IBM Systems and Technology Group
January 2010

Obtaining SNMP data from z/VM systems

Jay Brenneman – rjbrenn@us.ibm.com

Table of Contents

Why SNMP?

Let us explore a common scenario:

When the z/VM® Systems Programmer requests more than one hundred IP addresses to be used by guests inside z/VM, the network team will usually want to know how those hosts will connect to the one or two fibers that the network team provides for z/VM. At this point, the z/VM Systems Programmer starts talking about VSWITCHes and the network team wants to know how they will manage this new switch that the Systems Programmer has bought and installed. Hilarity ensues, as the mis-communication is eventually resolved, but the network team keeps coming back to the management issue: How to monitor the status of what is going on inside a VSWITCH?

Simple Network Management Protocol (SNMP) is an industry standard method of managing network devices. It was originally codified in RFC 1067 dated August 1988. The basic level of functionality an SNMP client offers includes:

- Identifying the host itself
- Providing the administrators' contact information
- Providing the port numbers and addresses of the network interfaces
- Sending notification of link state changes

Because SNMP is the foundation for many network infrastructure management tools, it makes sense to be able to include z/VM, and especially the VSWITCHes inside z/VM, in an SNMP management infrastructure.

While researching SNMP, the following tutorial might be helpful for establishing basic competency and learning fundamental terminology:
http://oreilly.com/perl/excerpts/system-admin-with-perl/twenty-minute-snmp-tutorial.html

SNMP has grown significantly since 1988. It is now common to hear the remark that the simplicity has been lost. We will see examples of SNMP's complexity later in this document.

SNMP Data

The following example is the output of a snmpwalk command as run from a Linux® system against a z/VM system. Snmpwalk is a Linux tool that iteratively traverses the entire SNMP data tree and prints the tree contents to standard output. For more information on snmpwalk refer to http://net-snmp.sourceforge.net/docs/man/snmpwalk.html

```
[root@litnetm1 ~]# snmpwalk -v 1 -c TICLNET 192.168.70.26
SNMPv2-MIB::sysDescr.0 = STRING: IBM 2097; z/VM Version 5 Release 4.0, service level 0901
(64-bit), VM TCP/IP Level 540; RSU 0901 running TCPIP MODULE M2 dated 05/28/09 at 11:41
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.2.2.1.2.3
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (180428500) 20 days, 21:11:25.00
SNMPv2-MIB::sysContact.0 = STRING:  BOB ADMIN (TL 555-1122) GARY SYSPROG (TL 555-
1133) DEPARTMENT OF REDUNDANCY DEPARTMENT
SNMPv2-MIB::sysName.0 = STRING: LTICVM9.PDL.POK.IBM.COM
SNMPv2-MIB::sysLocation.0 = STRING:  BUILDINGA 123 FAKE STREET POUGHKEEPSIE,
NY 12601
SNMPv2-MIB::sysServices.0 = INTEGER: 76
IF-MIB::ifNumber.0 = INTEGER: 1
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifDescr.1 = STRING: ETHERNET via QDIO
IF-MIB::ifType.1 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifMtu.1 = INTEGER: 8992
truncating unsigned value to 32 bits (2)
IF-MIB::ifSpeed.1 = Gauge32: 4294967295
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifAdminStatus.1 = INTEGER: up(1)
IF-MIB::ifOperStatus.1 = INTEGER: up(1)
IF-MIB::ifLastChange.1 = Timeticks: (455) 0:00:04.55
IF-MIB::ifInOctets.1 = Counter32: 94508498
IF-MIB::ifInUcastPkts.1 = Counter32: 731997
IF-MIB::ifInNUcastPkts.1 = Counter32: 0
IF-MIB::ifInDiscards.1 = Counter32: 0
IF-MIB::ifInErrors.1 = Counter32: 0
IF-MIB::ifInUnknownProtos.1 = Counter32: 0
IF-MIB::ifOutOctets.1 = Counter32: 167006388
truncating unsigned value to 32 bits (2)
IF-MIB::ifOutUcastPkts.1 = Counter32: 724173
IF-MIB::ifOutNUcastPkts.1 = Counter32: 0
IF-MIB::ifOutDiscards.1 = Counter32: 0
IF-MIB::ifOutErrors.1 = Counter32: 0
IF-MIB::ifOutQLen.1 = Gauge32: 0
IF-MIB::ifSpecific.1 = OID: SNMPv2-SMI::zeroDotZero
IP-MIB::ipForwarding.0 = INTEGER: forwarding(1)
IP-MIB::ipDefaultTTL.0 = INTEGER: 60
```

```
IP-MIB::ipInReceives.0 = Counter32: 451887592
IP-MIB::ipInHdrErrors.0 = Counter32: 25510
IP-MIB::ipInAddrErrors.0 = Counter32: 1975489
IP-MIB::ipForwDatagrams.0 = Counter32: 30172026
IP-MIB::ipInUnknownProtos.0 = Counter32: 0
IP-MIB::ipInDiscards.0 = Counter32: 0
IP-MIB::ipInDelivers.0 = Counter32: 5930
IP-MIB::ipOutRequests.0 = Counter32: 397795068
IP-MIB::ipOutDiscards.0 = Counter32: 0
IP-MIB::ipOutNoRoutes.0 = Counter32: 0
IP-MIB::ipReasmTimeout.0 = INTEGER: 255 seconds
IP-MIB::ipReasmReqds.0 = Counter32: 22
IP-MIB::ipReasmOKs.0 = Counter32: 11
IP-MIB::ipReasmFails.0 = Counter32: 0
IP-MIB::ipFragOKs.0 = Counter32: 11
IP-MIB::ipFragFails.0 = Counter32: 0
IP-MIB::ipFragCreates.0 = Counter32: 22
IP-MIB::ipAdEntAddr.192.168.70.26 = IpAddress: 192.168.70.26
IP-MIB::ipAdEntNetMask.192.168.70.26 = IpAddress: 255.255.255.0
IP-MIB::ipAdEntBcastAddr.192.168.70.26 = INTEGER: 1
IP-MIB::ipAdEntReasmMaxSize.192.168.70.26 = INTEGER: 9216
RFC1213-MIB::ipRouteDest.0.0.0.0 = IpAddress: 0.0.0.0
RFC1213-MIB::ipRouteDest.192.168.70.0 = IpAddress: 192.168.70.0
Error in packet.
Reason: (genError) A general failure occured
Failed object:
```

