Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

M. MacFaden
VMware Inc.
J. Schoenwaelder
Jacobs University
T. Tsou
Huawei Technologies (USA)
C. Zhou
Huawei Technologies

July 16, 2012

Definition of Managed Objects for Virtual Machines Controlled by a Hypervisor
draft-schoenw-opsawg-vm-mib-01

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it defines objects for managing virtual machines controlled by a hypervisor.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Table of Contents

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols. In particular, it defines objects for managing virtual machines controlled by a hypervisor.

The design of this MIB module has been derived from enterprise specific MIB modules, namely a MIB module for managing guests of the XEN hypervisor, a MIB module for managing virtual machines controlled by the VMware hypervisor, and a MIB module using the libvirt programming interface to access different hypervisors.

2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

3. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

4. Overview

The MIB module is organized into a group of scalars and tables. The scalars below vmHypervisor provide basic information about the hypervisor. The vmGuestTable lists the guests (virtual machines) that are known to the hypervisor. The vmStorageTable and the vmIfTable provide the mapping of logical storage areas and network interfaces to virtual machines.

The GuestState textual convention defines a state model for virtual machines. Events causing transitions between major states will cause the generation of notifications (vmGuestStarted, vmGuestStopped, vmGuestSuspended, vmGuestResumed).

The MIB module provides a few writable objects that can be used to make non-persistent changes, e.g., changing the memory allocation or the CPU allocation. It is not the goal of this MIB module to provide a configuration interface for virtual machines since other protocols and data modeling languages are more suitable for this task.

The OID tree structure of the MIB module is shown below.

```
--vmMib(1.3.6.1.2.1.XXXX)
  +--vmNotifications(0)
  |  +--vmGuestStarted(1) [vmGuestName,vmGuestUUID,vmGuestState]
  |  +--vmGuestStopped(2) [vmGuestName,vmGuestUUID,vmGuestState]
  |  +--vmGuestSuspended(3) [vmGuestName,vmGuestUUID,vmGuestState]
  |  +--vmGuestResumed(4) [vmGuestName,vmGuestUUID,vmGuestState]
  +--vmObjects(1)
     +--vmHypervisor(1)
     |  +-- r-n SnmpAdminString vmHypervisorVersion(1)
     +--vmGuestTable(2)
     |  +--vmGuestEntry(1) [vmGuestIndex]
     |     +-- --- GuestIndex      vmGuestIndex(1)
     |     +-- r-n SnmpAdminString vmGuestName(2)
     |     +-- r-n UUIDOrZero      vmGuestUUID(3)
     |     +-- r-n GuestState      vmGuestState(4)
     |     +-- r-n SnmpAdminString vmGuestOS(6)
     |     +-- r-n Unsigned32      vmGuestCurCPUs(7)
     |     +-- rwn Unsigned32      vmGuestMinCPUs(8)
     |     +-- rwn Unsigned32      vmGuestMaxCPUs(9)
     |     +-- r-n KBytes          vmGuestCurMem(10)
     |     +-- rwn KBytes          vmGuestMinMem(11)
     |     +-- rwn KBytes          vmGuestMaxMem(12)
     |     +-- r-n Unsigned32      vmGuestCPUTime(13)
     +--vmStorageTable(3)
     |  +--vmStorageEntry(1) [vmGuestIndex,vmStorageIndex]
     |     +-- --- GuestIndexOrZero vmStorageGuest(1)
     |     +-- --- StorageIndex     vmStorageIndex(2)
     |     +-- r-n SnmpAdminString  vmStorageName(3)
     +--vmIfTable(4)
        +--vmIfEntry(1) [vmGuestIndex,vmIfIndex]
           +-- --- GuestIndexOrZero vmIfGuest(1)
           +-- --- InterfaceIndex   vmIfIndex(2)
           +-- r-n PhysAddress      vmIfPhysAddr(3)
```

5. Relationship to Other MIB Modules

The MIB module IMPORTS definitions from SNMPv2-SMI [RFC2578], SNMPv2-TC [RFC2579], SNMPv2-CONF [RFC2580], SNMP-FRAMEWORK-MIB [RFC3411], and IF-MIB [RFC2863].

