

Access: Read-only

atNbpin

Indicates the total number of AppleTalk Name Binding Protocol (NBP) packets received.

Syntax: Integer

Access: Read-only

atNbpout

Indicates the total number of NBP packets sent.

Syntax: Integer

Access: Read-only

AppleTalk Group

atNoaccess

Indicates the total number of AppleTalk packets dropped due to access control.

Syntax: Integer

Access: Read-only

atNobuffer

Indicates the total number of AppleTalk packets lost due to no memory.

Syntax: Integer

Access: Read-only

atNoencap

Indicates the total number of AppleTalk packets that were dropped because they could not be encapsulated.

Syntax: Integer

Access: Read-only

atNoroute

Indicates the total number of number of AppleTalk packets dropped because the router did not know where to forward them.

Syntax: Integer

Access: Read-only

atNotgate

Indicates the total number of AppleTalk input packets received while AppleTalk routing was not enabled.

Syntax: Integer

Access: Read-only

atOutput

Indicates the total number of AppleTalk output packets.

Syntax: Integer

Access: Read-only

atRtmpin

Indicates the total number of AppleTalk Routing Table Maintenance Protocol (RTMP) packets received.

Syntax: Integer

Access: Read-only

atRtmpout

Indicates the total number of RTMP packets sent.

Syntax: Integer

Access: Read-only

atUnknown

Indicates the total number of unknown AppleTalk input packets.

Syntax: Integer

Access: Read-only

Chassis Group

atZipin

Indicates the total number of AppleTalk Zone Information Protocol (ZIP) packets received.

Syntax: Integer

Access: Read-only

atZipout

Indicates the total number of ZIP packets sent.

Syntax: Integer

Access: Read-only

Chassis Group

Variables in this group apply to the Cisco chassis and provide information about the hardware within the chassis such as the system software version of the read-only memory (ROM) and the type of chassis (Cisco 2000, Cisco 3000, and so on).

There are two tables in this group, the Chassis Card table and the Chassis Card Interface Index table.

The Chassis Card table itemizes the components contained within a chassis. The Card table includes the following variables: *cardIndex*, *cardType*, *cardDescr*, *cardSerial*, *cardHwVersion*, *cardSwVersion*, *cardSlotNumber*, *cardContainedByIndex*, *cardOperStatus*, and *cardSlots*. The index to this table is *cardIndex*. If the device has *n* number of cards, the table will contain *n* number of rows.

The Chassis Card Interface table provides a logical mapping between the interface index in the ifTable (MIB II) and the interface on a given card in the Chassis. The variables in this table support only the Cisco 4000, Cisco 4500, Cisco 7000, and Cisco 7500 product lines. The Card Interface Index table is made up of rows, where each row contains the following MIB objects: *cardIfIndex*, *cardIfSlotNumber*, *cardIfPortNumber*, *cardIfCardIndex*, and *cardIfConnectorTypeEnabled*.

chassisType

Indicates the type of chassis for the product. For example, c4000 indicates a Cisco 4000 router.

The following are integer values for this variable:

- 1 = Unknown
- 2 = Multibus (for example, CGS or ASM)
- 3 = AGS+
- 4 = IGS
- 5 = c2000
- 6 = c3000
- 7 = c4000
- 8 = c7000
- 9 = cs500
- 10 = c7010
- 11 = c2500
- 12 = c4500
- 13 = c2102
- 14 = c2202
- 15 = c2501
- 16 = c2502
- 17 = c2503
- 18 = c2504
- 19 = c2505
- 20 = c2506
- 21 = c2507
- 22 = c2508
- 23 = c2509
- 24 = c2510
- 25 = c2511
- 26 = c2512
- 27 = c2513
- 28 = c2514
- 29 = c2515
- 30 = c3101
- 31 = c3102
- 32 = c3103

Chassis Group

33 = c3104
34 = c3202
35 = c3204
36 = accessProRC
37 = accessProEC
38 = c1000
39 = c1003
40 = c1004
41 = c2516
42 = c7507
43 = c7513
44 = c7506
45 = c7505
46 = c1005
47 = c4700
48 = c2517
49 = c2518
50 = c2519
51 = c2520
52 = c2521
53 = c2522
54 = c2523
55 = c2524
56 = c2525
57 = c4700S
58 = c7206
60 = as5200
65 = c7204

Syntax: Integer

Access: Read-only

chassisVersion

Provides the chassis hardware revision level, or an empty string if the information is unavailable. Examples of the types of chassis versions are D or AO.

Syntax: Display string

Access: Read-only

chassisId

Provides the unique ID number for the chassis. This number contains the value of the CPU serial number or ID number (if available); otherwise, it will be empty. This number also can be set with *snmp-server chassis-id*. An example of a value for a CPU serial number is 00160917.

Syntax: Display string

Access: Read-write

romVersion

Provides the ROM system software version, or an empty string if unavailable. Following is an example of the type of information provided by the *romVersion* variable:

```
System Bootstrap, Version 4.5(3), SOFTWARE [fc1]
Copyright (c) 1986-1992 by cisco Systems
```

Syntax: Display string

Access: Read-only

romSysVersion

Provides the software version of the system software in ROM, or an empty string if the information is unavailable. Following is an example of the type of information provided by the *romSysVersion* variable:

```
GS Software (GS3), Version 10.2(3127) [jdoe 106]
Copyright (c) 1986-1993 by cisco Systems, Inc.
Compiled Thu 08-Apr-93 09:55
```

Chassis Group

Syntax: Display string

Access: Read-only

processorRam

Provides the bytes of RAM available to the CPU of the device.

Syntax: Integer

Access: Read-only

nvRAMSize

Provides the nonvolatile configuration memory in bytes.

Syntax: Integer

Access: Read-only

nvRAMUsed

Provides the number of bytes of nonvolatile configuration memory in use.