The snmpwalk output illustrates a couple of interesting points about SNMP. SNMP data is organized in a tree structure. The various subtrees of this tree that pertain to a single device or function are grouped together in a Management Information Block (MIB).

The first MIB in the output is SNMPv2-MIB, which contains information about the system itself, including the processor type, operating system, system availability time, and some more specific identifying information for this particular system.

The second MIB is the IF-MIB, which provides data about each of the network interfaces that this system has, such as frame counts, and error counts.

The third MIB is the IP-MIB, which describes the Internet Protocol (IP) level configuration of the host system including: packet counts, packet errors, IP addresses, and routing information.

Lastly, notice the error message at the end of the snmpwalk output. IBM z/VM implements SNMP Version 1, which cannot represent certain valid routing tables correctly within the

SNMP version 1 data structure. In this case, there are two routing table entries for the same logical network causing a loop to appear in the SNMP data tree; hence the error message. Because we are interested only in the data from the VSWITCHes rather than the data from z/VM itself, this is not a problem. To overcome the snmpwalk routing loop, snmpwalk must be instructed to start searching the tree elsewhere. The next example illustrates this.

The System MIB is rooted at 1.3.6.1.2.1 which is interpreted as: {iso(1) identified-organization(3) dod(6) internet(1) mgmt(2) mib-2(1)}. Check here: http://www.oid-info.com/get/1.3.6.1.2.1 for an easy to navigate example of the SNMP tree. The VSWITCH data is represented using the BRIDGE MIB, which is rooted at 1.3.6.1.2.1.17 {iso(1) identified-organization(3) dod(6) internet(1) mgmt(2) mib-2(1) bridge(17)}.

```
[root@litnetm1 ~]# snmpwalk -t 10 -c TICLNET -v 1 192.168.70.26 1.3.6.1.2.1.17
BRIDGE-MIB::dot1dBaseBridgeAddress.0 = Hex-STRING: 02 09 00 00 00 03
BRIDGE-MIB::dot1dBaseNumPorts.0 = INTEGER: 24
BRIDGE-MIB::dot1dBaseType.0 = INTEGER: transparent-only(2)
BRIDGE-MIB::dot1dBasePort.1 = INTEGER: 1
BRIDGE-MIB::dot1dBasePort.65 = INTEGER: 65
BRIDGE-MIB::dot1dBasePort.66 = INTEGER: 66
BRIDGE-MIB::dot1dBasePort.67 = INTEGER: 67
BRIDGE-MIB::dot1dBasePortIfIndex.1 = INTEGER: 1
BRIDGE-MIB::dot1dBasePortIfIndex.65 = INTEGER: 65
BRIDGE-MIB::dot1dBasePortIfIndex.66 = INTEGER: 66
BRIDGE-MIB::dot1dBasePortIfIndex.67 = INTEGER: 67
