INTERNET-DRAFT
draft-ietf-l3vpn-vr-mib-04.txt
Expires: January 2006

Elwin Stelzer
Corona Networks, Inc.

Sam Hancock
ACM Systems

Benson Schliesser
SAVVIS Communications

Joseph Laria (Ed.)
Level Stream Research

July 2005

Virtual Router Management Information Base Using SMIv2

Status of this Memo

This document is a product of the IETF's Layer 3 Virtual Private Network (l3vpn) working group. Comments should be addressed to WG's

mailing list at l3vpn@ietf.org. The charter for l3vpn may be found at http://www.ietf.org/html.charters/l3vpn-charter.html

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP based internets. In particular, it defines objects for managing networks using Virtual Routers (VR).

Table of Contents

1.0 Terminology

This document uses terminology defined in [PPVPN-FW] and [PPVPN-VR].

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

2.0 Introduction

This memo defines a MIB module for the Virtual Router [PPVPN-VR, PPVPN-VR-AS] model of Provider Provisioned VPNs [PPVPN-FW].

Following are the goals, in defining this MIB module:

- To have a means for Service Providers to provision VPN service for subscribers, at the PE device.

- To make the agent-side implementation simple, by not modifying the existing standard MIB modules.

- Define all the gluing tables that are needed toward this end.

3.0 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 module objects are generally accessed through the Simple Network Management (SNMP) Protocol. Objects in a MIB module 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].

4.0 Overview of the Virtual Router MIB Module

4.1 SNMP Contexts for Management for Virtual Routers

There is a need for a single agent to manage multiple Virtual Routers. The Architecture for describing SNMP Management Frameworks [RFC3411] provides a way to support such cases.

Managing multiple virtual routers requires that the PE management plane be subdivided into logical VR management domains. In the VR model of PPVPNs a single PE device may contain many virtual routers. Different management entities SHOULD be able to manage specific virtual routers and associated services. The Service Provider MUST be able to manage all virtual routers and associated services.

Using SNMP contexts to group a collection of management information provides the following benefits:

(1) Uses a standard framework defined by the IETF, allowing the product to remain flexible to all implementations of virtual router devices.

    (a) Use SNMPv2c Community String's

    (b) Use SNMPv3 contextName's

(2) Prevents vendors from having to modify the standard MIBs, allowing the implementation to remain standards compliant.

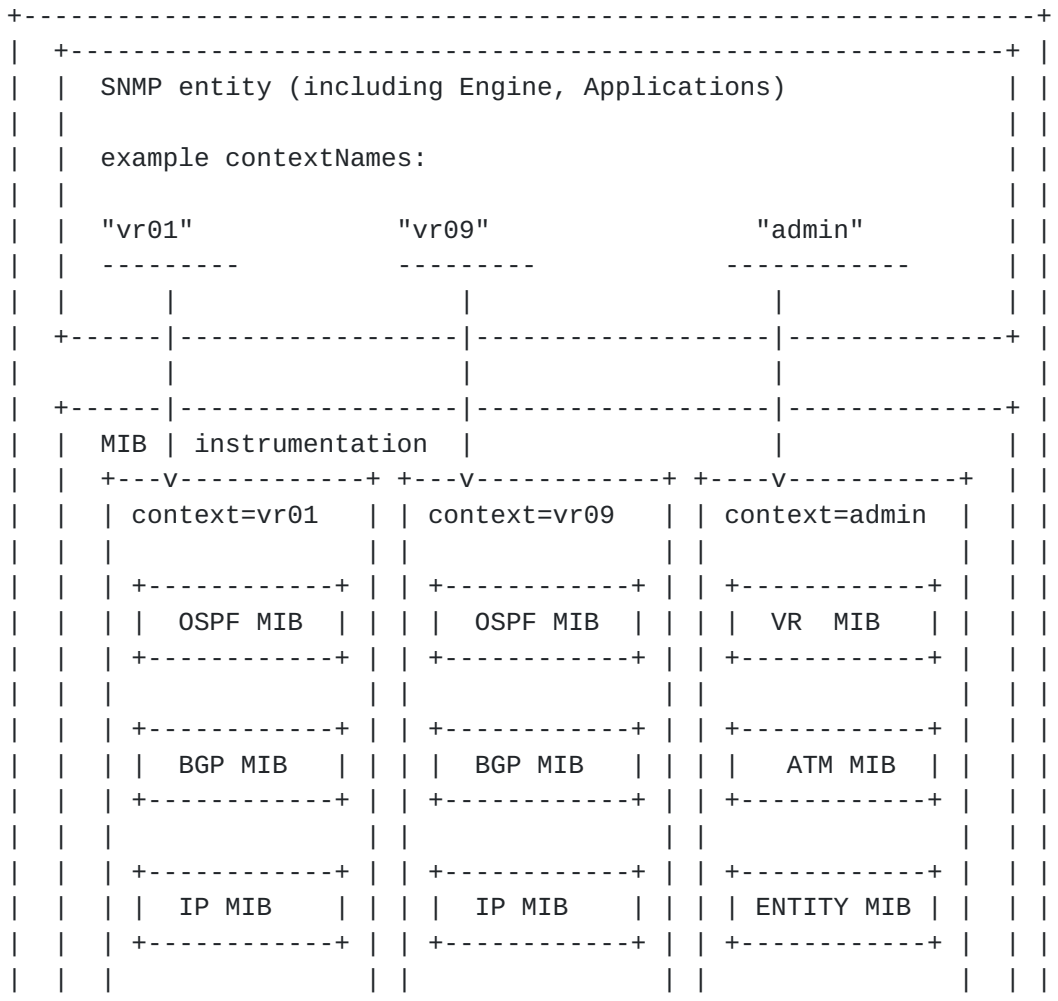
(3) Provides a framework that will work for RIP, OSPF, IS-IS, BGP, IP-FORWARDING, MPLS, and any other MIB module that can be administratively grouped with a VR.

