VMS

Northbound Interface

Revision:	Date:
R 1.3	June 27, 2014

Document ID:

NBI-083112-P1

Table of Contents

Introduction

The VMS SNMP module Northbound Interface (NBI) available in version 3.10 or greater provides two services to external network management systems. First, it allows an external NMS to query the VMS for certain operational status. Second, it can operate as a proxy to Comtech EF Data networking hardware, and fulfill certain requests with information collected via CEFD's proprietary management protocol thus minimizing satellite bandwidth utilization for common queries.

Typically all SNMP GET requests to a remote are handled directly from the modems built-in SNMP v1/v2c agent through satellite communication links. To support statistical reporting and control, these messages travel over each establish link sharing a small portion of the end users bandwidth. Even though the total amount of link capacity per remote is typically low, the aggregate bandwidth on both the outbound and all of the return links could potentially occupy much larger percentages; infringing on Service Level Agreement contracts. Knowing the high cost of satellite space segments, which represent a large portion of the end customers SLA, SNMP's requirement for bi-directional and Basic Encoding Rule (BER) formatted message exchange has at least one disadvantage, inefficient bandwidth resource usage, which as previously stated is multiplied by the number of remotes.

In order to reduce the management overhead for typical device monitor queries, the new NBI feature of VMS adds many advantages through caching techniques. Each of the devices (modems and gateway routers) in the network by design already report using an unsolicited message that contains majority of the key parameters required by monitor systems. These Status Update Messages "SUM" are sent on 60 second intervals to the active VMS. These messages are encoded using a highly efficient "over the air" format that can reduce the data per variable to as little as 5 bits as compared to a typical SNMP variable binding consuming hundreds of bits when considering the bi-directional nature of SNMP. Disregarding per packet overhead, a typical alarm query will require ~300 bits by SNMP, where as the worst case for a SUM would be 9 bits, and for no alarm states its 5 bits. That's a 30x to 60x saving overall, and that is only one example.

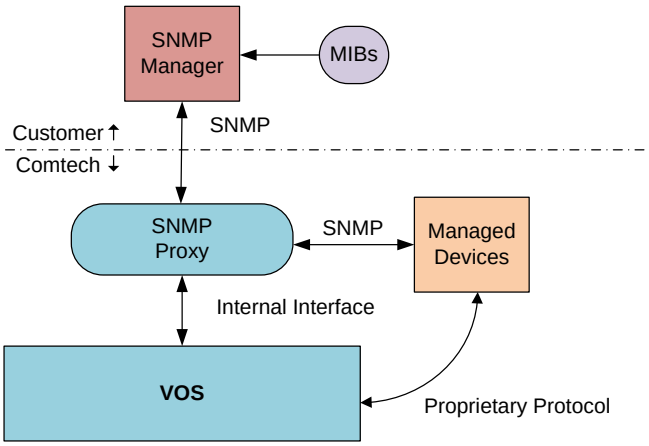
Per packet overhead is also significant, a round trip SNMP message will use around 128 bytes just for headers within the UDP payload whereas the SUM message has a per packet overhead around 30 bytes. The content of these SUM messages are parsed and processed to support the UI, system events and key internal processes. Some of these collected values are stored in volatile memory while others are stored to non-volatile memory. In either case an active VMS can fulfill all standard queries directly while reducing overhead significantly.

NBI Feature Breakdown

The nomenclature behind Northbound refers to an interface that conceptualizes lower level details, e.g. modems and VMS. It interfaces to higher level layers (managers) which are normally drawn at the top of an architectural network overview. With that said, the feature is an exposed single interface that accepts SNMP messages, GET, GET NEXT… parses packet data and redistributes to internals and network agents. This interface acts as a Proxy to incoming SNMP requests forwarding to the appropriate handlers providing a single point of entry for one or more mangers.

The Proxy cache supports 800 series, 570, 564 and 5650A MIB OID's as read only and processes a subset of variables using a proprietary CEFD caching mechanism. All other requests that fall outside of the scope of the local caching and set commands are directly forwarded to end agent for standard processing.

The following diagram depicts a simplistic overview of the module flow. Note that the Proxy function is integral to the core software libraries of the VMS.



SNMP Flow Diagram

Operational Status Queries

The VMS exposes certain operational status via SNMP to external agents. The status is exposed as a set of virtual SNMP entities identified via a unique community string. Branches of the defined MIBs are only valid on certain entities.

The following table describes the exposed entities, and what branch of the MIB they support.

Entity	Description	Valid MIB branches
system	Represents the VMS as a whole.	vms.system.health.systemStatus
site	A site from the network manager	vms.system.health.objectStatus vms.switching.site
unit	Represents a modem as a whole	vms.system.health.objectStatus vms.switching.unit
modulator	Represents a single modulator subcomponent of a modem.	vms.system.health.objectStatus vms.switching.managedDevice
demodulator	Represents a single demodulator subcomponent of a modem.	vms.system.health.objectStatus vms.switching.managedDevice

Entity Identifiers

The unique identifier for the system entity is the community string “server”. The other identifiers are for the purpose of the SNMP agent, without format and can only be obtained via queries to other entities. The exception is the unit which can also be referenced via the same community string used to perform a proxied request to the associated physical hardware.
As an example, the modulator identifier for the first modulator subcomponent of a modem with the IP address 172.18.100.1 can be obtained by querying the VMS for the “modulatorId” variable of the switching MIB with an instance of 1 and a community string of “public@172.18.100.1”. The resulting octet string will be the entity identifier “moniker” to be used as the community when querying related MIB variables.