BRIDGE-MIB::dot1dBasePortCircuit.1 = OID: SNMPv2-SMI::zeroDotZero
BRIDGE-MIB::dot1dBasePortCircuit.65 = OID: SNMPv2-SMI::zeroDotZero
BRIDGE-MIB::dot1dBasePortCircuit.66 = OID: SNMPv2-SMI::zeroDotZero
BRIDGE-MIB::dot1dBasePortCircuit.67 = OID: SNMPv2-SMI::zeroDotZero
BRIDGE-MIB::dot1dBasePortDelayExceededDiscards.1 = Counter32: 0
BRIDGE-MIB::dot1dBasePortDelayExceededDiscards.65 = Counter32: 0
BRIDGE-MIB::dot1dBasePortDelayExceededDiscards.66 = Counter32: 0
BRIDGE-MIB::dot1dBasePortDelayExceededDiscards.67 = Counter32: 0
BRIDGE-MIB::dot1dBasePortMtuExceededDiscards.1 = Counter32: 0
BRIDGE-MIB::dot1dBasePortMtuExceededDiscards.65 = Counter32: 0
BRIDGE-MIB::dot1dBasePortMtuExceededDiscards.66 = Counter32: 0
BRIDGE-MIB::dot1dBasePortMtuExceededDiscards.67 = Counter32: 0
BRIDGE-MIB::dot1dTpLearnedEntryDiscards.0 = Counter32: 0
BRIDGE-MIB::dot1dTpAgingTime.0 = INTEGER: 1000000
BRIDGE-MIB::dot1dTpFdbAddress.'......' = Hex-STRING: 02 09 00 00 00 0D
BRIDGE-MIB::dot1dTpFdbAddress.'......' = Hex-STRING: 02 09 00 00 00 0E
BRIDGE-MIB::dot1dTpFdbAddress.'......' = Hex-STRING: 02 09 00 00 00 12
BRIDGE-MIB::dot1dTpFdbPort.'......' = INTEGER: 67
BRIDGE-MIB::dot1dTpFdbPort.'......' = INTEGER: 65
```

```
BRIDGE-MIB::dot1dTpFdbPort.'......' = INTEGER: 66
BRIDGE-MIB::dot1dTpFdbStatus.'......' = INTEGER: learned(3)
BRIDGE-MIB::dot1dTpFdbStatus.'......' = INTEGER: learned(3)
BRIDGE-MIB::dot1dTpFdbStatus.'......' = INTEGER: learned(3)
BRIDGE-MIB::dot1dTpPort.1 = INTEGER: 1
BRIDGE-MIB::dot1dTpPort.65 = INTEGER: 65
BRIDGE-MIB::dot1dTpPort.66 = INTEGER: 66
BRIDGE-MIB::dot1dTpPort.67 = INTEGER: 67
BRIDGE-MIB::dot1dTpPortMaxInfo.1 = INTEGER: 9152
BRIDGE-MIB::dot1dTpPortMaxInfo.65 = INTEGER: 65472
BRIDGE-MIB::dot1dTpPortMaxInfo.66 = INTEGER: 65472
BRIDGE-MIB::dot1dTpPortMaxInfo.67 = INTEGER: 65472
BRIDGE-MIB::dot1dTpPortInFrames.1 = Counter32: 180755120
BRIDGE-MIB::dot1dTpPortInFrames.65 = Counter32: 17395512
BRIDGE-MIB::dot1dTpPortInFrames.66 = Counter32: 17238306
BRIDGE-MIB::dot1dTpPortInFrames.67 = Counter32: 22867486
BRIDGE-MIB::dot1dTpPortOutFrames.1 = Counter32: 228351243
BRIDGE-MIB::dot1dTpPortOutFrames.65 = Counter32: 6487738
BRIDGE-MIB::dot1dTpPortOutFrames.66 = Counter32: 2554372
BRIDGE-MIB::dot1dTpPortOutFrames.67 = Counter32: 13075687
BRIDGE-MIB::dot1dTpPortInDiscards.1 = Counter32: 7
BRIDGE-MIB::dot1dTpPortInDiscards.65 = Counter32: 0
BRIDGE-MIB::dot1dTpPortInDiscards.66 = Counter32: 0
BRIDGE-MIB::dot1dTpPortInDiscards.67 = Counter32: 0
```