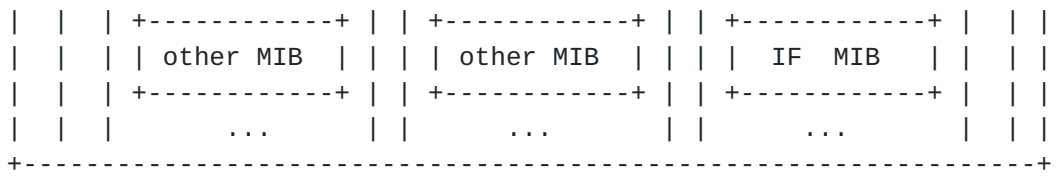
The SNMP context for the Virtual Router instance can be specified in the VrConfigTable. The VrContextName columnar object is used to set the SNMPv2c Community String or the SNMPv3 contextName for a given VR.

A virtual router context represents the set of MIB module objects that could be administratively grouped within a VR. For example, each VR would maintain its own instance of routing protocol MIB module tables. However, the ADMIN context would contain single instances of objects and tables that pertain to system wide configuration such as the Entity, Interfaces, and ATM MIB modules.

A management system using the SNMP context of a particular virtual router MUST be able to manage the virtual router without disrupting other virtual routers in the same PE device.

For example, a PE can be subdivided into two 2 VRs running the OSPF routing protocol. Each VR will maintain a unique instance of the OSPF-MIB. Therefore, the ospfAreaTable of VR-A is distinct from the ospfAreaTable of VR-B.




other MIB
other MIB
IF MIB

Filtering mechanisms based on the SNMP context of a particular virtual router may implemented to allow different management entities to manage those objects and services provisioned the 'Admin' context.

4.2 VR Indexing

While the standard protocol MIB module tables are instantiated in the context specified using SNMP contexts, there may be tables that are defined with the VRID as index.

The VRID is of local significance to a particular PE device, and need not be globally unique. Thus a particular VRID value assigned to a VR in one PE device may indicate a different VR in another PE device.

The VRID has an Unsigned32 value, and this value is assigned by the management station. To aid the management station in assigning a VRID without conflict, the management station can get the 'NextAvailableVRID' from the PE device.

A SNMP manager SHOULD NOT assume global significance of any VRID value other than 0.

For those MIB module tables instantiated in the virtual router context, indexing can only be assumed unique for that particular VR. However those indices in the "ADMIN" context are unique across the entire system, including all VRs.

4.3 Creation and Deletion of VRs

The VR Config Table is used for this purpose. This is a read-create table and adding an entry into this table will create a VR. Removing an entry from this table marks the deletion of a VR.