Syntax: Integer

Access: Read-only

configRegister

Indicates the value of the configuration register.

Syntax: Integer

Access: Read-only

configRegNext

Indicates the value of the configuration register at next reload.

Syntax: Integer

Access: Read-only

Chassis Card Table

cardIndex

Provides index into card table (not physical chassis slot number).

Syntax: Integer

Access: Read-only

cardType

Provides information that identifies the functional type of card.

The possible integer values follow:

- 1 = unknown
- 2 = csc1
- 3 = csc2
- 4 = csc3
- 5 = csc4
- 6 = rp
- 7 = cpu-igs
- 8 = cpu-2500
- 9 = cpu-3000
- 10 = cpu-3100
- 11 = cpu-accessPro
- 12 = cpu-4000
- 13 = cpu-4000m
- 14 = cpu-4500

Chassis Group

15 = rsp1
16 = rsp2
17 = cpu-4500m
18 = cpu01003
19 = cpu-4700
20 = csc-m
21 = csc-mt
22 = csc-mc
23 = csc-mcplus
24 = csc-envm
25 = chassisInterface
26 = cpu-4700S
27 = cpu-7200
28 = rsp7000
29 = chassisInterface7000
32 = cpu-as5200
33 = c7200-io1fe
34 = cpu-4700m
36 = c7200-io
40 = csc-16
41 = csc-p
50 = csc-a
51 = csc-e1
52 = csc-e2
53 = csc-y
54 = csc-s
55 = csc-t
56 = sci4s
57 = sci2s2t
58 = sci4t
59 = mci1t
60 = mci2t
61 = mci1s
62 = mci1s1t
63 = mci2s
64 = mci1e
65 = mci1e1t

66 = mci1e2t
67 = mci1e1s
68 = mci1e1s1t
69 = mci1e2s
70 = mci2e
71 = mci2e1t
72 = mci2e2t
73 = mci2e1s
74 = mci2e1s1t
75 = mci2e2s
80 = csc-r
81 = csc-r16
82 = csc-r16m
83 = csc-1r
84 = csc-2r
100 = csc-cct1
101 = csc-cct2
110 = csc-mec2
111 = csc-mec4
112 = csc-mec6
113 = csc-fci
114 = csc-fcit
115 = csc-hsci
116 = csc-ctr
150 = sp
151 = eip
152 = fip
153 = hip
154 = sip
155 = trip
156 = fsip
157 = aip
158 = mip
159 = ssp
160 = cip
161 = srs-fip
162 = srs-trip

Chassis Group

163 = feip
164 = rvip
166 = ssip
167 = smip
168 = posip
200 = npm-4000-fddi-sas
201 = npm-4000-fddi-das
202 = npm-4000-1e
203 = npm-4000-1r
204 = npm-4000-2s
205 = npm--4000-2e1
206 = npm-4000-2e
207 = npm--4000-2r1
208 = npm-4000-2r
209 = npm-4000-4t
210 = npm-4000-4b
211 = npm-4000-8b
212 = npm-4000-ct1
213 = npm-4000-ce1
214 = npm-4000-1a
215 = npm-4000-6e
230 = pa-1fe
231 = pa-8e
232 = pa-4e
233 = pa-5e
234 = pa-4t
235 = pa-4r
236 = pa-fddi
237 = epa
242 = pa-1fe-tx-isl
243 = pa-1fe-fx-isl
244 = pa-1fe-tx-nisl

Syntax: Integer

Access: Read-only

cardDescr

Provides a description of the card used by the router. Examples of the descriptions are *MEC Ethernet* for an MEC board, *25MHz 68040* for the CSC/4, and *CTR Token Ring* for a CTR board.

Syntax: Display string

Access: Read-only

cardSerial

Provides the serial number of this card, or zero if unavailable.

Syntax: Integer

Access: Read-only

cardHwVersion

Provides the hardware revision level of this card, or an empty string if unavailable.

Syntax: Display string

Access: Read-only

cardSwVersion

Provides the version of the firmware or microcode installed on this card, or an empty string if unavailable. For example, *1.8* indicates MCI microcode 1.8, and *3.0 MADGE 1.01/4.02, TI 000000* indicates CSC-R16M.

Syntax: Display string

Access: Read-only

Chassis Group

cardSlotNumber

Provides the chassis slot number. A value of -1 is provided if it is not applicable or cannot be determined.

Syntax: Integer

Access: Read-only

cardContainedByIndex

The cardIndex value of the parent card that directly contains this card, or 0 if cardIndex is contained by the chassis, or -1 if not applicable or cannot be determined.

Syntax: Integer

Access: Read-only

cardOperStatus

The operational status of the card. cardOperStatus is *up* when a card is recognized by the device and is enabled for operations. cardOperStatus is *down* if the card is not recognized by the device, or if it is not enabled for operation.

Syntax: Integer (1 = not specified, 2 = up, 3 = down)

Access: Read-only

cardSlots

The number of slots on this card, or 0 if the card contains no slots or slots are not applicable, or -1 if the number cannot be determined.

Syntax: Integer

Access: Read-only

End of Table

chassisSlots

Provides the number of slots in this chassis. A value of -1 is provided if the number is not applicable or cannot be determined.

Syntax: Integer

Access: Read-only

Chassis Card Interface Index Table**cardIfIndex**

Matches RFC 1213 ifTable IfIndex.

Syntax: Integer

Access: Read-only

cardIfSlotNumber

Specifies the chassis slot number, or -1 if neither is applicable nor determinable.

Syntax: Integer

Access: Read-only

cardIfPortNumber

Specifies the chassis port number, unique per port on a given card if available.

Syntax: Integer

Access: Read-only

Chassis Group

cardIfCardIndex

The cardIndex value of the card in the *cardTable* that contains this interface.