The VSWITCH that provided this data was configured with three z/VM guests coupled using one Network Interface Card (NIC) each. The guests couple to the VSWITCH in sequential order, starting at logical port number 65. A VSWITCH's primary OSA connection plugs into logical port number 1. The secondary OSA connection plugs into logical port number 2, and so on, up to logical port number 8 for the last possible OSA connection.

Note that there is no indication of which logical guest port is related to which guest. This data cannot be represented using SNMP version 1. Unfortunately, this is not something that can be easily scripted in REXX™ to be run from a CMS service machine either. Relating a logical VSWITCH port number to a guest is valid only for the instant that the command is run, because guests uncouple themselves from and then couple themselves to the VSWITCH using the lowest available port number. There is no way to force a guest to couple to a specific port on the VSWITCH.

What then, is useful about the SNMP data? Mostly, the data for port numbers 1 through 8. This data describes the VSWITCH logical uplink to the rest of the physical network hardware. These

intra-switch uplinks are usually the first objects of interest in an SNMP monitoring implementation, and therefore this data is very useful.

The SNMP daemon on z/VM will also send out link state change notifications, in addition to the data provided by snmpwalk. If the primary link from a monitored VSWITCH to its OSA goes down, SNMP will send a notification to all the configured network monitoring stations that port number 1 on the VSWITCH has gone down.

This function requires z/VM to have more than one network adapter available so that it can transmit notifications as interfaces go down and come back up.

A recipient of these messages, such as snmptrapd on Linux, can log these notifications (also called traps), to the /var/log/messages file:

```
Sep  1 11:13:26 litnetm1 snmptrapd[1178]: 2009-09-01 11:13:26 192.168.70.28(via UDP: [192.168.71.15]:161)
TRAP, SNMP v1, community LTICVM9  IF-MIB::ifIndex Link Down Trap (0) Uptime: 5 days, 15:36:55.00
IF-MIB::ifIndex = INTEGER: 1

Sep  1 11:13:31 litnetm1 snmptrapd[1178]: 2009-09-01 11:13:31 192.168.70.28(via UDP: [192.168.71.15]:161)
TRAP, SNMP v1, community LTICVM9  IF-MIB::ifIndex Link Up Trap (0) Uptime: 5 days, 15:36:59.00    IF-
MIB::ifIndex = INTEGER: 1
```

Note that the message timestamps show the interval between the link going down and coming back up, as well as the NODEID of the VM system (LTICVM9 in this case), and the “ifIndex” of the port affected. We care only about port numbers 1 through 8 in our implementation, because we do not yet have a way to map ports 65 and up to specific guests dynamically.

z/VM Setup

The SNMP daemon has been a part of z/VM for quite some time. Historically it was used to interface with the NetView® monitoring system. With z/VM Release 5.3, an SNMP subagent enables VSWITCH data retrieval through SNMP.

There is a chapter in the z/VM TCP/IP Planning and Customization book that describes "Configuring the SNMP Servers" as well as scattered references to "Setting up an SNMP Subagent". The four steps we performed are listed in order here. Some may already be completed for you, depending on which release of z/VM you are running and your specific deployment.

