Understanding Table Index Values in SNMP

Document ID: 40700

Contents

Introduction

When polling Simple Network Management Protocol (SNMP) objects, you must at times know exactly what is being polled. In order to fully understand this, you need to know how to correlate the object that is being polled with what you want to poll. This document covers the basics of how to use indexes in SNMP to group objects into tables.

Prerequisites

Requirements

Readers of this document should have knowledge of these topics:

- General knowledge of SNMP
- Software used to query Cisco devices via SNMP

Components Used

The information in this document is based on these software and hardware versions:

- UCD SNMP Version 4.2
- Cisco Catalyst 5509 with Cisco IOS® Software Release 5.5(7)

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

Getting Started With ifIndex

One of the first things to learn, when you are dealing with SNMP, is ifIndex. This is a primary key of all objects. Consider it a way that all of the interfaces (physical and logical) are broken down and assigned a value. This value is assigned during boot up of a device, and it may not be changed. If any information needs to be polled for that particular interface, it must use that assigned value.

IfIndex is defined in the IF–MIB (RFC 1213) in this manner:

```
InterfaceIndex ::= TEXTUAL-CONVENTION
       DISPLAY-HINT "d"
       STATUS       current
       DESCRIPTION
              "A unique value, greater than zero, for each interface
              or interface sub-layer in the managed system.  It is
              recommended that values are assigned contiguously
              starting from 1.  The value for each interface sub-
              layer must remain constant at least from one re-
              initialization of the entity's network management
              system to the next re-initialization."
       SYNTAX       Integer32 (1..2147483647)
```

For any MIB, a quick way to tell what index organizes a table is to look at the table entry:

```
ifEntry OBJECT-TYPE
       SYNTAX       IfEntry
       MAX-ACCESS   not-accessible
       STATUS       current
       DESCRIPTION
           "An entry containing management information applicable
           to a particular interface."
       INDEX    { ifIndex }
     ::= { ifTable 1 }
```

Given a MIB and a table entry, you can determine how the table is indexed. The next section provides examples of ifIndex.

Polling Objects

Polling Objects Based on ifIndex

When you issue the **snmpwalk** command to poll an ifIndex–based object (ifName) for port 7/4 on the switch, you get this output:

```
sj-cse-568: snmpwalk 172.16.99.60 public ifname

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.1 = sc0
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.2 = sl0
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.3 = VLAN-1
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.4 = VLAN-1002
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.5 = VLAN-1004
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.6 = VLAN-1005
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.7 = VLAN-1003
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.8 = 7/1
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.9 = 7/2
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.10 = 7/3

!--- This is the relevant line:

ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.11 = 7/4
```

```
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.12 = 7/5
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.13 = 7/6
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.14 = 7/7
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.15 = 7/8
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.16 = 7/9
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.17 = 7/10
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.18 = 7/11
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.19 = 7/12
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.20 = ATM8/0
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.22 = /A
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.23 = /B
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.24 = Nu0
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.25 = LEC/ATM8/0.10
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.532 = 3/1
ifMIB.ifMIBObjects.ifXTable.ifXEntry.ifName.533 = 3/2

!--- Output suppressed.
```

In that output from a poll of ifName (ifDescr on routers), notice that there is a number attached to each row, after `ifName`. This is the ifIndex that is assigned to the actual interface in the same row. This means that the second row of the poll, port 7/4, is assigned an ifIndex of 11. If you want information on port 7/4 from an ifIndexed object, use an index of 11. This means adding a `.11` to the end of an MIB object identifier (OID), to retrieve the instance of that object which corresponds to the same ifIndex values.

Polling Objects If the Table Is Not Indexed by ifIndex or is Cross Indexed

Sometimes, tables are not indexed by ifIndex, such as with BRIDGE–MIB. This output examines how it is indexed:

```
dot1dBasePortEntry OBJECT-TYPE
        SYNTAX  Dot1dBasePortEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "A list of information for each port of the
                bridge."
        REFERENCE
                "IEEE 802.1D-1990: Section 6.4.2, 6.6.1"
        INDEX  { dot1dBasePort }
        ::= { dot1dBasePortTable 1 }
```

That output shows that dot1dBasePortEntry is indexed by dot1dBasePort. How does that translate back to ifIndex? BRIDGE–MIB accesses an object called dot1dBasePortIfIndex. The object is defined in this manner:

```
dot1dBasePortIfIndex OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The value of the instance of the ifIndex object,
                defined in MIB-II, for the interface corresponding
                to this port."
        ::= { dot1dBasePortEntry 2 }
```

That output shows how to correlate from BRIDGE–MIB to IF–MIB. The next example shows how it all fits together.

Note: BRIDGE–MIB is built per vlan, hence community public@*vlan−id* must be used for non−vlan1 environments.

Correlating BRIDGE−MIB to IF−MIB

When you issue an **snmpwalk** on the BRIDGE−MIB, you get the next sample output for an Index. Use dot1dBasePortIfIndex (.1.3.6.1.2.1.17.1.4.1.2) to map it back to ifIndex. Once you have the ifIndex, use it to poll other objects based on ifIndex.

```
sj-cse-568: snmpwalk 172.16.99.60 public .1.3.6.1.2.1.17.1.4.1.2

17.1.4.1.2.203 = 671
17.1.4.1.2.204 = 672
17.1.4.1.2.205 = 673
17.1.4.1.2.206 = 674
17.1.4.1.2.207 = 675
17.1.4.1.2.208 = 676
17.1.4.1.2.209 = 677
17.1.4.1.2.210 = 678
17.1.4.1.2.211 = 679
17.1.4.1.2.212 = 680
17.1.4.1.2.213 = 681
17.1.4.1.2.214 = 682
17.1.4.1.2.215 = 683
17.1.4.1.2.216 = 684
17.1.4.1.2.257 = 581
17.1.4.1.2.385 = 8
17.1.4.1.2.386 = 9
17.1.4.1.2.387 = 10
17.1.4.1.2.388 = 11
17.1.4.1.2.389 = 12
17.1.4.1.2.390 = 13
17.1.4.1.2.391 = 14
17.1.4.1.2.392 = 15
17.1.4.1.2.393 = 16
17.1.4.1.2.394 = 17
17.1.4.1.2.395 = 18
17.1.4.1.2.396 = 19
17.1.4.1.2.449 = 22
```

The bold text line (`17.1.4.1.2.388 = 11`) shows that `.388` is an index. Because you polled dot1dBasePortIfIndex object from BRIDGE−MIB, .388 is the dot1dBasePortIfIndex. The `11` in the output line is actually the ifIndex. If you gather the information from this poll and from the previous poll, you can determine that port 7/4 has an ifIndex of 11 and a dot1dBasePortIfIndex (Index for BRIDGE−MIB) of .388.

Related Information

- **Technical Support − Cisco Systems**

Updated: Nov 01, 2005   Document ID: 40700