Hub Demodulator Eb/No

One of the main preferences is to correlate the Eb/No for the hub demodulator of a switched remote modulator, which can be obtained by querying the modem via the proxy for the “unitInbandReturnPathEbN0” variable in the switching MIB. This variable is designed to look like part of the remote modem, when in reality the VMS intercepts the request and fills in the Eb/No of the currently allocated hub demodulator. This allows for a very simple way of monitoring the quality of a dynamic link without the complexity of multiple queries involving different community strings.

For example, if your remote data unit was a CDM-840 with an IP address of 172.18.100.1, you would use the VMS as a proxy by directing your SNMP requests to the VMS using a community string like "public@172.18.100.1". Along with querying the remote modem for standard values like 'cdm840TxFrequency' or 'cdm840RxLock', you could include a request for 'unitInbandReturnPathEbN0' to get the currently receiving hub demodulators EbN0 as well. This variable operates much like one of the cached modem parameters. To determine the value for this parameter, the VMS searches for an allocation associated with the specified unit's first modulator. If the modulator has an associated allocation, it queries the first allocated demodulator (which is always at the hub) for its last known Eb/No value. If the modulator does not have an associated allocation, the value returned is null.

Tables Support

The current support for tables is limited. It is roughly equivalent to SNMP version 1. There is support for Get, Get Next and Walk, but no support for GETBULK requests. The way to enumerate a table is to send Get Next or Table View.

Proxy Caching Support

When operating as a proxy on behalf of Comtech network equipment, the VMS will fulfill the requests for a subset of the MIB using data collected via proprietary protocols. When a request is made for one of these MIB variables, the VMS will report the last known value without forwarding the request to the end node. This data is collected at a frequency of at least once per minute.

To use the proxy/cache support, send SNMP queries for the modem to the VMS, and use a specially formatted community string to identify what device is being queried. The community string format is community@ip-address. For example, to target a device with the IP address of 172.16.128.1, using a read community of public, the community string sent to VMS would be "public@172.17.128.1".

Operational Procedures

There are two sets of VMS MIB files that comprise the interface structure for internal caching parameters, VMS and Switching.

The following is the list of items objects available through this interface and the following will provide steps to exercise for a better understanding of functionality. Each parameter value queried will return results that can be compared to already available user interfaces as a sanity check and verification.

- Table of Remotes
- Alarm Status per remote
- Link Statistics:
  - Eb/No
  - Frequency
  - Data Rate
  - FEC
  - Modulation
  - Offset (frequency)

Setup Procedure

1. Backup current configurations
2. Update all lab components, VMS, CDM-800, CDD-880 and CDM-840 to latest builds.
3. Verify standard operations.
4. Install iReasoning or use supplied MIB browser.
5. Load all associated MIB's into browser's library.
6. Exercise each of the contractual parameters using SNMP command operations.
7. Set proper community String:
   a. Enter "server" in Read Community for System queries.

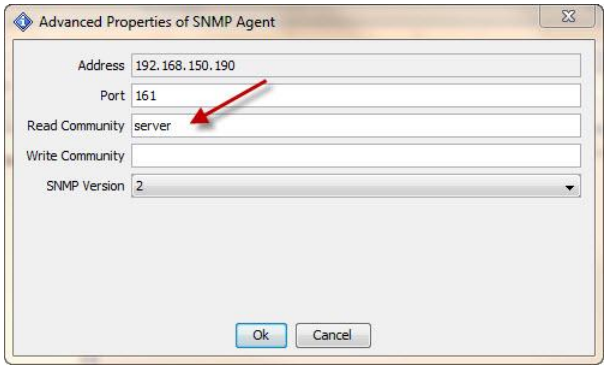
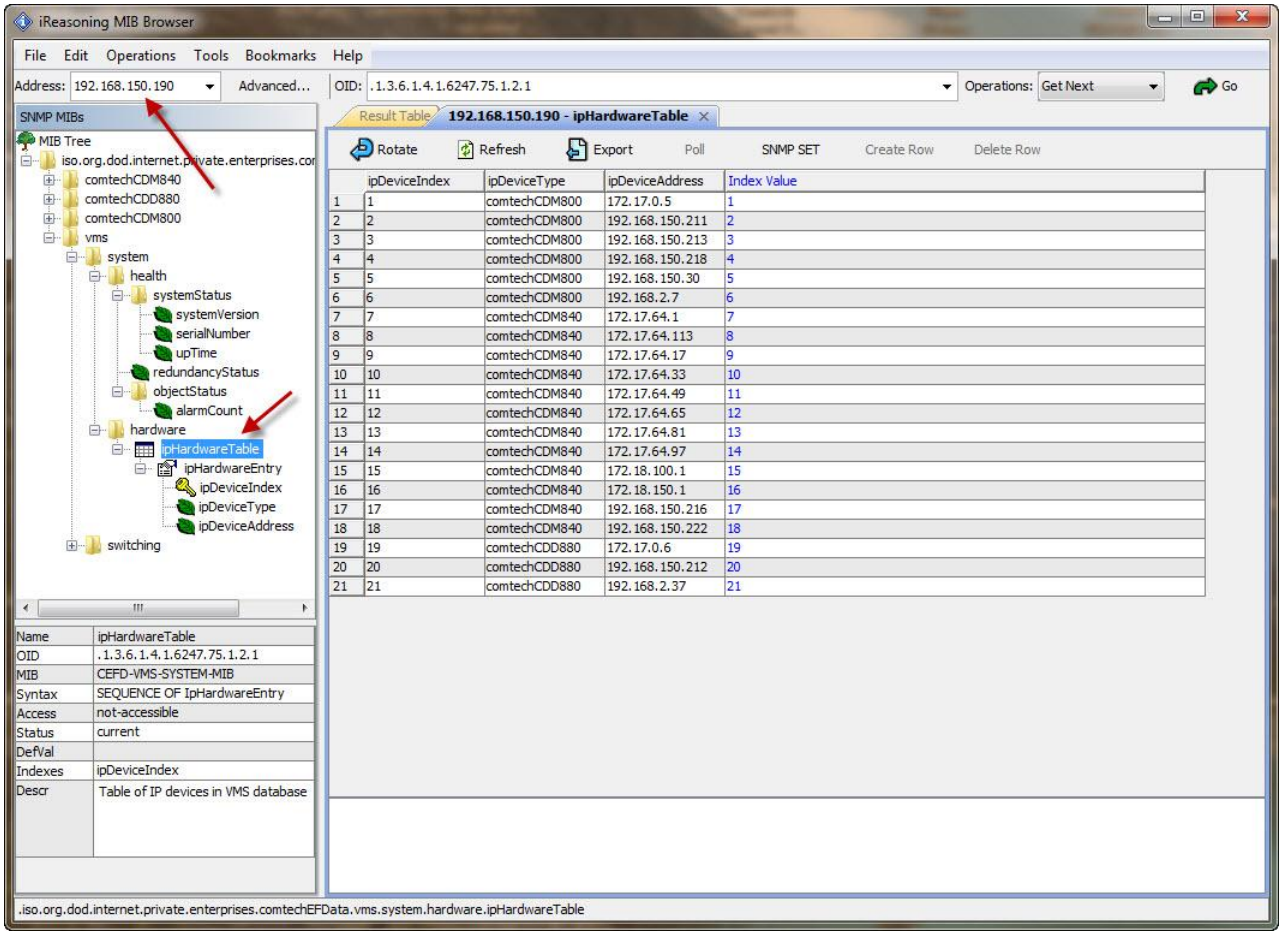