Hypervisors implementing this MIB module should implement the HOST-RESOURCES-MIB [RFC2790] and the IF-MIB [RFC2863] in order to export information about the resources (e.g., processors, memory, logical storage devices, network interfaces) of the physical machine. If the hypervisor emulates a bridge to network virtual machines, then it should implement the IEEE8021-BRIDGE-MIB. (Note that the BRIDGE-MIB defined in [RFC4188] is now further maintained by the IEEE [RFC4663].) Details of the hardware configuration of a physical machine can be made available by implementing the ENTITY-MIB [RFC4133].

5.1. Relationship to the HOST-RESOURCES-MIB

The HOST-RESOURCES-MIB implemented on the physical machine provides information about the number of CPUs and the amount of memory available. Furthermore, the HOST-RESOURCES-MIB provides information about logical storage devices.

The MIB module defined in this memo provides a mapping of logical storage devices to virtual machines. Further details about the storage devices (such as the size and the amount of allocated storage) is provided by the HOST-RESOURCES-MIB. Note that the number of storage types can be extended through the IANA maintained HOST-RESOURCES-TYPES MIB module.

5.2. Relationship to the IF-MIB

The MIB module provides a mapping of network interfaces to virtual machines. Further details about the network interfaces (such as statistics about the number of packets/bytes sent or received) can be obtained from the IF-MIB.

5.3. Relationship to the IEEE8021-BRIDGE-MIB

Hypervisors implementing virtual bridges should export the bridging topologies by implementing the IEEE8021-BRIDGE-MIB. For backwards compatibility with existing management applications, they may also choose to implement the BRIDGE-MIB [RFC4188].

5.4. Relationship to the ENTITY-MIB

The ENTITY-MIB [RFC4133] describes managed objects used for managing multiple logical and physical entities managed by a single SNMP agent. Implementations of the MIB module defined in this document may want to use the ENTITY-MIB to provide the logical to physical entity mapping and if needed to point to the agent in the virtual machine and vice versa.