VRID 0 is assigned to the Administrative VR, which exists by default, and need not be created. Deletion of the Administrative VR will not be permitted. The VRID of the Administrative VR (VRID 0) should be a reserved VRID number. VRID 0 could be termed the "null VR" and it could be the context that manages the resource pool of unattached interfaces. Routing would then not exist in the context of Administrative VR.

4.4 Administrative and Operational Status of VRs

VRs can be administratively turned down. When this is done, no packet forwarding via the VR takes place.

VrOperStatus denotes the operational status of a VR. Currently the VrOperStatus is expected to change along the VrAdminStatus unless an error condition exists.

4.4.1 VR Routing Protocol Trigger

A construct for independently instantiating routing protocol instances for each VR may be useful a solution especially in a PE router where scaling of resources would be necessary.

VrRpTrigger object represents the Routing Protocol (RP) triggers on a VR and is it meant to be used to initiate or shutdown routing protocols on a VR.

4.5 Binding interfaces to a VR

Interfaces are bound to a VR, using vrIfConfigTable. This is a read-write table, and note that interfaces are not created through this table. The vrIfConfigTable MIB module table is used to indicated the relationship between interfaces and virtual router IDs. For each interface present in the system, this table is used to provide the mapping from IfIndex to a unique VR. The table show which interfaces are ?attached or connected? to a virtual router. An interface can not be attached to more than one VR.

The "Admin" VR could be used to manage the resource pool of unattached interfaces. However interfaces would not be attached to VRID 0.

4.6 Setting per VR limits

VRs consume resources on a device, and hence the following parameters defined in vrConfigTable are used to specify an upper bound of resource utilization:

VrMaxRoutes -
Specify the maximum number of routes that will be permitted in VR. This includes all routes, such as the statically configured routes, and the routes learnt via dynamic routing protocols.

4.7 Per VR Statistics

In addition to those statistics available through the VR instantiated MIB module tables, there are some per-VR statistics available through vrStatTable.

4.8 Traps

This memo defines that VrUp and VrDown traps are generated just after VrOperStatus leaves, or just before it enters, the down state, respectively.

(1) A transition into the down state will occur when an error is detected on a VR instance.

(2) Departing the down state generally indicates that the VR is going to up, which is considered a "healthy" state.

An exception to the above generation of VrUp/VrDown traps on changes in VrOperStatus, occurs when an VR is "flapping", i.e., when it is rapidly oscillating between the up and down states. If traps were generated for each such oscillation, the network and the network management system would be flooded with unnecessary traps. In such a situation, the agent should limit the rate at which it generates traps.

This memo defines that enabling and disabling the VR traps is achieved by setting the VrTrapEnable to true(1) or false(2), respectively. By default, this object should have the value true(1).

On some devices where system memory is limited, there may be a need to restrict the maximum number of routes allowed on the system. This memo defines vrMaxRoutesExceeded trap to indicate when vrStatRouteEntries exceeds the maximum limit. There is a danger of notification storms with this type of notification. The definition of vrMaxRoutes is such that vrStatRouteEntries could never exceed it. So whenever vrMaxRoutes has been reached, each new attempt to add a route will cause a new Notification. In order to prevent notification storms of this type, this memo also defines and the enabling and disabling of this trap which is acheived by setting the VrMaxRouteTrapEnable to true(1) or false(2), respectively. By default, this object should have the value true(1).

4.9 Usability Considerations

4.9.1 Multiple Agents

The MIB module is based upon the premise that a single SNMP agent should represent every virtual router on a physical router. An alternative approach would be to deploy a separate SNMP agent for each virtual router. Creating multiple agents for use by the administrator (Service Provider) could be done, for instance by binding to different ports or addresses on the P-node. However from a resource perspective, it is more efficient to use a single agent and multiplex based on the community/context as described in this document. In either case, though, a VR-MIB module is needed to map each VR to its respective agent or context.

There could be a case where a separate agent per VR may be useful, though not as a replacement for the VR-MIB module. If the platform supports instantiation of an agent *within* the VR then the VPN user could query stats, etc., from that agent. This would not be a replacement for the VR-MIB module because (in addition to the above points) the Service Provider may very well not have reachability/connectivity (not to mention uniqueness in addressing) into the VPN. For example, the Service Provider may not have management-network access to the customers' networks.