Table of Remotes

The VMS provides a table of configured devices through the ipHardwareTable MIB branch. This allows a northbound management entity to list the hardware configured in the VMS database. The hardware is identified by its IP address and type. To retrieve this table the VMS must be targeted with a community string of "server".



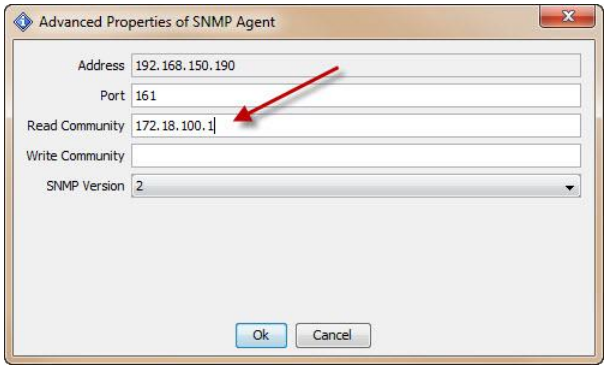
The example above used a Tree View or table get to poll all instances within the table. Get Next will step/walk the table. This option is not like network discovery where a manager will poll through a range of addresses for any MIBII devices connected to a network populating map views. These are local database entries that were either previously discovered or manually declared to the VMS only. Other devices outside the VMS database will not be listed.

Alarm Status per Remote

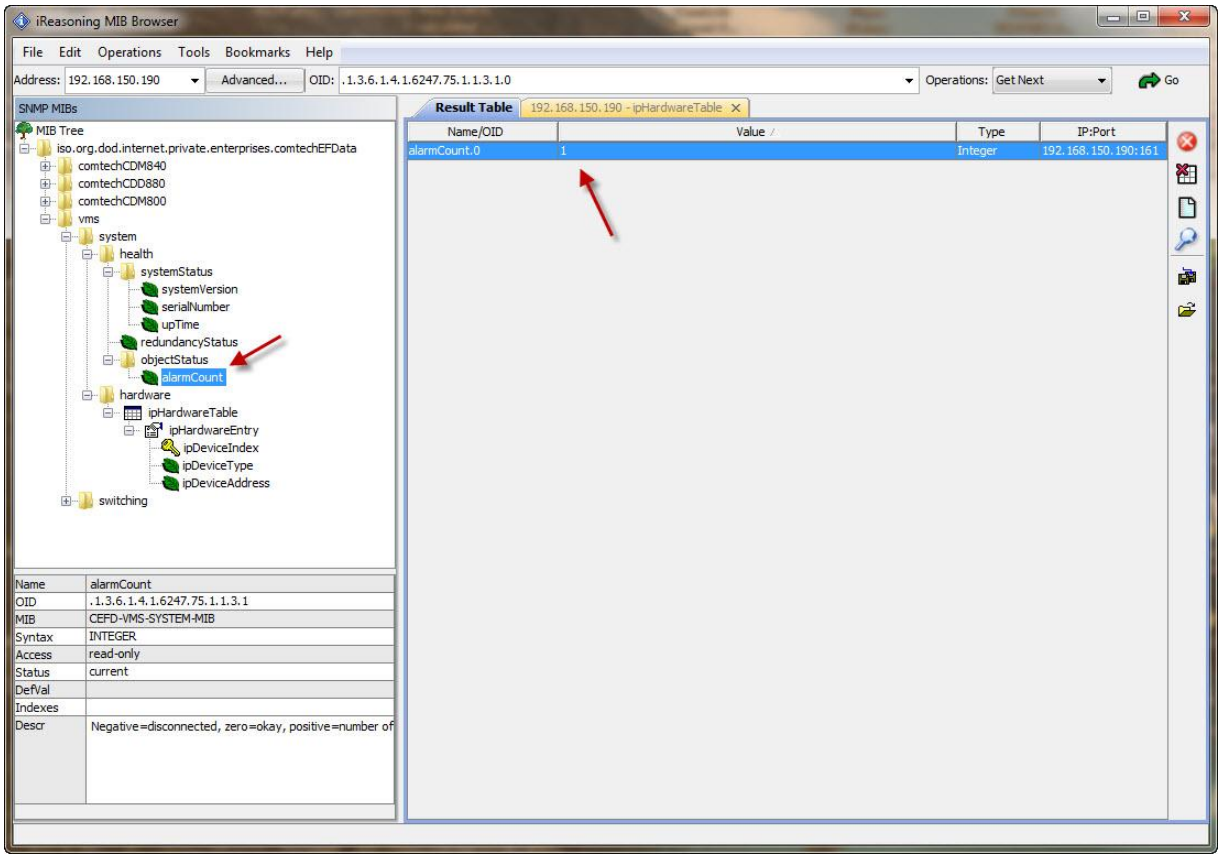
Each of the structured devices forward SUM messages on interval containing not only parameter settings and status values, but also alarm information. What is presented through this call is an integer value representing a count of alarms set within the device, unit, modulator, demodulator… To further evaluate the alarm information or type the devices MIB would be used to query alarm lists.