6. Definitions

```
VM-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    Integer32, Unsigned32, mib-2
        FROM SNMPv2-SMI                                 -- RFC 2578
    TEXTUAL-CONVENTION, PhysAddress
        FROM SNMPv2-TC                                  -- RFC 2579
    OBJECT-GROUP, NOTIFICATION-GROUP, MODULE-COMPLIANCE
        FROM SNMPv2-CONF                                -- RFC 2580
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB                         -- RFC 3411
    InterfaceIndex
        FROM IF-MIB;                                    -- RFC 2863

vmMib MODULE-IDENTITY
    LAST-UPDATED "201203150000Z"
    ORGANIZATION
        "Jacobs University Bremen"
    CONTACT-INFO
        "Michael MacFaden
         VMware Inc.
         Email: mrm@vmware.com

         Juergen Schoenwaelder
         Jacobs University Bremen
         Email: j.schoenwaelder@jacobs-university.de

         Tina Tsou
         Huawei Technologies (USA)
         Email: tina.tsou.zouting@huawei.com

         Cathy Zhou
         Huawei Technologies
         Email: cathyzhou@huawei.com"
    DESCRIPTION
        "The MIB module for monitoring virtual machines controlled
         by a hypervisor.

         Copyright (c) 2012 IETF Trust and the persons identified as
         authors of the code.  All rights reserved.

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject
         to the license terms contained in, the Simplified BSD
         License set forth in Section 4.c of the IETF Trust's
```

```
          Legal Provisions Relating to IETF Documents
          (http://trustee.ietf.org/license-info)."

    REVISION "201203150000Z"
    DESCRIPTION
        "Initial version, published as RFC XXXX."
    -- RFC Ed.: replace XXXX with actual RFC number & remove this note
    ::= { mib-2 XXXX }

vmNotifications OBJECT IDENTIFIER ::= { vmMib 0 }
vmObjects       OBJECT IDENTIFIER ::= { vmMib 1 }
vmConformance   OBJECT IDENTIFIER ::= { vmMib 2 }

-- Textual convention definitions:

GuestIndex ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "A unique value, greater than zero, identifying a virtual
         machine. The value for each virtual machine must remain
         constant at least from one re-initialization of the
         hypervisor to the next re-initialization."
    SYNTAX      Integer32 (1..2147483647)

GuestIndexOrZero ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "This textual convention is an extension of the VmGuestIndex
         convention.  This extension permits the additional value
         of zero.  The meaning of the value zero is object-specific
         and must therefore be defined as part of the description of
         any object which uses this syntax.  Examples of the usage of
         zero might include situations where a virtual machine is
         unknown, or when none or all virtual machines need to be
         referenced."
    SYNTAX      Integer32 (0..2147483647)

StorageIndex ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "A unique value, greater than zero, identifying a logical
         storage area. The value for each logical storage area must
         remain constant at least from one re-initialization of the
         hypervisor to the next re-initialization."
    SYNTAX      Integer32 (1..2147483647)
```

```
 UUID ::= TEXTUAL-CONVENTION
     DISPLAY-HINT "4x-2x-2x-2x-6x"
     STATUS      current
     DESCRIPTION
         "The Universally Unique IDentifier (UUID) identifying a
          virtual machine. The UUID format is defined in RFC 4122."
     REFERENCE
         "RFC4122: A Universally Unique IDentifier (UUID) URN Namespace"
     SYNTAX      OCTET STRING (SIZE (16))

 UUIDOrZero ::= TEXTUAL-CONVENTION
     DISPLAY-HINT "4x-2x-2x-2x-6x"
     STATUS      current
     DESCRIPTION
         "The Universally Unique IDentifier (UUID) identifying a
          virtual machine or a zero-length string. The UUID format is
          defined in RFC 4122. The meaning of the zero-length string is
          object-specific and must therefore be defined as part of the
          description of any object which uses this syntax."
     SYNTAX      OCTET STRING (SIZE (0|16))

 GuestState ::= TEXTUAL-CONVENTION
     STATUS      current
     DESCRIPTION
         "The state of a guest (virtual machine):

          unknown(1)     The state is unknown, e.g., because the
                         implementation failed to obtain the state
                         from the hypervisor.

          other(2)       The state has been obtained but it is
                         not a known state.

          running(3)     The virtual machine is currently running.

          blocked(4)     The virtual machine is currently blocked.

          paused(5)      The virtual machine is currently paused.

          migrating(6)   The virtual machine is currently migrating.

          shutdown(7)    The virtual machine is currently in the