4.9.2 Provisioning vs. Monitoring

The VR-MIB module goes to some length the support configuration using SNMP. Other MIB modules tend to be for monitoring purposes, with an occasional read-write variable. There is value in having configuration capabilities in this MIB module. The VR-MIB module fills in a gap, allowing for creation of the VR, while the VR context MIB modules allow for configuration of the VR itself. This might prove useful, perhaps even allowing for interoperable management tools.

Some Service Provider may intend to use it only for monitoring. This

is because there may be other mechanisms available to them for configuration of a specific platforms, such as Corba or XML interfaces that they may find more valuable for this function.

5.0 Sample VR MIB module Configuration Scenario

5.1 Creation of a VR

Creating VR instances can be achieved using the following example.

(1) Get the next available Virtual Router Id using the
NextAvailableVrId, to create a VR:

```
Using a context with 'read' access for system level entities.
GetRequest { NextAvailableVrId.0 }
Response   { NextAvailableVrId.0  =  5555 }
```

(2) In VrConfigTable, create VR Instance using VrRowStatus:

```
Using a context with 'read-write' access for system level entities
SetRequest {
    VrRowStatus.5555                createAndGo(4),
    VrName.5555                     "BigTelcoVR",
    VrContextName.5555              "vr5555",
    VrTrapEnable.5555               true(1),
    VrAdminStatus.5555              up(1)
}
```

6.0 Definition of the Virtual Router MIB Module

```
--
-- VIRTUAL-ROUTER-MIB
--

VIRTUAL-ROUTER-MIB DEFINITIONS ::= BEGIN

    IMPORTS
        InterfaceIndex
            FROM IF-MIB
        InetAddressType, InetAddress
            FROM INET-ADDRESS-MIB
-- RFC Ed.: VPN-TC-STD-MIB in Last Call in L3VPN WG
        VPNId
            FROM VPN-TC-STD-MIB
        OBJECT-GROUP, MODULE-COMPLIANCE, NOTIFICATION-GROUP
            FROM SNMPv2-CONF
        Unsigned32, OBJECT-TYPE, MODULE-IDENTITY, TimeTicks,
        NOTIFICATION-TYPE,  mib-2
            FROM SNMPv2-SMI
        TruthValue, DisplayString, RowStatus, TEXTUAL-CONVENTION
            FROM SNMPv2-TC;

    virtualRouterMIB MODULE-IDENTITY
        LAST-UPDATED "200507221200Z"
        ORGANIZATION
            "IETF L3VPN WG"
        CONTACT-INFO

            "
            Elwin Stelzer Eliazer
            Corona Networks, Inc.
            630 Alder Drive
            Milpitas, CA 95035
            USA
            Phone: +1-408-519-3832
            Email: elwinietf@yahoo.com

            Samuel Hancock
            ACM Systems
            3034 Gold Canal Drive
            Rancho Cordova, CA 95670
            USA
            Phone: +1-916-463-7949
            Email: hancoc_s@yahoo.com

            Benson Schliesser
            SAVVIS Communications
```