To queries individual unit alarm status, set the community read string to the IP address of the device.

1. Enter “public@IP Address” in Read Community for Unit queries.



2. Select the “alarmCount” OID for result.



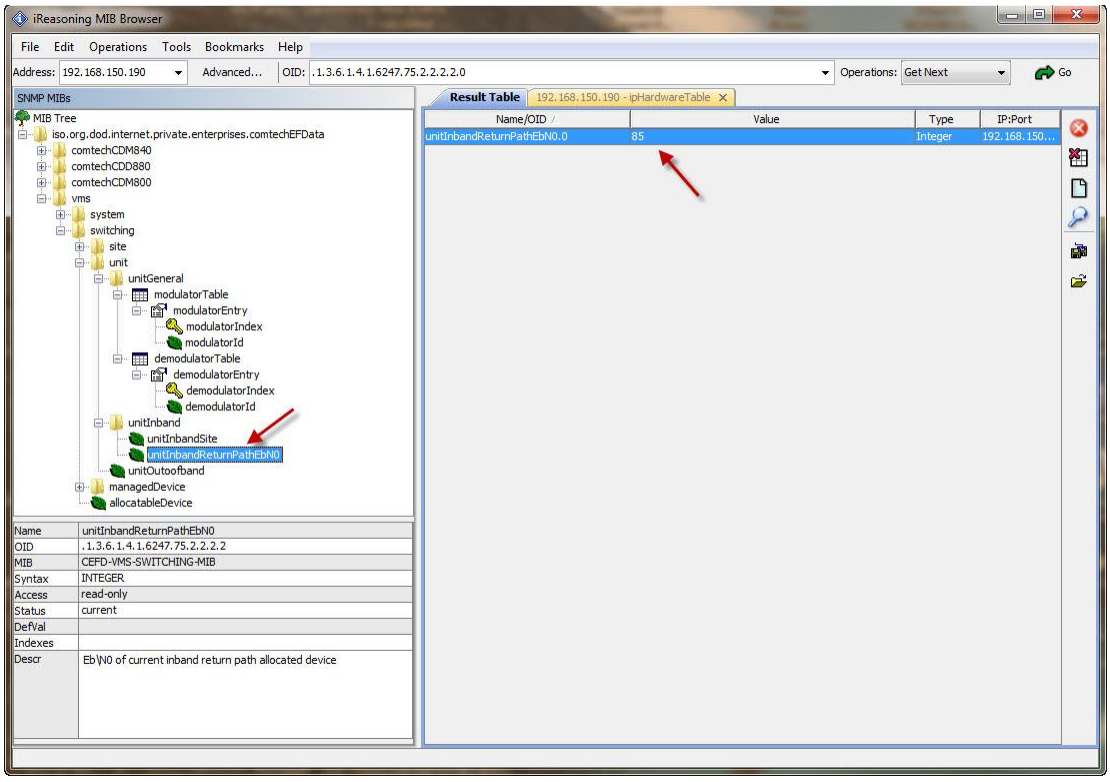
Link Statistics

Hub Demodulator EbN0

One of the main preferences is to correlate the Eb/No for the hub demodulator of a switched remote modulator. This can be obtained by querying the modem via the proxy for the "unitInbandReturnPathEbN0" variable in the switching MIB. This variable is designed to look like part of the remote modem, when in reality the VMS intercepts the request and fills in the Eb/No of the currently allocated hub demodulator.
This allows for a very simple way of monitoring the quality of a dynamic link without the complexity of multiple queries involving different community strings.

For example, if your remote data unit was a CDM-840 with an IP address of 172.18.100.1, you would use VMS as a proxy by directing your SNMP requests to the VMS using a community string like "public@172.18.100.1". Along with querying the remote modem for standard values like 'cdm840TxFrequency' or 'cdm840RxLock', you could include a request for 'unitInbandReturnPathEbN0' to get the currently receiving hub demodulators EbN0 as well. This variable operates much like one of the cached modem parameters.

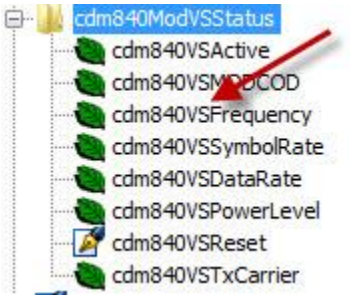
To determine the value for this parameter, the VMS searches for an allocation associated with the specified unit's modulator. If the modulator has an associated allocation, it queries the first allocated demodulator (which is always at the hub) for its last known Eb/No value. If the module does not have an associated allocation, the value returned is null.