                         process of shutting down.

          shutoff(8)     The virtual machine is down.

          crashed(9)     The virtual machine has crashed."
     SYNTAX      INTEGER {
```

```
            unknown(1),
            other(2),
            running(3),
            blocked(4),
            paused(5),
            migrating(6),
            shutdown(7),
            shutoff(8),
            crashed(9)
    }

KBytes ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "Storage size measured in units of 1024 octets (bytes). This
         textual convention allows to represent storage sizes up to
         4096 gigabytes."
    SYNTAX Unsigned32

-- Object definitions

vmHypervisor OBJECT IDENTIFIER ::= { vmObjects 1 }

vmHypervisorVersion OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The version string indicating the version of the hypervisor
        running on the physical host."
    ::= { vmHypervisor 1 }

    -- The number of CPUs and the amount of memory can be found
    -- in the objects of the HOST-RESOURCES-MIB

vmGuestTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VmGuestEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A (conceptual) table of all guests (virtual machines)
         on the physical host."
    ::= { vmObjects 2 }

vmGuestEntry OBJECT-TYPE
    SYNTAX      VmGuestEntry
    MAX-ACCESS  not-accessible
```

```
    STATUS      current
    DESCRIPTION
        "An (conceptual) table entry describing a particular
         guest (virtual machine)."
    INDEX   { vmGuestIndex }
    ::= { vmGuestTable 1 }

VmGuestEntry ::= SEQUENCE {
    vmGuestIndex        GuestIndex,
    vmGuestName         SnmpAdminString,
    vmGuestUUID         UUIDOrZero,
    vmGuestState        GuestState,
-- XXX add information about the CPU type
-- XXX the cpu type may be different from the host CPU
    vmGuestOS           SnmpAdminString,
    vmGuestCurCPUs      Unsigned32,
    vmGuestMinCPUs      Unsigned32,
    vmGuestMaxCPUs      Unsigned32,
    vmGuestCurMem       KBytes,
    vmGuestMinMem       KBytes,
    vmGuestMaxMem       KBytes,
    vmGuestCPUTime      Unsigned32
}

vmGuestIndex OBJECT-TYPE
    SYNTAX      GuestIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A unique value identifying a guest (virtual machine)."
    ::= { vmGuestEntry 1 }

vmGuestName OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The name of this guest (virtual machine)."
    ::= { vmGuestEntry 2 }

vmGuestUUID OBJECT-TYPE
    SYNTAX      UUIDOrZero
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A UUID identifying this guest (virtual machine). The UUID
         is expected to be a long-term persistent identifier and
         to remain the same across reboots of the virtual machines
```

```
          and the hypervisor. The zero-length string is returned
          in case a virtual machine does not have a suitable
          persistent UUID."
      ::= { vmGuestEntry 3 }

  vmGuestState OBJECT-TYPE
      SYNTAX      GuestState
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "The current operational state of the guest (virtual
          machine)."
      ::= { vmGuestEntry 4 }

  vmGuestOS OBJECT-TYPE
      SYNTAX      SnmpAdminString
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "The operating system running on this guest (virtual
          machine). This value corresponds to the operating
          system the hypervisor assumes to be running when the
          virtual machine is started. This may differ from the
          actual operating system in case the virtual machine
          boots into a different operating system."
      ::= { vmGuestEntry 6 }

  vmGuestCurCPUs OBJECT-TYPE
      SYNTAX      Unsigned32
      UNITS       "CPUs"
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
          "The number of CPUs currently assigned to this guest
          (virtual machine). Virtual machines that are not
          operational typically have 0 CPUs assigned."
      ::= { vmGuestEntry 7 }

  vmGuestMinCPUs OBJECT-TYPE
      SYNTAX      Unsigned32
      UNITS       "CPUs"
      MAX-ACCESS  read-write
      STATUS      current
      DESCRIPTION
          "The minimum number of CPUs that are assigned to this
          guest (virtual machine) when it is in a running state.
          Changes to this value may not persist across restarts
          of the hypervisor."
```

```
    ::= { vmGuestEntry 8 }

 vmGuestMaxCPUs OBJECT-TYPE
     SYNTAX      Unsigned32
     UNITS       "CPUs"
     MAX-ACCESS  read-write
     STATUS      current
     DESCRIPTION
         "The maximum number of CPUs that are assigned to this
          guest (virtual machine) when it is in a running state.
          The value zero denotes that there is no limit. Changes
          to this value may not persist across restarts of the
          hypervisor."
     ::= { vmGuestEntry 9 }