1 Savvis Parkway
Town and Country, MO 63017
USA
Phone: +1-314-628-7036
Email: bensons@savvis.net

Joseph Laria (Editor)
Level Stream Research
Wilmington, MA 01887
USA
Phone: +1-978-223-9908
Email: jlaria@levelstream.com
"

```
        DESCRIPTION
            "The MIB module is the definition of the managed
            objects for the Virtual Router."
        REVISION "200507221200Z"  -- 22 July 2005 12:00:00 GMT
        DESCRIPTION "Initial version, published as RFC yyyy."
-- RFC Ed.: replace yyyy with actual RFC number & remove this note
        ::= { mib-2 xxxx } -- To be assigned
-- RFC Ed.: replace xxxx with IANA-assigned number & remove this note



--
-- Textual conventions
--


    VrIdentifier ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
            "Virtual Router Identifier.
             VRID 0 is reserved for the Administrative VR
             and cannot be used to create VR's.
            "
        SYNTAX Unsigned32 (0..4294967295)

    VrRpTriggerBitCode ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
            "This object represents Routing Protocol (RP)
             Triggers on a Virtual Router.  The BITS
             represent an Action-code that specifies the
             action on the Routing Protocols.

             The actions are:  initiate or shutdown.

             When encoding the RP using the BITS construct,
             the value is encoded as an OCTET STRING where
             the first bit (bit 0) is the highest bit of the
             octet.

             Bits 0-3 may be specified in any combination to
             allow multiple Routing Protocols to be acted on
             simultaneously or individually.
            "
        SYNTAX BITS {
                 rip (0),
                 ospf(1),
                 bgp (2),
                 isis (3)
```

```
                }
```

```
--
-- Node definitions
--

    vrMIBObjects OBJECT IDENTIFIER ::= { virtualRouterMIB 1 }

    vrConfig OBJECT IDENTIFIER ::= { vrMIBObjects 1 }

    vrConfigScalars OBJECT IDENTIFIER ::= { vrConfig 1 }

    vrConfigNextAvailableVrId OBJECT-TYPE
        SYNTAX VrIdentifier
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The next available Virtual Router Id (index).
            This object provides a hint for the vrID value
            to use when administratively creating a new
            vrConfigEntry.

            A GET of this object returns the next available vrId
            value to be used to create an entry in the associated
            vrConfigTable; or zero, if no valid vrId
            value is available. A value of zero(0) indicates that
            it is not possible to create a new vrConfigEntry
            This object also returns a value of zero when it is the
            lexicographic successor of a varbind presented in an
            SNMP GETNEXT or GETBULK request, for which circumstance
            it is assumed that ifIndex allocation is unintended.

            Successive GETs will typically return different
            values, thus avoiding collisions among cooperating
            management clients seeking to create table entries
            simultaneously.

            Unless specified otherwise by its MAX-ACCESS and
            DESCRIPTION clauses, an object of this type is read-only,
            and a SET of such an object returns a notWritable error."
        ::= { vrConfigScalars 1 }

    vrConfigTable OBJECT-TYPE
        SYNTAX SEQUENCE OF VrConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "This table is for creating the new Virtual Routers."
        ::= { vrConfig 2 }
```

```
    vrConfigEntry OBJECT-TYPE
        SYNTAX VrConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The entries in this table can be added/deleted
            using the vrRowStatus."
        INDEX { vrId }
        ::= { vrConfigTable 1 }

    VrConfigEntry ::=
        SEQUENCE {
            vrId
                VrIdentifier,
            vrRowStatus
                RowStatus,
            vrName
                DisplayString,
            vrContextName
                DisplayString,
            vrTrapEnable
                TruthValue,
            vrMaxRoutes
                Unsigned32,
            vrAdminStatus
                INTEGER,
            vrVpnId
                VPNId,
            vrRpTrigger
                VrRpTriggerBitCode,
            vrMaxRoutesTrapEnable
                TruthValue
        }

    vrId OBJECT-TYPE
        SYNTAX VrIdentifier
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The unique id of this virtual router instance. A Virtual
             Router cannot not be created with vrId = 0.
             VRID 0 is reserved for the Administrative VR.
            "
    ::= { vrConfigEntry 1 }

    vrRowStatus OBJECT-TYPE
        SYNTAX RowStatus
        MAX-ACCESS read-create
```

```
        STATUS current
        DESCRIPTION
            "The status column has three defined values:

            - `active', which indicates that the conceptual row is
            available for use by the managed device;
```

```
            - `createAndGo', which is supplied by a management
            station wishing to create a new instance of a
            conceptual row and to have its status automatically set
            to active, making it available for use by the managed
            device;

            - `destroy', which is supplied by a management station
            wishing to delete all of the instances associated with
            an existing conceptual row."
    ::= { vrConfigEntry 2 }

    vrName OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "The Name of the Virtual Router."
        ::= { vrConfigEntry 3 }

    vrContextName OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "The SNMPv2 Community String or SNMPv3 contextName