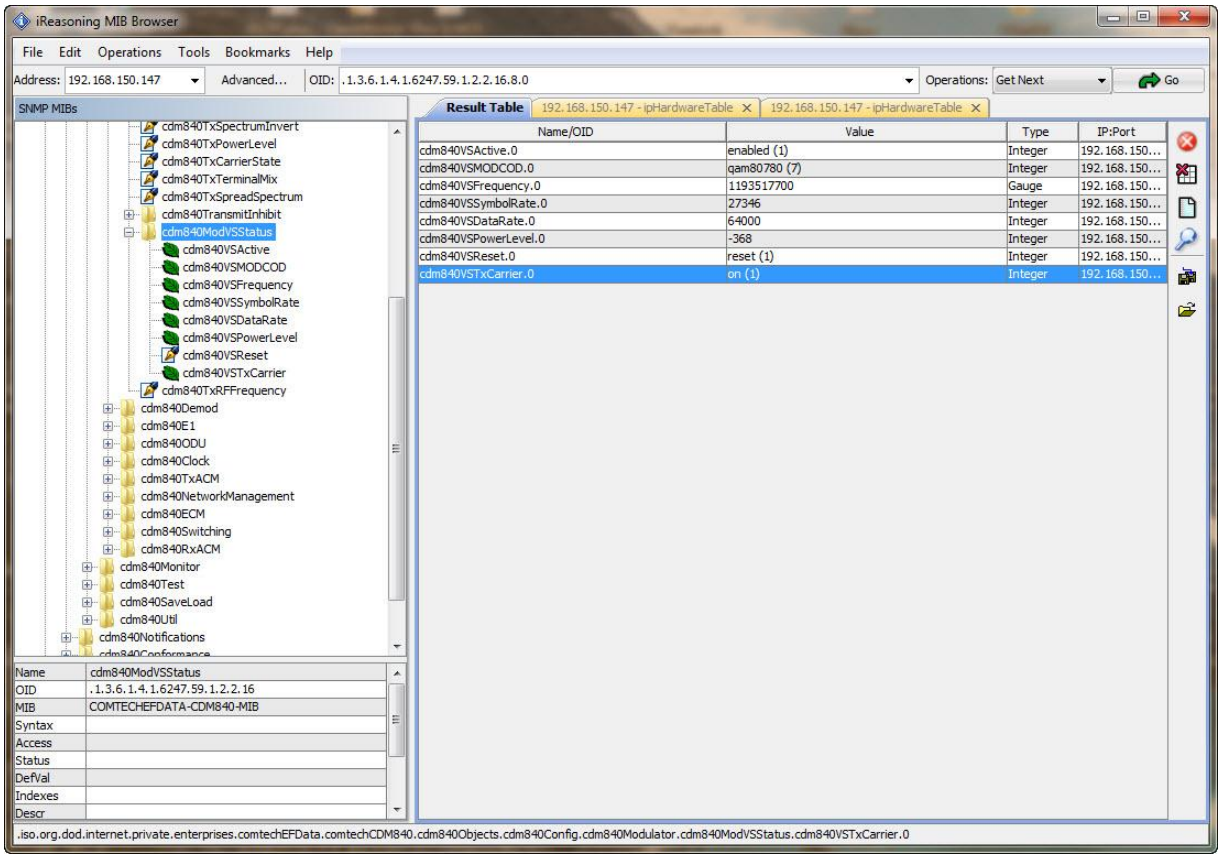
The example above shows a single instance of an In-banded remote switched into dSCPC with correlated hub demodulator's signal link quality.

This next set of OID queries will further demonstrate caching through device MIB interception, where we step through the objects that represent the dynamic switch state. Note for VMS managed (switched) devices "CDM-840", "CDD-880" there is a separate set of objects that provide the current dynamic switched state, not be confused with static state objects. All dynamic OID's are labeled with "VS" with signifies Vipersat Switched.

Example VS OID's



This example below shows a step through of 840 dynamic parameters.



Offset (Frequency)

The demodulator acquisition frequency offset is in two parts, one from the demodulator at the remote receiver and the other from the coordinated or associated (switched) demodulator at the hub. The outbound receiver offset is a pass- through not cached, whereby the proxy forwarder sends the request to the remote agent.

The second hub associated (switched) demodulator is known in the VMS switching engine, thus cached value. To retrieve this information requires some finesse as the association is not as straightforward as Eb/No. We first need to learn the allocated device through a series of steps and once the association is known the internal value can be polled.