1) Apply APAR VM64646 and reIPL z/VM.
   This APAR fixes two problems with the VSWITCH support, which would otherwise lead to abends in the subagent code and incorrect output.
2) Configure the z/VM TCPIP PROFILE to allow the SNMP daemon to answer requests and configure the VSWITCHes for SNMP monitoring.

   First and foremost – to monitor VSWITCHes with SNMP you must have one pair of DEVICE and LINK statements for each VSWITCH. Unfortunately, there is no way using the SNMP Bridge MIB to represent a device that contains multiple logical switches. If you have five VSWITCHes to monitor, five pairs of DEVICE and LINK statements are needed.

   As of z/VM 5.4, you can have z/VM TCP/IP use NICs which are coupled to VSWITCHes, so that you do not have to use real OSA devices if they are in short supply.

   2.1) An example of the DEVICEs and LINKs to use for SNMP monitoring:

```
DEVICE  DEVETH4    OSD      0800
LINK    LNKETH4    QDIOETHERNET  DEVETH4
;
DEVICE  DEVETH5    OSD      0804
LINK    LNKETH5    QDIOETHERNET  DEVETH5
;
DEVICE  DEVETH6    OSD      0808
LINK    LNKETH6    QDIOETHERNET  DEVETH6
;
DEVICE  DEVETH7    OSD      080C
LINK    LNKETH7    QDIOETHERNET  DEVETH7
;
DEVICE  DEVETH8    OSD      0810
LINK    LNKETH8    QDIOETHERNET  DEVETH8
```

   In this example, each new OSA interface is using the same OSA port. In a production configuration, it is recommended to distribute these interfaces across all the available ports that the LPAR has access to.

Make sure that the device used to monitor a VSWITCH is on a different OSA port than the VSWITCH provides service through! If a VSWITCH is monitored using the same physical OSA port that provides the VSWITCH service, then the SNMP daemon will not be able to send a notification if the whole port loses connectivity.

2.2) To insert identifying data into the SNMP-v2 MIB, add the following sections to TCPIP PROFILE:

```
SYSCONTACT
Bob Admin     (TL 555-1122)
Gary Sysprog (TL 555-1133)
Department of Redundancy Department
ENDSYSCONTACT
SYSLOCATION
BUILDINGA
123 Fake Street
Poughkeepsie, NY 12601
ENDSYSLOCATION
```

2.3) Enable TCP/IP to start the SNMP daemon automatically by adding this line to the AUTOLOG section of TCPIP PROFILE:

```
SNMPD     password    ; SNMP VM Agent Virtual Machine
```

2.4) Allow the SNMP daemon to bind to UDP port number161, by adding this to the PORT section of TCPIP PROFILE:

```
161  UDP SNMPD                ; SNMP Agent
```

2.5) The VSWITCH parameter in the HOME configuration binds the specified IP address to that VSWITCH as a management interface. The management operation of that IP address is in addition to the normally provided z/VM TCP/IP services. In the following example, z/VM itself is a single SNMP device that contains five logical switches. Define the management IP addresses of the VSWITCHes by editing the HOME section of TCPIP PROFILE:

```
192.168.70.24                  VSWITCH DT70TAG  LNKETH4
192.168.70.25                  VSWITCH PRVV68   LNKETH5
192.168.70.26                  VSWITCH PRVV71L2 LNKETH6
192.168.70.27                  VSWITCH PRVV74L2 LNKETH7
192.168.70.28                  VSWITCH 9DOTTAG  LNKETH8
```

If an SNMP management station queries 192.168.70.24 for the Bridge MIB data, it will receive only data pertaining to the DT70TAG VSWITCH. This is again due to the SNMP specification not being able to represent multiple logical switches inside a single device.

Configure TCP/IP to automatically start the interfaces by adding the start statements at the end of TCPIP PROFILE:

```
START DEVETH4
START DEVETH5
START DEVETH6
START DEVETH7
START DEVETH8
```

3) Configure the SNMPD service machine.

3.1) Edit the SYSTEM DTCPARMS file to add the following sections:

```
:nick.SNMPD :type.SERVER :class.snmp
```

```
:owner.TCPMAINT
:parms.-s SNMPSUBA
:nick.SNMPSUBA :type.SERVER :class.snmp_agent
:owner.TCPMAINT
:parms.-u SNMPD
```

This will allow the SNMPD to be aware of and start its SNMP subagent machine (and vice versa). Make sure that the SNMPD guest has the disk with this file accessed, such as the TCPMAINT 198 disk.