 vmGuestCurMem OBJECT-TYPE
     SYNTAX      KBytes
     UNITS       "KBytes"
     MAX-ACCESS  read-only
     STATUS      current
     DESCRIPTION
         "The amount of main memory currently assigned to this
          guest (virtual machine). Virtual machines that are not
          operational typically have no memory assigned."
     ::= { vmGuestEntry 10 }

 vmGuestMinMem OBJECT-TYPE
     SYNTAX      KBytes
     UNITS       "KBytes"
     MAX-ACCESS  read-write
     STATUS      current
     DESCRIPTION
         "The minimum amount of main memory that is assigned to
          this guest (virtual machine) when it is in a running
          state. Changes to this value may not persist across
          the restart of the hypervisor."
    ::= { vmGuestEntry 11 }

 vmGuestMaxMem OBJECT-TYPE
     SYNTAX      KBytes
     UNITS       "KBytes"
     MAX-ACCESS  read-write
     STATUS      current
     DESCRIPTION
         "The maximum amount of main memory that can be assigned to
          this guest (virtual machine) when it is in a running state.
          The value zero denotes that there is no limit. Changes to
          this value may not persist across the restart of the
```

```
          hypervisor."
    ::= { vmGuestEntry 12 }

 vmGuestCPUTime OBJECT-TYPE
     SYNTAX      Unsigned32
     UNITS       "seconds"
     MAX-ACCESS  read-only
     STATUS      current
     DESCRIPTION
         "The number of CPU seconds consumed by this guest (virtual
          machine). Note that on a virtual machines with multiple
          CPUs, this value may increment by more than one second
          in a second of real (wall clock) time."
     ::= { vmGuestEntry 13 }

 vmStorageTable OBJECT-TYPE
     SYNTAX      SEQUENCE OF VmStorageEntry
     MAX-ACCESS  not-accessible
     STATUS      current
     DESCRIPTION
         "A (conceptual) table of storage devices attached to
          guests (virtual machines)."
     ::= { vmObjects 3 }

 vmStorageEntry OBJECT-TYPE
     SYNTAX      VmStorageEntry
     MAX-ACCESS  not-accessible
     STATUS      current
     DESCRIPTION
         "An (conceptual) table entry describing a particular
          storage device attached to a guest (virtual machine)"
     INDEX       { vmStorageGuest, vmStorageIndex }
     ::= { vmStorageTable 1 }

 VmStorageEntry ::= SEQUENCE {
     vmStorageGuest      GuestIndexOrZero,
     vmStorageIndex      StorageIndex,
     vmStorageName       SnmpAdminString
 }

 vmStorageGuest OBJECT-TYPE
     SYNTAX      GuestIndexOrZero
     MAX-ACCESS  not-accessible
     STATUS      current
     DESCRIPTION
         "Identifies the guest (virtual machine) this storage has
          been allocated to. The value 0 indicates that the storage
          is currently not allocated to a guest (virtual machine)."
```

```
    ::= { vmStorageEntry 1 }

vmStorageIndex OBJECT-TYPE
    SYNTAX      StorageIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A unique value identifying a logical storage area. On
         systems implementing the HOST-RESOURCES-MIB, the value
         must be the same value that is used as the index into
         the hrStorageTable (hrStorageIndex)."
    ::= { vmStorageEntry 2 }

vmStorageName OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The name of the storage area as seen on the hypervisor."
    ::= { vmStorageEntry 3 }

vmIfTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VmIfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A (conceptual) table of network interfaces attached to
         guests (virtual machines)."
    ::= { vmObjects 4 }

vmIfEntry OBJECT-TYPE
    SYNTAX      VmIfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An (conceptual) table entry describing a particular
         network interface attached to a guest (virtual machine)"
    INDEX       { vmGuestIndex, vmIfIndex }
    ::= { vmIfTable 1 }

VmIfEntry ::= SEQUENCE {
    vmIfGuest      GuestIndexOrZero,
    vmIfIndex      InterfaceIndex,
    vmIfPhysAddr   PhysAddress
}

vmIfGuest OBJECT-TYPE
    SYNTAX      GuestIndexOrZero
```

```
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Identifies the guest (virtual machine) this network interface
         has been allocated to. The value 0 indicates that the network
         interface is currently not allocated to a guest (virtual
         machine)."
    ::= { vmIfEntry 1 }

vmIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The interface index of the network interface under which it
         is known on the system running the hypervisor. If the
         interface is a port of a virtual bridge, then the port
         of the virtual bridge should map to this interface index."
    ::= { vmIfEntry 2 }

vmIfPhysAddr OBJECT-TYPE
    SYNTAX      PhysAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The physical address used by the interface. For interfaces
         associated to a port of a virtual bridge, this object