Syntax: Integer

Access: Read-only

cardIfConnectorTypeEnabled

The interface connector type currently enabled. The value is 1 if the type is not known or not used, or 2 if none of the port's interface connectors on this port are enabled.

Syntax: Integer (1 = not specified, 2 = none, 3 = rj-45, 4 = db-40, 5 = db-15)

Access: Read-only

End of Table

chassisPartner

Specifies whether this chassis is a variant partner of a product.

Syntax: Integer (1 = Cisco, 2 = Synoptics, 3 = Chipcom, 4 = Cabletron, 5 = DEC, 6 = NCR, 7 = USRobotics, 8 = Alcatel, 9 = NEC, 10 = DSC, 11 = Microcom, 14 = HP)

Access: Read-only

sysUpTimeAtLastChassisChange

The time in hundredths of a second from the last cold start to the last change in the chassis configuration. This value is updated whenever the chassis experiences a change in the count, type, or slot position of a card in cardTable.

Syntax: TimeTicks

Access: Read-only

DECnet Group

This section describes the Cisco MIB variables pertaining to monitoring and managing a device running the DECnet protocol. These variables gather information, such as hop count, host name, total packets received and sent, and number of packets with header errors.

Note The terms *Level 1* and *Level 2* are used often with these variables. Level 1 routers can communicate with end nodes and with other Level 1 routers in an area. Level 2 routers can communicate with Level 1 routers in the same area and with Level 2 routers in different areas. The term *hellos* is also used. Hosts acknowledge the addresses of other hosts by listening to host hello messages. Hosts learn about nearby routers by listening to router hello messages.

dnBadhello

Provides the total number of received bad hello messages.

Syntax: Integer

Access: Read-only

dnBadlevel1

Provides the total number of bad Level 1 routing packets that have been received.

Syntax: Integer

Access: Read-only

dnBigaddr

Provides the total number of addresses that are too large.

Syntax: Integer

Access: Read-only

dnDdatas

Provides the total number of received data packets.

Syntax: Integer

Access: Read-only

dnFormaterr

Provides the total number of DECnet packets received with header errors.

Syntax: Integer

Access: Read-only

dnForward

Provides the total number of DECnet packets forwarded.

Syntax: Integer

Access: Read-only

dnHellos

Provides the total number of hello messages received.

Syntax: Integer

Access: Read-only

dnHelloSent

Provides the total number of output hello messages.

Syntax: Integer

Access: Read-only

dnLevel1s

Provides the total number of Level 1 routing packets received.

Syntax: Integer

Access: Read-only

dnLevel1sent

Provides the total number of Level 1 routing packets sent.

Syntax: Integer

Access: Read-only

dnLevel2s

Provides the total number of Level 2 routing packets received.

Syntax: Integer

Access: Read-only

dnLevel2sent

Provides the total number of Level 2 routing packets sent.

Syntax: Integer

Access: Read-only

dnNoaccess

Provides the total number of packets dropped due to access control failure.

Syntax: Integer

Access: Read-only

dnNoencap

Provides the total number of packets that were dropped because they could not be encapsulated.

Syntax: Integer

Access: Read-only

dnNomemory

Provides the total number of transactions denied due to lack of memory.

Syntax: Integer

Access: Read-only

dnNoroute

Provides the total number of packets that were dropped because the router did not know where to forward them.

Syntax: Integer

Access: Read-only

dnNotgateway

Provides the total number of packets that were received while not routing DECnet.

Syntax: Integer

Access: Read-only

dnNotimp

Provides the total number of unknown control packets received.

Syntax: Integer

Access: Read-only

dnNotlong

Provides the total number of received packets not in the long DECnet format. This number should always be zero.

Syntax: Integer

Access: Read-only

dnNovector

Provides the total number of missing routing vectors. Occurs when a packet is received for which there is no entry in the Routing table.

Syntax: Integer

Access: Read-only

dnOtherhello

Provides the total number of hello messages received from another area by a Level 1 router.

Syntax: Integer

Access: Read-only

dnOtherlevel1

Provides the total number of Level 1 routing packets received from another area.

Syntax: Integer

Access: Read-only

dnOtherlevel2

Provides the total number of Level 2 routing packets received from another area.

Syntax: Integer

Access: Read-only

dnReceived

Provides the number of total DECnet packets received.

Syntax: Integer

Access: Read-only

dnToomanyhops

Provides the total number of packets received that exceeded the maximum hop count set for this device and have been discarded.

Syntax: Integer

Access: Read-only

DECnet Area Routing Table

The DECnet Area Routing table, *dnAreaTable*, includes seven variables: *dnAAge*, *dnACost*, *dnAHop*, *dnAlfIndex*, *dnANextHop*, *dnAPrio*, and *dnArea*. The index for this table is the DECnet area, or *dnArea*. If there are *n* number of areas for the device, there will be *n* rows in the table.

For example, in Table 4-1, the DECnet area is 44, the cost is 3, and the maximum number of hops allowed is 2. The interface used to get to area 44 is number 1, the address for the next hop is 46.5, the Routing table was updated 30 seconds ago, and the next hop area is prioritized as 1.

Table 4-1 **DECnet Area Routing**

dnArea	dnACost	dnAHop	dnAlfIndex	dnANextHop	dnAAge	dnAPrio
44	3	2	1	46.5	30	1
24	60	4	2	24.7	12	2
6	17	2	3	6.4	60	3

dnAAge

Provides the age (in seconds) of an area route. When a route is used or has been verified as functional, its age is reset to 0. If a route is not used, its age will gradually grow. Eventually, routes with large ages are cleared out.

Syntax: Integer

Access: Read-only

dnACost

Provides the cost of the router area. The cost value can be an integer from 1 through 63. The cost signifies routing preference. The lower the cost, the better the path.