3.2) This step sets up part of the authorization system for the SNMP daemon. Only systems or networks listed in PW SRC are allowed to query SNMP data. In this case, hosts 192.168.71.249, 192.168.71.48, and 192.168.71.49 are allowed to query SNMP data, and only if they use the community name TICLNET. You are also able to grant authority to an entire network by entering the correct network address and netmask instead of a hosts IP address and all 255's. Make sure that the SNMPD guest has the disk with this file accessed, such as the TCPMAINT 198 disk.

Create a PW SRC file containing the SNMP community names and management nodes that are allowed to talk to SNMPD:

```
TICLNET 192.168.71.249 255.255.255.255
TICLNET 192.168.71.48  255.255.255.255
TICLNET 192.168.71.49  255.255.255.255
```

3.3) This part sets up the list of hosts to send SNMP traps when links go up and down. Make sure that this file is on a disk that SNMPD has access to, such as the TCPMAINT 198 disk.
Create a SNMPTRAP DEST file that contains something like the following:

```
192.168.71.249 UDP
192.168.71.48  UDP
192.168.71.49  UDP
```

3.4) Make sure that the MIB_EXIT DATA file exists on TCPMAINT's 198 disk. If not, copy and rename MIB_EXIT SDATA from TCPMAINT's 591 disk. This file describes the subagent and the data it will provide to SNMPD.

3.5) Copy and rename MIBX2DSC SAMPEXEC from TCPMAINT's 592 disk to MIBX2DSC EXEC on TCPMAINT's 592 disk. This exec will be used to copy some data from the MIB_EXIT DATA file to the MIB_DESC DATA file in the next step.

3.6) Make sure that the MIB_DESC DATA file exists on TCPMAINT's 198 disk. If not, copy and rename MIB_DESC SDATA from TCPMAINT's 591 disk. Then use the MIBX2DSC EXEC to copy some of the MIB data from the MIB_EXIT DATA file to the MIB_DESC DATA file. If both DATA files are on the L disk, the command would look like this:

```
MIBX2DSC  MIB_EXIT DATA L   MIB_DESC DATA L
```

You can verify that the MIBX2DSC exec worked correctly by opening MIB_DESC DATA in xedit and paging all the way to the end. If you see a section marked “The following entries were extracted from MIB_EXIT DATA on <somedate> at <sometime> using the MIB2DSX EXEC” followed by many Object Ids (OIDs) that start with 1.3.6.1.2.1.17, then it worked correctly.

4) Configure the SNMP subagent service machine (SNMPSUBA). The subagent runs the exits defined in MIB_EXIT DATA in response to queries that come through SNMPD, and returns the results back to SNMPD to transmit back to the requester.

4.1) Make sure that the SNMPSUBA guest has class E privileges. Ensure than SNMPSUBA can access the disks where MIB_EXIT DATA and MIB_DESC DATA are, as well as SNMPBRGX TEXT. Also check that a PROFILE EXEC is accessible on the A disk. If not, you can just copy the one from SNMPD.

The SNMPSUBA service machine uses the same config files as SNMPD does.

At this point, you can force the SNMPD and SNMPSUBA guests to logoff, then XAUTOLOG SNMPD. You should see SNMPD start. SNMPD will then xautolog SNMPSUBA in turn. On completion the SNMP support in z/VM is up and running.

SNMP Monitor data

Once you have the SNMPD and SNMPSUBA running, use the snmpwalk command from a Linux system to test that it is all working correctly. You should be able to reproduce the data from the two earlier snmpwalk examples.

The SNMP monitor data can be polled by any SNMP network management station. The network management station can keep the SNMP data and chart it over time for performance tracking, as well as sending SNMP alerts up to higher level systems management tools.

A follow on paper covering the SNMP network management station is expected in 2010.

IBM Systems and Technology Group
Route 100
Somers, New York 10589
U.S.A.
Produced in the United States of America,
01/2010

ZSW03145-USEN-00