         normally contains a MAC address. For interfaces which do not
         have such an address, this object should contain a
         zero-length octet string."
    ::= { vmIfEntry 3 }


-- Notification definitions:

vmGuestStarted NOTIFICATION-TYPE
    OBJECTS     {
                  vmGuestName,
                  vmGuestUUID,
                  vmGuestState
                }
    STATUS      current
    DESCRIPTION
        "This notification is generated when a guest (virtual machine)
         has been started and the start process has reached a stable
         state (e.g., running or crashed)."
    ::= { vmNotifications 1 }
```

```
 vmGuestStopped NOTIFICATION-TYPE
     OBJECTS     {
                   vmGuestName,
                   vmGuestUUID,
                   vmGuestState
                 }
     STATUS      current
     DESCRIPTION
         "This notification is generated when a guest (virtual machine)
          has been stopped and the shutdown process has reached a stable
          state (e.g., shutdown or shutoff or crashed)."
     ::= { vmNotifications 2 }

 vmGuestSuspended NOTIFICATION-TYPE
     OBJECTS     {
                   vmGuestName,
                   vmGuestUUID,
                   vmGuestState
                 }
     STATUS      current
     DESCRIPTION
         "This notification is generated when a guest (virtual machine)
          has been suspended and the suspension process has reached a
          stable state (e.g., paused or crashed)."
     ::= { vmNotifications 3 }

 vmGuestResumed NOTIFICATION-TYPE
     OBJECTS     {
                   vmGuestName,
                   vmGuestUUID,
                   vmGuestState
                 }
     STATUS      current
     DESCRIPTION
         "This notification is generated when a guest (virtual machine)
          has been resumed and the resumption process has reached a
          stable state (e.g., running or crashed)."
     ::= { vmNotifications 4 }

 -- Compliance definitions:

 vmGroups      OBJECT IDENTIFIER ::= { vmConformance 1 }
 vmCompliances OBJECT IDENTIFIER ::= { vmConformance 2 }

 vmFullCompliance MODULE-COMPLIANCE
     STATUS      current
     DESCRIPTION
         "Compliance statement for implementations supporting
```

```
          read/write access, according to the object definitions."
      MODULE      -- this module
      MANDATORY-GROUPS {
          vmHypervisorGroup,
          vmGuestGroup,
          vmStorageGroup,
          vmIfGroup,
          vmNotificationGroup
      }
      ::= { vmCompliances 1 }

 vmReadOnlyCompliance MODULE-COMPLIANCE
      STATUS      current
      DESCRIPTION
          "Compliance statement for implementations supporting
           only readonly access."
      MODULE      -- this module
      MANDATORY-GROUPS {
          vmHypervisorGroup,
          vmGuestGroup,
          vmStorageGroup,
          vmIfGroup,
          vmNotificationGroup
      }

      OBJECT vmGuestMinCPUs
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT vmGuestMaxCPUs
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT vmGuestMinMem
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."

      OBJECT vmGuestMaxMem
      MIN-ACCESS  read-only
      DESCRIPTION
          "Write access is not required."
      ::= { vmCompliances 2 }

 vmHypervisorGroup OBJECT-GROUP
      OBJECTS {
```

```
        vmHypervisorVersion
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing insight into the
         hypervisor itself."
    ::= { vmGroups 1 }

vmGuestGroup OBJECT-GROUP
    OBJECTS {
        -- vmGuestIndex,
        vmGuestName,
        vmGuestUUID,
        vmGuestState,
        vmGuestOS,
        vmGuestCurCPUs,
        vmGuestMinCPUs,
        vmGuestMaxCPUs,
        vmGuestCurMem,
        vmGuestMinMem,
        vmGuestMaxMem,
        vmGuestCPUTime
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing insight into the
         guests (virtual machines) controlled by a hypervisor."
    ::= { vmGroups 2 }

vmStorageGroup OBJECT-GROUP
    OBJECTS {
        -- vmStorageGuest,
        -- vmStorageIndex,
        vmStorageName
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing insight into the
         logical storage areas controlled by a hypervisor."
    ::= { vmGroups 3 }

vmIfGroup OBJECT-GROUP
    OBJECTS {
        -- vmIfGuest,
        -- vmIfIndex,
        vmIfPhysAddr
    }
    STATUS      current
```

```
    DESCRIPTION
        "A collection of objects providing insight into the
         network interfaces controlled by a hypervisor."
    ::= { vmGroups 4 }

vmNotificationGroup NOTIFICATION-GROUP
    NOTIFICATIONS {
        vmGuestStarted,
        vmGuestStopped,
        vmGuestSuspended,
        vmGuestResumed
    }
    STATUS      current
    DESCRIPTION
        "A collection of notifications for virtual machines
         controlled by a hypervisor."
    ::= { vmGroups 5 }

END
```