            denotes the VR 'context' and is used to logically
            separate the MIB module management."
        ::= { vrConfigEntry 4 }

    vrTrapEnable OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "This objects is used to enable the generation
            of the VrUp and VrDown traps.
                true(1)     - VR Traps Enabled
                false(2)    - VR Traps Disabled"
        DEFVAL { true }
        ::= { vrConfigEntry 5 }

    vrMaxRoutes OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "This object specifies the maximum number of routes that
            this VR can support. The default value is 4 Gig (meaning
            unlimited)."
```

```
        DEFVAL { 4294967295 }
        ::= { vrConfigEntry 6 }
```

```
    vrAdminStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                 up(1),
                 down(2),
                 testing(3),
                 unknown(4)
                }
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "The administrative state of the Virtual Router."
        DEFVAL { down }
        ::= { vrConfigEntry 7 }

    vrVpnId OBJECT-TYPE
        SYNTAX  VPNId
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "The Virtual Private Network Identifier of the Virtual
             Router."
        ::= { vrConfigEntry 8 }

    vrRpTrigger OBJECT-TYPE
        SYNTAX  VrRpTriggerBitCode
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "This object represents Routing Protocol (RP)
             Triggers on a Virtual Router and it meant to
             be used to initiate or shutdown routing
             protocols on a VR.  Multiple RPs can be acted
             on simultaneously.  Also, individual RPs can
             be brought up in steps, which should not
             affect the RPs that were running. The BITS
             represent an Action-code that specifies what
             action is to be performed for the RPs.
             The actions are:  initiate(1) or shutdown(0).

             The running status of an RP shall be available
             in the VR stats table's vrRpStatus, which has
             a similar format, but represents the status.

             Bits 0-3 may be specified in any combination.
             Individual routing protocols may be enabled
             and disabled independently.  Protocols are
             enabled by setting the respective BIT and are
             disabled by resetting the BIT.
```

```
            So, for example, to enable RIP and BGP protocols
            the vrRpTrigger bits 0 and 2 need to be set, and
            as encoded as 10100000.
            All zeros should be interpreted as all protocols
            disable.
           "
        DEFVAL { '00000000'b }
        ::= { vrConfigEntry 9 }
```

```
        vrMaxRoutesTrapEnable  OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "This objects is used to enable the generation
             of the VR Max Routes Exceeded traps.
                true(1)     - VR Max Routes Exceeded Traps Enabled
                false(2)    - VR Max Routes Exceeded Traps Disabled"
        DEFVAL { true }
        ::= { vrConfigEntry 10 }



    vrStat OBJECT IDENTIFIER ::= { vrMIBObjects 2 }

    vrStatScalars OBJECT IDENTIFIER ::= { vrStat 1 }

    vrConfiguredVRs OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of VRs configured on this network element."
        ::= { vrStatScalars 1 }

    vrActiveVRs OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of VRs that are active on the network element.
            These are VRs for which the
            vrStatOperStatus  = up(1)"
        ::= { vrStatScalars 2 }

    vrStatTable OBJECT-TYPE
        SYNTAX SEQUENCE OF VrStatEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "This table contains statistics for the Virtual Router."
        ::= { vrStat 2 }

    vrStatEntry OBJECT-TYPE
        SYNTAX VrStatEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
```

```
            "Entries in this table a per vrId."
        INDEX { vrId }
        ::= { vrStatTable 1 }
```

```
    VrStatEntry ::=
        SEQUENCE {
            vrStatRouteEntries
                Unsigned32,
            vrStatFIBEntries
                Unsigned32,
            vrStatUpTime
                TimeTicks,
            vrOperStatus
                INTEGER,
            vrRpStatus
                VrRpTriggerBitCode,
            vrRouterAddressType
                InetAddressType,
            vrRouterAddress
                InetAddress
         }

    vrStatRouteEntries OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Total number of routes for this VR."
        ::= { vrStatEntry 1 }

    vrStatFIBEntries OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Total number of FIB Entries for this VR."
        ::= { vrStatEntry 2 }

    vrStatUpTime OBJECT-TYPE
        SYNTAX TimeTicks
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The time in (in hundredths of a second) since
            this VR entry has been operational."
        ::= { vrStatEntry 3 }

    vrOperStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                 up(1),
                 down(2),
                 unknown(3)
```

```
            }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The operational status of the Virtual Router."
        ::= { vrStatEntry 4 }
```

```
    vrRpStatus OBJECT-TYPE
        SYNTAX  VrRpTriggerBitCode
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "This object represents the status of Routing
             Protocols on this VR corresponding to the list
             of RP specified in vrRpTrigger.

             The BITS represent an Action-code that specifies
             the status of the RPs.
             The status are:  initiated (1) or shutdown (0).
             Initiated status is indicated when the respective