Syntax: Integer

Access: Read-only

dnAHop

Provides the maximum number of hops for a route to a distant area that the router will accept.

Syntax: Integer

Access: Read-only

dnAIfIndex

Provides the instance ID of the interface providing the next hop address to the area. A zero denotes self. The DECnet table is indexed by *dnArea*. For example, *dnAIfIndex.5* is the *ifIndex* for the next hop to *DECnet area 5*; *dnAIfIndex 7* is the *ifIndex* for the next hop to DECnet area 7; and so on.

If *dnAIfIndex.5* is set to the value of 4, to get to the next hop for DECnet area 5, the router sends the packet via the interface that has an *ifIndex* of 4.

Syntax: Integer

Access: Read-only

dnANextHop

Provides the DECnet address for the next hop.

Syntax: Octet string

Access: Read-only

dnAPrio

Provides the priority of the next hop router for an area route.

Syntax: Integer

Access: Read-only

dnArea

Indicates the DECnet area for the device.

Syntax: Integer

Access: Read-only

End of Table

DECnet Host Table

The DECnet host table, *dnHostTable*, contains seven variables: *dnHAge*, *dnHCost*, *dnHHop*, *dnHIIndex*, *dnHNextHop*, *dnHost*, and *dnHPrio*.

In Table 4-2, the first DECnet host address in the table is 44.5. Its cost is 3, the number of hops to the host is 4, and the interface number 1 provides the next hop to address 55.6. The route was updated 30 seconds ago, and the priority for the next hop is set to 4.

Table 4-2 DECnet Host

dnHost	dnHCost	dnHHop	dnHIfIndex	dnHNextHop	dnHAge	dnHPrio
44.5	3	4	1	55.6	30	4
54.6	1	3	2	33.2	20	3
23.2	2	1	3	25.1	60	2

dnHAge

Provides the age (in seconds) of the route to the host. When a route is used or has been verified as functional, its age is reset to 0. If a route is not used, the age of the route will gradually grow. Eventually, routes with large ages are cleared out.

Syntax: Integer

Access: Read-only

dnHCost

Provides the cost of the path to this device.

Syntax: Integer

Access: Read-only

dnHHop

Provides the number of hops to this device.

Syntax: Integer

Access: Read-only

dnHIIndex

Provides the index of the interface to the next hop address to the node. 0 denotes self.

Syntax: Integer

Access: Read-only

dnHNextHop

Provides the DECnet address of the next hop destination.

Syntax: Octet string

Access: Read-only

dnHost

Provides the DECnet node address.

Syntax: Integer

Access: Read-only

dnHPrio

Provides the priority of the next hop router for the node.

Syntax: Integer

Access: Read-only

End of Table

DECnet Interface Table

The DECnet Interface table, *dnIfTable*, contains the *dnIfCost* variable. The index to this table is *ifIndex*, or the interface number. If there are *n* number of interfaces associated with the device, there will be *n* rows in the table.

For example, in Table 4-3, interface 1 has a cost of 20, interface 2 has a cost of 31, and so on.

Table 4-3 DECnet Interface

Interface Number	dnIfCost
1	20
2	31
3	12

dnIfCost

Indicates the cost of this interface.

Syntax: Integer

Access: Read-only

End of Table

Novell Group

The variables in this group can be used with all Cisco products running the Novell protocol. These variables provide such information as total number of input and output packets, number of packets with errors, and number of packets with service access point (SAP) requests and replies.

novellBcastin

Indicates the total number of Novell input broadcast packets.

Syntax: Integer

Access: Read-only

novellBcastout

Indicates the total number of Novell output broadcast packets.

Syntax: Integer

Access: Read-only

novellChksum

Indicates the total number of Novell input packets with checksum errors.

Syntax: Integer

Access: Read-only

novellFormerr

Indicates the total number of Novell input packets with header errors.

Syntax: Integer

Access: Read-only

novellForward

Indicates the total number of Novell packets forwarded.

Syntax: Integer

Access: Read-only

novellHopcnt

Indicates the total number of Novell input packets that exceeded the maximum hop count.

Syntax: Integer

Access: Read-only

novellInmult

Indicates the total number of Novell input multicast packets.

Syntax: Integer

Access: Read-only

novellInput

Indicates the total number of Novell input packets.

Syntax: Integer

Access: Read-only

novellLocal

Indicates the total number of Novell input packets for this host.

Syntax: Integer

Access: Read-only

novellNoencap

Indicates the total number of Novell packets dropped due to output encapsulation failure.

Syntax: Integer

Access: Read-only

novellNoroute

Indicates the total number of Novell packets dropped because the router did not know where to forward them.

Syntax: Integer

Access: Read-only

novellOutput

Indicates the total number of Novell output packets.

Syntax: Integer

Access: Read-only

novellSapout

Indicates the total number of Novell service access point (SAP) request packets sent.

Syntax: Integer

Access: Read-only

novellSapreply

Indicates the total number of Novell SAP reply packets sent.

Syntax: Integer

Access: Read-only

novellSapreqin

Indicates the total number of Novell SAP request packets received.

Syntax: Integer

Access: Read-only

novellSapresin

Indicates the total number of Novell SAP response packets received.

Syntax: Integer

Access: Read-only

novellUnknown

Indicates the total number of unknown Novell input packets.

Syntax: Integer

Access: Read-only

ipxActLostByts

Provides the total bytes of lost IPX packets.

Syntax: Counter

Access: Read-only

ipxActLostPkts

Provides the number of lost IPX packets due to memory limitations.

Syntax: Counter

Access: Read-only

ipxActThresh

Provides the threshold of IPX accounting records in use before IPX traffic will be unaccounted.

Syntax: Integer

Access: Read-only

Local IPX Accounting Table

The Local IPX Accounting table (see Table 4-4), *lipxAccountingTable*, provides access to the Cisco IPX accounting support. This table includes the following variables: *ipxActByts*, *ipxActDst*, *ipxActPkts*, and *ipxActSrc*.