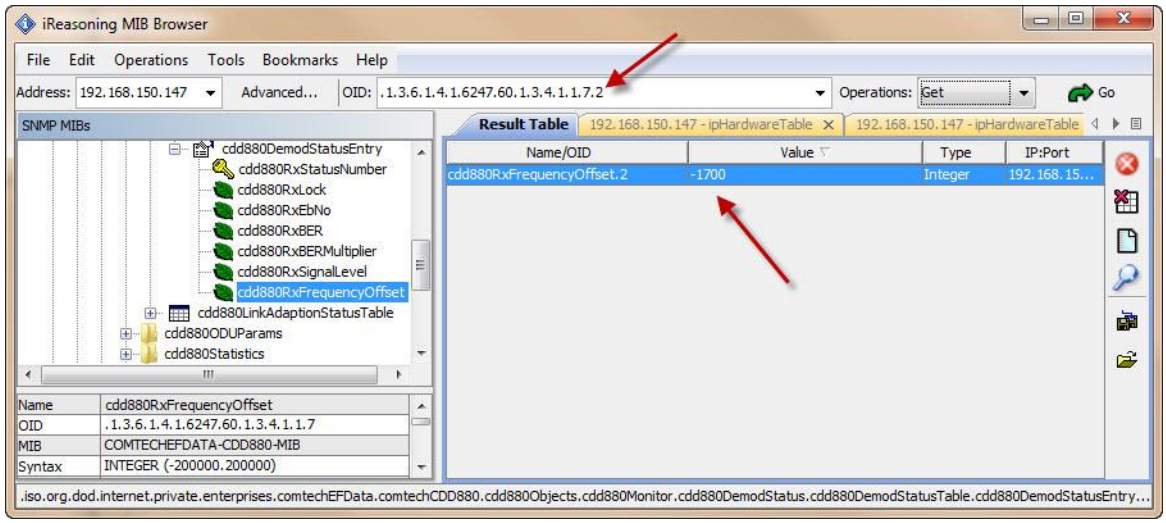
Steps to indentify

1. Set Read Community string "public@IP Address" to modulator device in question.
2. Query "modulatorId" OID to learn the entity identifier "moniker" ves:cdm840-172.18.100.1,1,0
3. Copy the moniker or octet string into the Read Community
4. Next query the "deviceAllocationAllocatedDeviceId" OID to identify associated demodulator, ves:cdd880-192.168.150.212,2,0 – Note the device # in the octet string.

   This is the instance that is part of the query:

   ves:cdd880-192.168.150.212,**2**,0

5. Now write the learned demodulator IP Address (example, 192.168.150.212) into the Read Community
6. Query the modems MIB "cdd880RxFrequencyOffset" with an instance from learned (example #2) octet string.



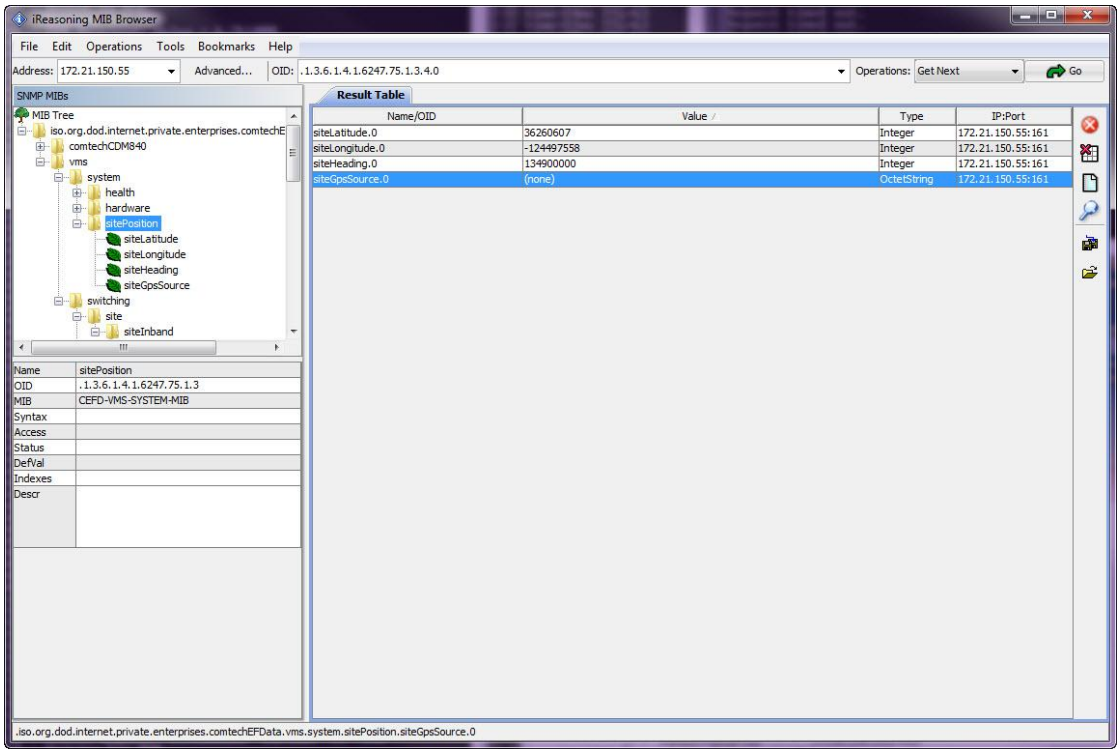
Example above shows the results of the learned association.

Site Position

The Site Position is associated with the unit inband site object, which contains network and site index identification within the database. The site index or number is irrelevant as it is a runtime generated number.

Steps to indentify

1. Set Read Community string “public@IP Address” to modulator device in question.
2. Query “unitInBandSite” OID to learn the entity identifier “moniker” → network:site,4