             BIT value is 1 and indicates shutdown when the
             respective BIT value is 0.

             Bits 0-3 may appear in any combination to
             indicate that RPs may be enabled and disabled
             independently or that multiple RP are acted on
             simultaneously.
            "
        ::= { vrStatEntry 5 }

    vrRouterAddressType OBJECT-TYPE
        SYNTAX  InetAddressType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Router Address Type of this VR."
        ::= { vrStatEntry 6 }

    vrRouterAddress OBJECT-TYPE
        SYNTAX  InetAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Router Address of this VR.  It is derived from one of the
            interfaces. If loopback interface is present, the loopback
            interface address can be used. However, loopback interface
            is optional."
        ::= { vrStatEntry 7 }


    vrIfConfig OBJECT IDENTIFIER ::= { vrMIBObjects 3 }

    vrIfConfigScalars OBJECT IDENTIFIER ::= { vrIfConfig 1 }

    vrIfConfigTable OBJECT-TYPE
        SYNTAX SEQUENCE OF VrIfConfigEntry
```

```
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "This table is for configuring VR Interfaces."
        ::= { vrIfConfig 2 }
```

```
    vrIfConfigEntry OBJECT-TYPE
        SYNTAX VrIfConfigEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Entries in this table correspond to the entries in
            the ifTable that apply to the Virtual Router."
        INDEX { vrId,
                vrIfId }
        ::= { vrIfConfigTable 1 }


    VrIfConfigEntry ::=
        SEQUENCE {
            vrIfId
               InterfaceIndex,
            vrIfConfigRowStatus
               RowStatus
         }

    vrIfId OBJECT-TYPE
        SYNTAX InterfaceIndex
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Virtual Router Interface Index."
        ::= { vrIfConfigEntry 1 }

     vrIfConfigRowStatus  OBJECT-TYPE
        SYNTAX RowStatus
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            " This object is used to create, delete or
            modify a row in this table."
        ::= { vrIfConfigEntry 2 }



-- ****************************************************************
-- Module Traps/Notifications
-- ****************************************************************


    vrNotificationsPrefix OBJECT IDENTIFIER ::= { vrMIBObjects 4 }

    vrNotifications OBJECT IDENTIFIER ::= { vrNotificationsPrefix 0 }
```

```
    vrUp NOTIFICATION-TYPE
        OBJECTS { vrOperStatus }
        STATUS current
        DESCRIPTION
            "This notification is generated when the specified
            VR is about to be initialized or change the VR's
            operational status from down to up."
        ::= { vrNotifications 1 }
```

```
    vrDown NOTIFICATION-TYPE
        OBJECTS { vrOperStatus }
        STATUS current
        DESCRIPTION
            "This notification is generated when the specified
            VR's operational status is about to go down."
        ::= { vrNotifications 2 }

    vrMaxRoutesExceeded NOTIFICATION-TYPE
        OBJECTS { vrRowStatus, vrMaxRoutes, vrStatRouteEntries }
        STATUS current
        DESCRIPTION
            "This notification is generated when the specified VR has
             exceeded the maximum number of routes specified.
            "
        ::= { vrNotifications 3 }

-- ****************************************************************
-- Module Compliance/Conformance Statements
-- ****************************************************************

    vrConformance OBJECT IDENTIFIER ::= { virtualRouterMIB 2 }

    vrCompliances OBJECT IDENTIFIER ::= { vrConformance 1 }

    vrMIBCompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
            "The compliance statement for entities that implement the
            VIRTUAL-ROUTER-MIB.  Implementation of this MIB module
            is strongly recommended for any platform targeted for a
            carrier-class environment.
            When this MIB module is implemented with support for
            read-create, then such an implementation can claim full
            compliance.
            Such devices can then be both monitored and configured
            with this MIB."
        MODULE -- this module
            MANDATORY-GROUPS { vrConfigGroup, vrStatGroup,
                               vrIfGroup, vrNotificationGroup }
        ::= { vrCompliances 1 }

    vrGroups OBJECT IDENTIFIER ::= { vrConformance 2 }

    vrConfigGroup OBJECT-GROUP
        OBJECTS { vrRowStatus, vrName,
                  vrContextName, vrTrapEnable,
                  vrMaxRoutes, vrAdminStatus,
```

```
            vrVpnId, vrRpTrigger,
            vrMaxRoutesTrapEnable,
            vrConfigNextAvailableVrId }
        STATUS current
        DESCRIPTION
            "A collection of attributes that support provisioning
            of a virtual router."
        ::= { vrGroups 1 }
```

```
    vrStatGroup OBJECT-GROUP
        OBJECTS { vrConfiguredVRs, vrActiveVRs,
                  vrStatRouteEntries, vrStatFIBEntries, vrStatUpTime,
                  vrOperStatus, vrRpStatus, vrRouterAddress,
                  vrRouterAddressType }
        STATUS current
        DESCRIPTION
            "A collection of attributes that contain stats about the
            virtual router."
        ::= { vrGroups 2 }



    vrIfGroup OBJECT-GROUP
        OBJECTS { vrIfConfigRowStatus  }
        STATUS current
        DESCRIPTION
            "A collection of attributes that support provisioning of
            a virtual router interfaces."
        ::= { vrGroups 3 }

    vrNotificationGroup NOTIFICATION-GROUP
        NOTIFICATIONS { vrUp, vrDown, vrMaxRoutesExceeded }
        STATUS current
        DESCRIPTION
            "A collection of traps that are supported by the VR."
        ::= { vrGroups 4 }

END
```