Table 4-4 Local IPX Accounting Table

ipxActByts	ipxActDst	ipxActPkts	ipxActSrc
10,000	1.000.0230.0110	40	BADDAD.0110.0220.0333

ipxActByts

Provides the total number of bytes in IPX packets from source to destination.

Syntax: Counter

Access: Read-only

ipxActDst

Provides the IPX destination address for the host traffic matrix.

Syntax: Octet String

Access: Read-only

ipxActPkts

Provides the number of IPX packets sent from source to destination.

Syntax: Counter

Access: Read-only

ipxActSrc

Provides the IPX source address for the host traffic matrix.

Syntax: Octet String

Access: Read-only

End of Table

ipxActAge

Provides the age of the data in the current IPX data matrix.

Syntax: TimeTicks

Access: Read-only

Local IPX Checkpoint Accounting Table

The Local IPX Checkpoint Accounting table *ipxCkAccountingTable* (see Table 4-5), includes four related variables: *ipxCkactByts*, *ipxCkactDst*, *ipxCkactPkts*, and *ipxCkactSrc*. The indexes for this table are *ckActSrc* and *ckActDst*.

Table 4-5 IPX Checkpoint Accounting

ipxCkactByts	ipxCkactDst	ipxCkactPkts	ipxCkactSrc
10,000	1.000.0230.0110	40	BADDAD.0110.0220.0333

ipxCkactByts

Provides the number of bytes in IPX packets from source to destination in the checkpoint matrix.

Syntax: Counter

Access: Read-only

ipxCkactDst

Provides the IPX destination address for host in the checkpoint traffic matrix.

Syntax: IPXaddress

Access: Read-only

Virtual Integrated Network Service (VINES) Group

ipxCkactPkts

Provides the number of IPX packets sent from source to destination in checkpoint matrix.

Syntax: Integer

Access: Read-only

ipxCkactSrc

Provides the IPX source address for host in the checkpoint traffic matrix.

Syntax: IPXaddress

Access: Read-only

End of Table

ipxCkactAge

Provides the age of data in the IPX the checkpoint matrix.

Syntax: TimeTicks

Access: Read-only

ipxActCheckPoint

Provides a checkpoint to the IPX accounting database. This MIB variable must be read and then set with the same value for the checkpoint to succeed. The value that is read and then set is incremented after a successful set request.

Syntax: Integer

Access: Read-write

Virtual Integrated Network Service (VINES) Group

The variables in this section, from the VINES group in Cisco IOS Release 10.2, have been deprecated and replaced with the Cisco VINES (cv) group, found in the ciscoMgmt tree.

The variables in this group can be used with all Cisco products running the Banyan Virtual Integrated Network Service (VINES) protocol. This protocol is derived from the Xerox Network Systems (XNS) protocol. These variables provide information such as total number of input and output packets, number of packets with errors, and number of packets with Internet Control Protocol(ICP) requests and replies.

vinesBcastfwd

Indicates the total number of VINES broadcast packets forwarded.

Syntax: Integer

Access: Read-only

vinesBcastin

Indicates the total number of VINES input broadcast packets.

Syntax: Integer

Access: Read-only

vinesBcastout

Indicates the total number of VINES output broadcast packets.

Syntax: Integer

Access: Read-only

vinesCksumerr

Indicates the total number of VINES input packets with checksum errors.

Syntax: Integer

Access: Read-only

vinesClient

Indicates the next VINES subnetwork number that this router will assign to a client.

Syntax: Integer

Access: Read-only

vinesEchoIn

Indicates the total number of VINES echo packets received.

Syntax: Integer

Access: Read-only

vinesEchoOut

Indicates the total number of VINES echo packets generated.

Syntax: Integer

Access: Read-only

vinesEncapsfailed

Indicates the total number of VINES packets dropped because they could not be encapsulated.

Syntax: Integer

Access: Read-only

vinesFormaterror

Indicates the total number of VINES input packets with header errors.

Syntax: Integer

Access: Read-only

vinesForwarded

Indicates the total number of VINES packets forwarded.

Syntax: Integer

Access: Read-only

vinesHopcount

Indicates the total number of VINES input packets that have exceeded the maximum hop count.

Syntax: Integer

Access: Read-only

vinesIcpIn

Indicates the total number of VINES Internet Control Protocol (ICP) packets received.

Syntax: Integer

Access: Read-only

vinesIcpOut

Indicates the total number of VINES ICP packets generated.

Syntax: Integer

Access: Read-only

vinesInput

Indicates the total number of VINES input packets.

Syntax: Integer

Access: Read-only

Virtual Integrated Network Service (VINES) Group

vinesLocalDest

Indicates the total number of VINES input packets for this host.

Syntax: Integer

Access: Read-only

vinesMacEchoIn

Indicates the total number of VINES MAC level echo packets received.

Syntax: Integer

Access: Read-only

vinesMacEchoOut

Indicates the total number of VINES Media Access Control (MAC) level echo packets generated.

Syntax: Integer

Access: Read-only

vinesMetricOut

Indicates the total number of VINES ICP metric notification packets generated.

Syntax: Integer

Access: Read-only

vinesNet

Indicates the VINES network number of this router.

Syntax: Integer

Access: Read-only

vinesNoCharges

Indicates the total number of VINES broadcast packets not forwarded to all interfaces because the *no charges only* bit in the packet was set to *on*.

Syntax: Integer

Access: Read-only

vinesNoRoute

Indicates the total number of VINES packets dropped because the router did not know where to forward them.

Syntax: Integer

Access: Read-only

vinesNotgt4800

Indicates the total number of VINES broadcast packets not forwarded to all interfaces because the *over 4800 bps* bit in the packet was set to *on*.