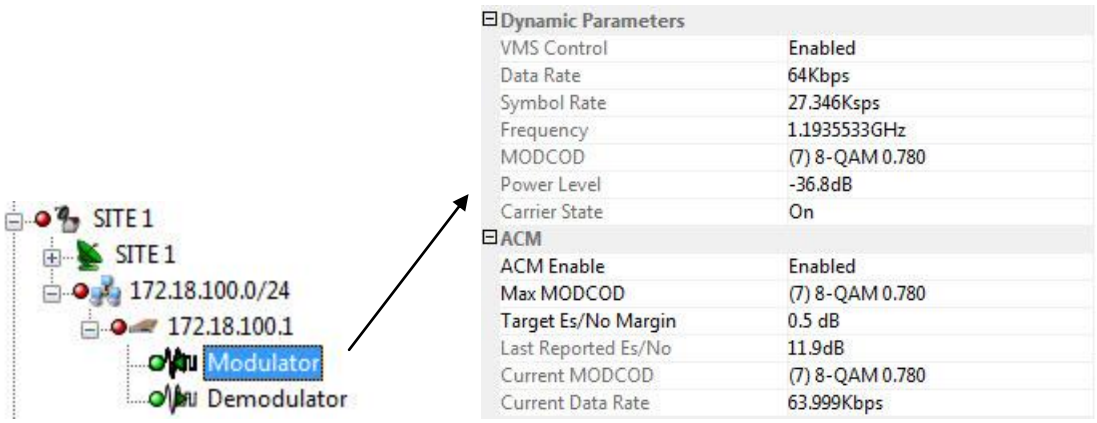
3. Copy the moniker string into the Read Community
4. Next query the “sitePosition” OID list to identify location information.

Caching Test Verification

We will use one or all listed variables in the "Vipersat Management System SNMP Module" to exercise the caching capabilities, at customer's discretion.

Steps:

1. Verify normal communications to remote device using VMS device view.



2. Select OID from supported list.
3. Disable remote unit, power off.
4. Query selected OID.

Note during the time of communications failure the caching will be valid up to ~3 minutes, once the VMS times-outs the connections state "Disconnected" the query return will error, timeout. This a simple way to determine if caching is working correctly.

Appendix A: Cached MIB Variables

The specific MIB variables that are cached vary per supported modem, and potentially per revision of a specific modem. The following lists summarize what MIB variables are cached for supported modems at their latest release.

Cached 8xx Series MIB Values

The following are cached MIB values supported through VMS unsolicited system updates.

CDM-800 Version 1.6.x

cdm800UnitAlarms
cdm800TrafficEthernetAlarms
cdm800TxAlarms
cdm800TxFrequency
cdm800TxDataRate
cdm800TxMODCOD
cdm800TxFECType
cdm800TxPowerLevel
cdm800TxCarrierState
cdm8005vPowerAlarm
cdm80012vPowerAlarm
cdm800TxSynthPLLLockAlarm
cdm800FPGALockAlarm
cdm800TXFPGALoadAlarm
cdm800PrimaryFPGALoadAlarm
cdm800ExtFPGALoadAlarm
cdm800NoExtRefAlarm
cdm800ExtRefLockAlarm
cdm800NoLinkGEAlarm
cdm800NoLinkEthAlarm
cdm800Tx10PLLLockAlarm
cdm800TxLMKPLLLockAlarm
cdm800FIFOSlipAlarm
cdm800S2DataLengthMismatchAlarm
cdm800ModCardAlarm
cdm800TempExceededAlarm
cdm800E1ExceedsMinus50PPMAlarm
cdm800E1ExceedsPlus50PPMAlarm
cdm800E1RefInactiveAlarm
cdm800HardResetAlarm
cdm800Tx130PLLLockAlarm
cdm800BUCCurrentAlarm
cdm800BUCVoltageAlarm
cdm800PTPConfigErrorAlarm
cdm800PTPErrorThreshAlarm
cdm800PTPSyncThreshAlarm
cdm800PTPFollowupThreshAlarm
cdm800PTPDelayResThreshAlarm
cdm800PTPMasterNotAcceptableAlarm
cdm800ctogNoLinkLANAlarm
cdm800ctogNoLinkExpansionAlarm
cdm800ctogFanSpeedAlarm
cdm800ctogCPUTempAlarm
cdm800ctogDriveFailureAlarm
cdm800ctogPowerSupplyAlarm
cdm800ctogHeartbeatTimeoutAlarm

CDM-840 Version 1.6.x

cdm840UnitAlarms
cdm840TrafficEthernetAlarms
cdm840TxAlarms
cdm840TxFrequency
cdm840TxSymbolRate
cdm840TxDataRate
cdm840TxMODCOD
cdm840TxFECType
cdm840TxPowerLevel
cdm840TxCarrierState
cdm840TxACMLastMsgEsNo
cdm840TxACMCurrentModcod
cdm840TxACMCurrentDataRate
cdm840VSActive
cdm840VSMODCOD
cdm840VSFrequency
cdm840VSSymbolRate
cdm840VSDataRate
cdm840VSPowerLevel
cdm840VSTxCarrier
cdm840RxAlarms
cdm840RxFrequency
cdm840RxSymbolRate
cdm840RxDataRate
cdm840RxMODCOD
cdm840RxLock
cdm840RxEsNo

CDM-850 Version 1.6.x

cdm840UnitAlarms
cdm840TrafficEthernetAlarms
cdm840TxAlarms
cdm840TxFrequency
cdm840TxSymbolRate
cdm840TxDataRate
cdm840TxMODCOD
cdm840TxFECType
cdm840TxPowerLevel
cdm840TxCarrierState
cdm840TxACMLastMsgEsNo
cdm840TxACMCurrentModcod
cdm840TxACMCurrentDataRate
cdm840VSActive
cdm840VSMODCOD
cdm840VSFrequency
cdm840VSSymbolRate
cdm840VSDataRate
cdm840VSPowerLevel
cdm840VSTxCarrier
cdm840RxAlarms
cdm840RxFrequency
cdm840RxSymbolRate
cdm840RxDataRate
cdm840RxMODCOD
cdm840RxLock
cdm840RxEsNo

CDM-880 Version 1.6.x

cdd880UnitAlarms
cdd880TrafficEthernetAlarms
cdd880BaseFrequency
cdd880RxAlarms
cdd880RxLock
cdd880RxEbNo
cdd880RxFrequency
cdd880RxSymbolRate
cdd880RxDataRate
cdd880RxMODCOD
cdd880RxEnable
cdd880LinkAdaptionStatusCurrentDataRate
cdd880LinkAdaptionStatusCurrentEsNo
cdd880LinkAdaptionStatusCurrentModCod
cdd880VSActive
cdd880VSMODCOD
cdd880VSFrequency
cdd880VSSymbolRate
cdd880VSDataRate
cdd880VSEnable

Cached 570/564 MIB Values

The following are cached MIB values supported through VMS unsolicited system updates.