7. Security Considerations

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

- Unauthorized changes to vmGuestMinCPUs, vmGuestMaxCPUs, vmGuestMinMem, and vmGuestMaxMem can significantly slow down virtual machines or prevent the start of new virtual machines.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- The tables vmGuestTable, vmStorageTable, and vmIfTable provide insight into the resources allocated to virtual machines and this knowledge might be exploited for targeted denial of service attacks.

- The vmGuestStarted, vmGuestStopped, vmGuestSuspended, and vmGuestResumed notifications provides information about state changes of virtual machines and implicitly also on which physical hosts virtual machines are located. Furthermore, the generation of fake notifications might trigger false alarms and subsequent actions in a network management system, which can amplify denial of service attacks or simply lead to less efficient resource usage.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

8. IANA Considerations

IANA is requested to assign a value for "XXXX" under the 'mib-2' subtree and to record the assignment in the SMI Numbers registry. When the assignment has been made, the RFC Editor is asked to replace "XXXX" (here and in the MIB module) with the assigned value and to remove this note.

9. Acknowledgements

Thanks to David Black and Robert Story for helpful comments during the development of this specification.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC2790] Waldbusser, S. and P. Grillo, "Host Resources MIB", RFC 2790, March 2000.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC4133] Bierman, A. and K. McCloghrie, "Entity MIB (Version 3)", RFC 4133, August 2005.

10.2. Informative References

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC4188] Norseth, K. and E. Bell, "Definitions of Managed Objects for Bridges", RFC 4188, September 2005.

[RFC4663] Harrington, D., "Transferring MIB Work from IETF Bridge MIB WG to IEEE 802.1 WG", RFC 4663, September 2006.

Appendix A. Open Issues

This file is used to track issues that were discussed during the development of the SMIv2 to YANG translation in the IETF NETMOD working group. This issues covered here concern major design choices; this file does not attempt to track minor clarification requests etc.

To comment on issues on the mailing list, please include the issue number in the subject line of the email message.

* vm-mib-01: storage sizes

The MIB does not provide storage sizes, assuming this is provided by the hrStorageTable of the HOST-RESOURCES-MIB. However, some well known implementations of the HOST-RESOURCES-MIB only report about file systems used by the host system and not file systems residing in files used by virtual machines. Furthermore, the hrStorageTable reports sizes "usable by the requesting entity", "excluding loss due to formatting of file system reference information". For storage provided to virtual machines, this information is often not readily available since all you have is the raw block size.

** Solution #01-01

Provide the storage block sizes as part of the VM-MIB. Provide a pointer to the hrStorageTable on systems that can provide this linkage but allow the pointer to be NULL.