Syntax: Integer

Access: Read-only

vinesNotlan

Indicates the total number of VINES broadcast packets not forwarded to all interfaces because the *lan only* bit in the packet was set to *on*.

Syntax: Integer

Access: Read-only

vinesOutput

Indicates the total number of VINES output packets.

Syntax: Integer

Access: Read-only

vinesProxyCnt

Indicates the total number of VINES packets that were sent to an actual Banyan server as a proxy for a client.

Syntax: Counter

Access: Read-only

vinesProxyReplyCnt

Indicates the total number of received VINES packets that were responses to proxy packets sent by the router.

Syntax: Counter

Access: Read-only

vinesSubnet

Indicates the VINES subnet number of this router.

Syntax: Integer

Access: Read-only

vinesUnknown

Indicates the total number of unknown VINES input packets.

Syntax: Integer

Access: Read-only

Banyan VINES Interface Table

The Banyan VINES Interface table, *vinesIfTableEntry*, contains all the variables described in the Banyan VINES group. The index to the table is *ifIndex*. *ifIndex* indicates the number of the interface. If the device has *n* number of interfaces, the VINES Interface table will contain *n* rows.

In Table 4-6, the first column indicates the number of interfaces on the devices. Each of the variables in the VINES Interface table occupies one column; for example, *vinesIfMetric* is shown in a column, followed by *vinesIfEncType* in the next column, and so on.

Table 4-6 Banyan VINES Interface Table

Interface Number	vinesIfMetric	vinesIfEncType	and so on
1	3	1	
2	5	3	
and so on			

vinesIfAccesslist

Provides the outgoing access list number for the VINES protocol.

Syntax: Integer

Access: Read-only

vinesIfArpEnabled

Indicates how the router responds to the VINES protocol ARP.

Syntax: Integer (0 = never respond to ARP packets, 1 = always respond to ARP packets, 2 = respond to ARP packets only if servers are not present on the network)

Access: Read-only

vinesIfEncType

Indicates the type of data link encapsulation that will be used on broadcasts sent by the router.

Syntax: Integer (1 = ARPA, 3 = SNAP, 5 = HDLC, 12 = X.25, 13 = X.25, 25 = VINES TR, 27 = Frame Relay, 28 = SMDS, 30 = PPP)

Access: Read-only

vinesIfFastOkay

Indicates whether fast switching is supported for the VINES protocol.

Syntax: Integer (0 = fast switching not requested or not supported, 1 = fast switching requested and supported)

Access: Read-only

vinesIfInputNetworkFilter

Provides the access list number for filtering the content of received VINES routing information.

Syntax: Integer

Access: Read-only

vinesIfInputRouterFilter

Provides the access list number for filtering on the source of received VINES routing information.

Syntax: Integer

Access: Read-only

vinesIfLineup

Indicates whether the VINES protocol line is up or down.

Syntax: Integer (0 = down, 1 = up)

Access: Read-only

vinesIfMetric

Provides the metric value for the VINES protocol. Banyan servers use delay metrics to compute timeouts when communicating with other hosts. The metric value is either manually assigned to the interface by using the **vines metric** command or is automatically assigned by the system. This number is returned in the format defined in the VINES Protocol Definition.

Syntax: Integer

Access: Read-only

vinesIfOutputNetworkFilter

Provides the access list number for filtering the content of transmitted VINES routing information.

Syntax: Integer

Access: Read-only

vinesIfPropagate

Indicates whether the VINES protocol “propagate” is enabled.

Syntax: Integer (0 = never enabled, 1 = always enabled, 2 = enabled only if there are no local servers on any interface)

Access: Read-only

vinesIfRedirectInterval

Provides the redirect interval for the VINES protocol.

Syntax: Integer

Access: Read-only

vinesIfRouteCache

Indicates whether fast switching is supported for the VINES protocol.

Syntax: Integer

Access: Read-only

vinesIfRxArp0Cnt

Provides the number of input ARP query request messages for the VINES protocol. The four types of ARP messages following apply to *vinesIfRxArp0*–*vinesIfRxArp3*:

- Service request (type 0)—Query to find servers
- Service response (type 1)—Server indicating its presence
- Assignment request (type 2)—Client asking to be assigned a VINES IP address
- Assignment response (type 3)—Server assigning a VINES IP address to a client

Syntax: Counter

Access: Read-only

vinesIfRxArp1Cnt

Provides the number of input ARP query response messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxArp2Cnt

Provides the number of input ARP assignment request messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxArp3Cnt

Provides the number of input ARP assignment response messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxArpIllegalCnt

Provides the number of input illegal ARP messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxBcastDuplicateCnt

Provides the input duplicate broadcast count for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxBcastForwardedCnt

Provides the VINES protocol number of input packets forwarded to another interface.

Syntax: Counter

Access: Read-only

vinesIfRxBcastHelperedCnt

Provides the VINES protocol number of input packets helpered to another server. Helpered packets are broadcasts received from a serverless network that should be thrown away according to the fields in the VINES IP header. Instead of being thrown away, they are resent on another interface, so that they will be received by a VINES server.

Syntax: Counter

Access: Read-only

vinesIfRxBcastinCnt

Provides the input broadcast count for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxChecksumErrorCnt

Provides the number of input packets with checksum errors for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxEchoCnt

Provides the number of input IPC echo messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxFormatErrorCnt

Provides the number of input packets with format errors for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxForwardedCnt

Provides the VINES protocol number of input packets forwarded to another interface.

Syntax: Counter

Access: Read-only

vinesIfRxIcpErrorCnt

Provides the number of input interprocess communications (ICP) error messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxIcpIllegalCnt

Provides the number of input illegal ICP messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxIcpMetricCnt

Provides the number of input ICP metric messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxIpcCnt

Provides the number of input IPC messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxLocalDestCnt

Provides the VINES protocol number of input packets destined for this router.