CDM-570

cdm570TxFrequency
cdm570TxDataRate
cdm570TxFECCodeRate
cdm570TxModType
cdm570TxFECType
cdm570TxPowerLevel
cdm570TxCarrierState
cdm570TxAlarms
cdm570RxFrequency
cdm570RxDataRate
cdm570RxFECCodeRate
cdm570RxDemodType
cdm570RxFECType
cdm570RxEbNo
cdm570RxAlarms
cdm570UnitAlarms

CDD-564

cdd564RxFrequency
cdd564RxDataRate
cdd564RxCodeRate
cdd564RxDemodulation
cdd564RxFEC
cdd564RxEbNo
cdd564RxAlarm
cdd564UnitAlarm

Cached 5650A MIB Values

The following are cached MIB values supported through VMS unsolicited system updates.

SLM-5650A

unitAlarmPower5V
unitAlarmPower33V
unitAlarmPower25V
unitAlarmPower15V
unitAlarmPower12V
unitAlarmPowerMinus12V
unitAlarmPower18V
unitAlarmCoolingFan
unitAlarmExtRefActivity
unitAlarm192MhzClockNotLocked
unitAlarm10MhzClockNotLocked
unitAlarmMnCFPGANotLoaded
unitAlarmModFPGANotLoaded
unitAlarmDemodFPGANotLoaded
unitAlarmDecoderFPGANotLoaded
unitAlarmTxIntfFPGANotLoaded
unitAlarmRxIntfFPGANotLoaded
unitAlarmFEC1FPGANotLoaded
unitAlarmFEC2FPGANotLoaded
unitAlarmOPCFPGANotLoaded
unitAlarmFPGADCMNotLocked

txFrequency
txFECType
txModType
txFECCodeRate
txDataRate
txPowerLevel
txCarrierState
txAlarmSymbolClockNotLocked
txAlarmRFSynNotLocked
txAlarmNoIQActivity
txAlarmNyquistFilterClipping
txAlarmIntfClockPLLNotLocked
txAlarmIntfTeresClockNoActivity
txAlarmIntfSCTPLLNotLocked
txAlarmIntfNoDataActivity

rxFrequency
rxFECType
rxDemodType
rxFECCodeRate
rxDataRate
rxAlarmIfNotlocked
rxAlarmDataDecodeNotLocked
rxAlarmRFSynNotLocked
rxAlarmNoIQActivity
rxAlarmIntfDemuxNotLocked
rxAlarmIntfBufferFault
rxAlarmIntfBufferAboutToSlip
rxAlarmIntfBufferOverflow
rxAlarmIntfBufferUnderflow
rxAlarmIntfBufferClockPLLNotLocked

rxAlarmIntfBufferClkRefNoActivity
rxAlarmIntfDataAIS
rxAlarmIntfEbNoThreshold
rxAlarmCompositePwrExceeds40dBc
rxAlarmCompositePwrExceeds20dBc
rxEbNo

SLM-5650D

unitAlarmPower5V
unitAlarmPower33V
unitAlarmPower25V
unitAlarmPower15V
unitAlarmPower12V
unitAlarmPowerMinus12V
unitAlarmPower18V
unitAlarmCoolingFan
unitAlarmExtRefActivity
unitAlarm192MhzClockNotLocked
unitAlarm10MhzClockNotLocked
unitAlarmMnCFPGANotLoaded
unitAlarmModFPGANotLoaded
unitAlarmDemodFPGANotLoaded
unitAlarmDecoderFPGANotLoaded
unitAlarmTxIntfFPGANotLoaded
unitAlarmRxIntfFPGANotLoaded
unitAlarmFEC1FPGANotLoaded
unitAlarmFEC2FPGANotLoaded
unitAlarmOPCFPGANotLoaded
unitAlarmFPGADCMNotLocked

txFrequency
txFECType
txModType
txFECCodeRate
txDataRate
txPowerLevel
txCarrierState
txAlarmSymbolClockNotLocked
txAlarmRFSynNotLocked
txAlarmNoIQActivity
txAlarmNyquistFilterClipping
txAlarmIntfClockPLLNotLocked
txAlarmIntfTeresClockNoActivity
txAlarmIntfSCTPLLNotLocked
txAlarmIntfNoDataActivity

rxFrequency
rxFECType
rxDemodType
rxFECCodeRate
rxDataRate
rxAlarmIfNotlocked
rxAlarmDataDecodeNotLocked
rxAlarmRFSynNotLocked
rxAlarmNoIQActivity
rxAlarmIntfDemuxNotLocked

rxAlarmIntfBufferFault
rxAlarmIntfBufferAboutToSlip
rxAlarmIntfBufferOverflow
rxAlarmIntfBufferUnderflow
rxAlarmIntfBufferClockPLLNotLocked
rxAlarmIntfBufferClkRefNoActivity
rxAlarmIntfDataAIS
rxAlarmIntfEbNoThreshold
rxAlarmCompositePwrExceeds40dBc
rxAlarmCompositePwrExceeds20dBc
rxEbNo