** Resolution

TBD

* vm-mib-02: scaling and caching support

It was mentioned that large data centers are characterized by 100.000 physical hosts running 2.000.000 virtual machines. The NASA is reported with 1.000.000 physical hosts and 60.000.000 virtual machines. Bottom line is that we need to make the MIB module scalable. We can assume up hundreds of VMs running on a single virtual machine.

** Solution #02-01

Add ...LastChange objects to tables so that management applications can easily validate cached information without having to read through potentially larger tables. For the vmGuestTable, we might also provide a ...LastStateChange object so that state changes can be polled with reading a simple scalar.

** Solution #02-02

Make some tables time filtered. Unclear which tables would have to be time filtered.

** Resolution

TBD

* vm-mib-03: virtual cpu type identification

It is necessary to identify the CPU architecture or type since some virtual machine systems can emulate different CPU types.

** Solution #03-01

Provide an IANA controlled enumeration that provides a CPU classification. The problem will be to provide rules about what constitutes a new CPU type and what not.

** Solution #03-02

Use OBJECT IDENTITIES to identify CPU types. Such a distributed enumeration will not achieve a great deal of interoperability and is likely close to #03-03.

** Solution #03-03

Use a string data type and rely on systems to put meaningful information there, perhaps provide guidelines how to structure the CPU type names, e.g. vendor-arch-model(-features)* that is amd-x86_64-opteron or intel-i686-pentium3-vmx-acpi (perhaps using a different separator character since a dash might easily clash). Applications may have to do some normalization across VM-MIB implementations (e.g., regular expression matching) but on the other hand this allows to provide details where necessary.

** Solution #03-04

Following #03-03, we provide ...GuestCpuVendor, ...GuestCpuArch and ...GuestCpuModel objects plus an additional table that provides details about the features of the CPUs used by a certain virtual machine. This essentially breaks the string into a set of separate MIB objects.

** Solution #03-05

Following #03-04, we provide ...GuestCpuVendor, ...GuestCpuArch and ...GuestCpuModel objects plus a string object containing a list of features. This way, things are more compact but still the most important components (vendor, arch, model) are broken out as separate objects.

** Resolution

TBD

* vm-mib-04: physical CPU type identification

VM migration sometimes requires to match physical CPUs and more important also feature sets of physical CPUs.

** Solution #04-01:

Extend the ENTITY-MIB with a new MIB module, say an ENTITY-CPU-MIB, providing an entPhyCPUTable, sparsely augmenting the entPhysicalTable for physical entities with entPhysicalClass = cpu. The entPhyCPUTable would contain information about CPU vendor, CPU architecture, CPU mode, CPU features, clock speeds, etc. (see also vm-mib-03).

** Resolution

TBD

* vm-mib-05: per virtual cpu statistics

It seems to be useful to provide statistics for each virtual CPU. However, it remains unclear what can be expected to be provided by a typical hypervisor implementation. There are a number of things to consider:

a) Reporting the time the virtual CPU has been running (CPU time consumed) seems relatively straight forward.

b) Reporting the current state of a virtual CPU requires to first define a suitable state model that is course grained enough to be useful (otherwise CPU state changes far too quickly to yield meaningful results). Libvirt, for example, has CPU states offline, running, blocked on resource. It is not further defined what blocked on resource really means. Anyway, with a suitable state model, the MIB could provide the time spent in the various CPU states rather than or in addition to the current snapshot state.

c) Reporting the affinity mapping of virtual CPUs to physical CPUs. This, of course, requires to have a representation of physical CPUs.

** Resolution

TBD

* representing networks (vmNetTable)

Not yet well enough understood to write up this issue. ;-)

Authors' Addresses

Michael MacFaden
VMware Inc.

EMail: mrm@vmware.com

Juergen Schoenwaelder
Jacobs University
Campus Ring 1
Bremen 28759
Germany

EMail: j.schoenwaelder@jacobs-university.de

Tina Tsou
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara CA 95050
USA

EMail: tina.tsou.zouting@huawei.com

Cathy Zhou
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

EMail: cathyzhou@huawei.com