Syntax: Counter

Access: Read-only

vinesIfRxMacEchoCnt

Provides the number of input MAC layer echo frames for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxNoRouteCnt

Provides the VINES protocol number of input packets that were dropped because there was no route to the destination.

Syntax: Counter

Access: Read-only

vinesIfRxNotEnabledCnt

Provides the VINES protocol number of input packets that were discarded because the interface was not configured.

Syntax: Counter

Access: Read-only

vinesIfRxProxyReplyCnt

Provides the VINES protocol number of responses to proxy packets.

Syntax: Counter

Access: Read-only

vinesIfRxRtp0Cnt

Provides the number of illegal input Routing Table Protocol (RTP) type 0 messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxRtp1Cnt

Provides the number of input RTP type 1 (request for information) messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxRtp2Cnt

Provides the number of illegal input RTP type 2 messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxRtp3Cnt

Provides the number of illegal input RTP type 3 messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxRtp4Cnt

Provides the number of input RTP type 4 update messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxRtp5Cnt

Provides the number of input RTP type 5 response messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxRtp6Cnt

Provides the number of input RTP type 6 redirect messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxRtpIllegalCnt

Provides the number of all other illegal input RTP messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxIpUnknownCnt

Provides the number of input messages from unknown VINES protocols.

Syntax: Counter

Access: Read-only

vinesIfRxIpcUnknownCnt

Provides the number of input messages from unknown VINES IPC ports.

Syntax: Counter

Access: Read-only

vinesIfRxSppCnt

Provides the number of input SPP messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfRxZeroHopCountCnt

Provides VINES protocol number of input packets dropped due to a zero hop count.

Syntax: Counter

Access: Read-only

vinesIfServerless

Indicates whether the VINES protocol serverless support is enabled.

Syntax: Integer (0 = never enabled, 1 = enabled only if servers are not present on the network, 2 = always enabled, 3 = always enabled to flood broadcasts)

Access: Read-only

vinesIfServerlessBcast

Indicates whether VINES protocol serverless broadcasting support is enabled.

Syntax: Counter (0 = not enabled, 1 = enabled)

Access: Read-only

vinesIfSplitDisabled

Indicates whether the VINES protocol split horizon is enabled.

Syntax: Integer (0 = enabled, 1 = disabled)

Access: Read-only

vinesIfTxArp0Cnt

Provides the number of output ARP query request messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxArp1Cnt

Provides the number of output ARP query response messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxArp2Cnt

Provides the number of output ARP assignment request messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxArp3Cnt

Provides the number of input ARP assignment response messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxBcastCnt

Provides broadcast packets that were generated by the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxBcastForwardedCnt

Provides the VINES protocol output broadcast forwarded from another interface.

Syntax: Counter

Access: Read-only

vinesIfTxBcastHelperedCnt

Provides the VINES protocol output broadcast helpered to a Banyan server.

Syntax: Counter

Access: Read-only

vinesIfTxEchoCnt

Provides the number of output IPC echo messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxFailedAccessCnt

Provides the number of packets to be output that failed on access list for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxFailedDownCnt

Provides the number of VINES packets that could not be output because the interface was down.

Syntax: Counter

Access: Read-only

vinesIfTxFailedEncapsCnt

Provides VINES packets to be output that could not be encapsulated.

Syntax: Counter

Access: Read-only

vinesIfTxForwardedCnt

Provides the number of forwarded packets for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxIcpErrorCnt

Provides the number of output IPC error messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxIcpMetricCnt

Provides the number of output IPC metric messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxIpcCnt

Provides the number of output ICP messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxMacEchoCnt

Provides the number of output IPC MAC-layer echo frames for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxNotBcastNotgt4800Cnt

Provides the VINES protocol output broadcast not sent due to high-speed class. This occurs if a received packet is marked to be sent only on network interfaces with a speed of 4800 bps or greater. The counter is incremented on interfaces with a speed of less than 4800 whenever this type of packet should have been transmitted.

Syntax: Counter

Access: Read-only

vinesIfTxNotBcastNotLanCnt

Provides the VINES protocol output broadcast not sent due to *LanOnly* class. This occurs if a received packet is marked to be sent only if the network interface type is *LanOnly*. The counter is incremented on interfaces other than type *LanOnly* whenever this type of packet should have been transmitted.

Syntax: Counter

Access: Read-only

vinesIfTxNotBcastPpchargeCnt

Provides VINES protocol output broadcast not sent due to *No Charges* class. This occurs if a received packet is marked to be sent only if the sender's transmission is free of charge. The counter is incremented on interfaces carrying per-packet charges whenever this type of packet should have been transmitted.

Syntax: Counter

Access: Read-only

vinesIfTxNotBcastToSourceCnt

Provides the VINES protocol output broadcast packets that were not sent due to the interface leading back to the source.

Syntax: Counter

Access: Read-only

vinesIfTxProxyCnt

Provides the number of proxy packets sent by the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxRtp0Cnt

Provides the number of illegal output RTP type 0 messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxRtp1Cnt

Provides the number of output RTP type 1 (request messages) for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxRtp2Cnt

Provides the number of illegal output RTP type 2 messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxRtp3Cnt

Provides the number of illegal output RTP type 3 messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxRtp4Cnt

Provides the number of output RTP type 4 (update messages) for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxRtp5Cnt

Provides the number of output RTP type 5 (response messages) for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxRtp6Cnt

Provides the number of output RTP type 6 (redirect messages) for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxSppCnt

Provides the number of output SPP messages for the VINES protocol.

Syntax: Counter

Access: Read-only

vinesIfTxUnicastCnt

Provides the unicast packets that were generated for the VINES protocol.