7.0 Acknowledgments

Funding for the RFC Editor function is currently provided by the Internet Society.

Special thanks to Joan Cucchiara for providing valuable comments on this MIB.

8.0 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:

The Administrative VR provides visibility into and control over multiple VPNs. As such, security considerations for implementations of the Administrative VR and associated control plane(s) are critical to the security of the VPNs supported on each PE device.

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:

Use of any vrContextName MUST be allowed in the Administrative VR. Additional authentication and security mechanisms SHOULD be used for SNMP access in the Administrative VR.

VRs other than the Administrative VR MUST NOT have access to other VR's Instantiated MIB modules, and MAY have access to their own instantiated MIB modules.

In VRs other than the Administrative VR, access to that VR's instantiated MIB modules MAY be permitted via that VR's vrContextName. Use of any vrContextName other than that assigned to the accessed VR MUST result in an error, and implementations SHOULD provide a logging mechanism for such events.

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.

9.0 References

9.1 Normative References

[RFC2571] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.

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

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

[RFC2579] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

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

[VPNTCMIB] B. Schliesser, and T. Nadeau, "Definition of Textual Conventions for Provider Provisioned Virtual Private Network (PPVPN) Management.", Internet Draft <draft-ietf-l3vpn-tc-mib-03.txt>, May 2004.

9.2 Informative References

[RFC2685] Fox B., et al, "Virtual Private Networks Identifier", RFC 2685, September 1999.

[PPVPN-FW] R. Callon, et al., "A Framework for Layer 3 Provider Provisioned Virtual Private Networks", draft-ietf-l3vpn-framework-00.txt, March 2003.

[PPVPN-VR] P. Knight, et al., "Network based IP VPN Architecture using Virtual Routers", draft-ietf-l3vpn-vpn-vr-02.txt, April 2004.

[PPVPN-VR-AS] A. Nagarajan, et al., "Applicability Statement for Virtual Router-based Layer 3 PPVPN approaches", draft-ietf-l3vpn-as-vr-01.txt, March 2004.

10.0 Authors' Addresses

Elwin Stelzer Eliazer
Corona Networks, Inc.
630 Alder Drive
Milpitas, CA 95035
USA
Phone: +1-408-519-3832
Email: elwinietf@yahoo.com

Samuel Hancock
ACM Systems
3034 Gold Canal Drive
Rancho Cordova, CA 94123
USA
Phone: +1-916-463-7949
Email: hancoc_s@yahoo.com

Benson Schliesser
SAVVIS Communications
1 Savvis Parkway
Town and Country, MO 63017
USA
Phone: +1-314-628-7036
Email: bensons@savvis.net

Joseph Laria
Level Stream Research
Wilmington, MA 01887
USA
Phone: +1-978-223-9908
Email: jlaria@levelstream.com

11.0 Intellectual Property Considerations

of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

12.0 Full Copyright Statement

13.0 IANA Considerations for L3VPN-VR-MIB Module

The IANA is requested to assign { mib-2 XXXX } to the L3VPN-VR-MIB module specified in this document.