Syntax: Counter

Access: Read-only

Xerox Network Systems (XNS) Group

This group is present in all router-based products running the Xerox Network Systems (XNS) protocol. These variables provide such information as the number of packets forwarded, total number of input packets, and total number of packets transmitted with errors.

xnsBcastIn

Indicates the total number of XNS input broadcast packets.

Syntax: Integer

Access: Read-only

xnsBcastout

Indicates the total number of XNS output broadcast packets.

Syntax: Integer

Access: Read-only

xnsChksum

Indicates the total number of XNS input packets with checksum errors.

Syntax: Integer

Access: Read-only

xnsEchorepin

Indicates the total number of XNS echo reply packets received.

Syntax: Integer

Access: Read-only

xnsEchorepout

Indicates the total number of XNS echo reply packets sent.

Syntax: Integer

Access: Read-only

xnsEchoreqin

Indicates the total number of XNS echo request packets received.

Syntax: Integer

Access: Read-only

xnsEchoreqout

Indicates the total number of XNS echo request packets sent.

Syntax: Integer

Access: Read-only

xnsErrin

Indicates the total number of XNS error input packets.

Syntax: Integer

Access: Read-only

xnsErrout

Indicates the total number of XNS error output packets.

Syntax: Integer

Access: Read-only

xnsForward

Indicates the total number of XNS packets forwarded.

Syntax: Integer

Access: Read-only

xnsFormerr

Indicates the total number of XNS input packets with header errors.

Syntax: Integer

Access: Read-only

xnsFwdbrd

Indicates the total number of XNS broadcast packets forwarded.

Syntax: Integer

Access: Read-only

xnsHopcnt

Indicates the total number of XNS input packets that exceeded the maximum hop count.

Syntax: Integer

Access: Read-only

xnsInmult

Indicates the total number of XNS input packets received with multicast addresses.

Syntax: Integer

Access: Read-only

xnsInput

Indicates the total number of input XNS packets.

Syntax: Integer

Access: Read-only

xnsLocal

Indicates the total number of XNS input packets for this host.

Syntax: Integer

Access: Read-only

xnsNoencap

Provides the total number of XNS packets dropped because they could not be encapsulated.

Syntax: Integer

Access: Read-only

xnsNoroute

Indicates the total number of XNS packets that were discarded because the router did not know where to forward them.

Syntax: Integer

Access: Read-only

xnsNotgate

Indicates the total number of XNS input packets received while XNS routing was not enabled.

Syntax: Integer

Access: Read-only

xnsOutput

Indicates the total number of XNS output packets.

Syntax: Integer

Access: Read-only

xnsUnknown

Indicates the total number of unknown XNS input packets.

Syntax: Integer

Access: Read-only

Public SNMP Traps Supported by Cisco

SNMP traps are set up on specific devices to obtain useful information such as the change in a device configuration or the absence of proper user authentication with a request. When the SNMP agent on the device detects a change, it immediately sends an SNMP trap to the NMS system.

Network managers must be wary of depending on traps for vital information. Because traps are sent as datagrams with no acknowledgement they can be lost in the network due to, for example, congestion or errors.

Cisco products, including the routers, access servers and communication servers, and protocol translators, support the SNMP traps specified in RFC 1213, *Management Information Base for Network Management of TCP/IP-based Internets: MIB-II*. The *warmStart* trap in MIB II is not supported by Cisco.

Following are the standard SNMP traps supported by Cisco:

authenticationFailure

This trap is sent to the NMS system if the SNMP agent detects that proper user authentication was not provided with a request. User authentication enhances the security of the devices by ensuring that only privileged users with valid community strings are allowed to access the system.

coldStart

The SNMP agent sends a *coldStart* trap when its device has reinitialized itself.

egpNeighborloss

An *egpNeighborLoss* trap indicates that an EGP (Exterior Gateway Protocol) neighbor is down. Neighboring routers are two routers that have interfaces to a common network and exchange routing information. An exterior router uses EGP to advertise its knowledge of routes to networks within its autonomous system. It sends these advertisements to the core routers, which then readvertise their collected routing information to the exterior router. A neighbor or peer router is any router with which the router communicates using EGP.

linkDown

A *linkDown* trap is sent by the SNMP agent to the NMS system if a link in a configuration of a device has been shutdown. For example, the link could be a serial line connecting two devices or an Ethernet link between two networks.

linkUp

A *linkUp* trap indicates the recognition of an SNMP agent that a link in a configuration of a device has become active.

SNMP Traps Defined by Cisco

Following are the Cisco private SNMP traps that are implemented in Cisco products including the router, access server and communication server, and protocol translator.

ipxTrapCircuitUp

This trap signifies that the specified circuit has come up.

ipxTrapCircuitDown

This trap signifies that the specified circuit has gone down.

ciscoPingCompletionTrap

A *ciscoPingCompleted* trap is sent at the completion of a sequence of pings if such a trap was requested when the sequence was initiated.

reload

This trap is sent after a reload command is issued.

tcpConnectionClose

The *tty* trap indicates that a TCP connection, which existed previously for a tty session, has been terminated.

Variables Supported in RFC 1285

The following variables in RFC 1285 are supported in Software Release 9.0 and later:

snmpFddiSMTNumber

snmpFddiSMTIndex

snmpFddiSMTStationId

snmpFddiSMTOpVersionId

snmpFddiSMTHiVersionId

snmpFddiSMTLoVersionId

snmpFddiSMTCFState

snmpFddiMACNumber

snmpFddiMACSMTIndex

snmpFddiMACIndex

snmpFddiMACTReq

snmpFddiMACTNegj

snmpFddiMACTMax

snmpFddiMACTvxValue

snmpFddiMACMin

snmpFddiMACFrameCts

snmpFddiMACErrorCts

snmpFddiMACLostCts

snmpFddiMACChipSet

