DATUM SYSTEMS

SnIP SNMP Guide

SnIP875 Technical Note

Revision History

Rev 0.1	3-25-2009	Initial Release.
Rev 0.2	3-8-2010	Addition of m500-agent for M500 modems.
Rev 0.3	5-20-2010	Add SNMP Trap information for M500 modems.
Rev 0.4	8-10-2010	Add BUC and LNB elements for M500 modems.
Rev 0.5	8-28-2010	Add Unit Redundancy elements for M500 modems.
Rev 0.6	3-16-2011	New Ubuntu 10.10 procedures.
Rev 0.7	4-10-2011	New m500-agent to add virtual LO and RF frequencies.
Rev 0.8	11-4-2011	Add alerts to create trap capabilities.

1.0 SnIP SNMP Overview

This Tech Note described the resources and procedures necessary to use the SNMP facilities provided by the Datum System's SnIP Ethernet Interface. It does not describe SNMP itself or methods and procedures for network management other than basic concepts.

The SnIP is a fairly complete Linux computer and contains a standard set of over 100 individual programs, scripts and other elements required for proper functioning. It also has a small separate Operating System “kernel” which is the base of Linux and held in a separate location from other files. The SnIP implementation of SNMP is based on the standard and popular “Net-SNMP” system, originated as UCD-SNMP. It is compiled and installed as part of the standard SnIP resources. SNMP standards for Simple Network Management Protocol.

The current SNMP version is 5.2.1.2 in SnIP filesystem versions 0.5.xx, and 5.4.2.1 in SnIP filesystem versions 0.6.xx. It is capable of using the v1, v2c and v3 protocol versions. The Datum Systems “PEN” for SNMP is 31978 which is used for the branch of the standard SNMP tree dealing with Datum System’s Private Enterprise Network objects.

1.0.1 SNMP – Some Definitions

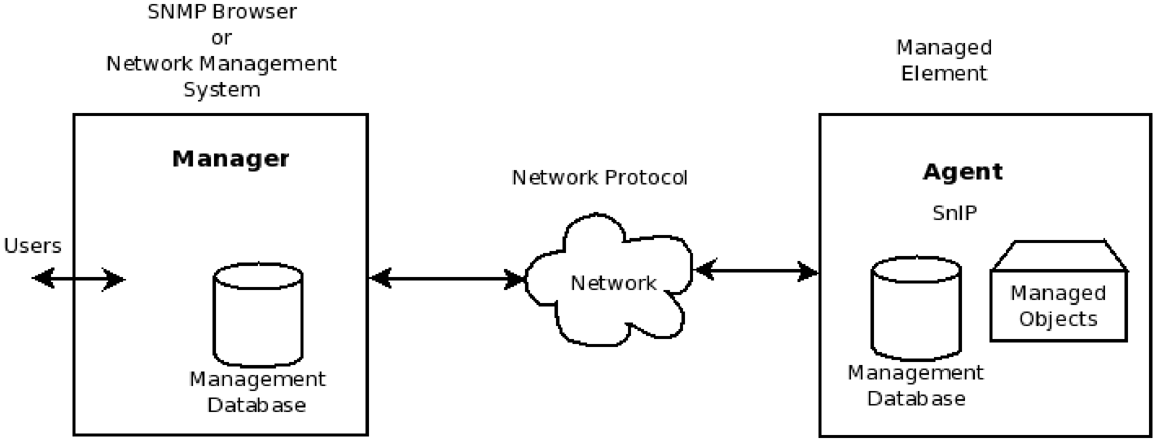
For the purposes of the remainder of this document the conventions for naming the different objects that are part of SNMP plus elements of the Linux operating system are:

- “***Kernel and Kernel Space***”. These refer to the Linux kernel itself and the modules that are compiled with the kernel to be loaded as part of the kernel. The current kernel is version 2.6.15.
- “***User and User Space***” Multiple types of programs can exist in the user space. The user space is where all the files that a user can directly interact with reside. This includes the control programs and configuration files for the routing and bridging functions.
- “***Root Filesystem***” The Root Filesystem consists of the kernel modules not built into the kernel, common and user libraries and all of the user space programs and configuration.
- “***SNMP***” The acronym for Simple Network Management Protocol.
- “***Manager***” The controlling device or computer. It can be a Windows, Linux or SnIP with the SNMP manager programs installed. Can also be considered the Client.
- “***Agent***” The controlled device. In this case the SnIP. The agent runs either/or snmpd and sub-agents such as the modem's m500-agent daemons to function as an agent. The agent performs a Server function, providing information is response to requests from a Client or manager.

- "***Trap***" A background task running in a daemon that determines if a parameter is outside normal bounds. The condition is trapped and an autonomous message sent to a client/manager.
- "***MIB***" The Management Information Base. This is the database of the individual elements (objects)that can be managed via SNMP. The MIB can be both textual and compiled. The MIB basically is an address book of OIDs and the parameters and methods they control. MIBs may have extensions of either ".mib" or ".txt" with the same context. They typically use the line endings from Linux which is slightly different from Windows.
- "***OID***" short for Object Identifier - The unique number that represents the position of a particular parameter in the SNMP tree structure. Standard Internet management OIDs begin with 1.3.6.1.2 followed by additional object identifiers. Datum Systems private OIDs begin with 1.3.6.1.4.1. followed by the PEN of the company developing the MIB (Datum Systems) and followed by up to 6 additional numbers separated by decimal points. All proprietary Datum System OIDs begin with 1.3.6.1.4.1.31978.
-

1.0.2 SNMP – Basic Model

SNMP consists of 3 main elements: The "Manager", the "Agent" and the device(s) that the agent controls. The manager and agent use a Management Information Base or "MIB". Notice that both the manager and agent must have the MIBs although they may be in different formats. The manager and agent communicate using the SNMP Protocol over the network between them. You can also think of the system as a client-server, where the agent is the server providing information in response to a client that requests it – somewhat like a web server providing web pages in response to a browser client requests.



There are multiple manager end and agent end programs installed in the SnIP: The main purpose of SNMP in the SnIP is to act as an agent, returning answers to requests from a network manager station, however the standard programs installed in the SnIP include the manager programs and one Snip can act as the manager for other SnIP agents.

1.0.3 SNMP – Tree Structure

SNMP information is arranged in a tree like structure with branches and leaves. The first few digits of the tree are defined by the ISO organization and begin with 1.3.6.1.4.1 which stands for iso.org.dod.internet.private.enterprises.manufacturer.device.model.

```
 1    3   6      1          4          1              31978
```

The OID used for the SnIP itself and its standard Linux network parameters is:
1.3.6.1.4.1.xxx.yyy... where xxx, yyy and following digits as defined by Net-SNMP standards.

And the top level OID for the M500 modem as controlled by a SnIP device is:
1.3.6.1.4.1.31978.3.1.2

All objects and events for the M500 series of modems will begin with this OID number. For example the full OID for the modem modulator IF output level is:

1.3.6.1.4.1.31978.3.1.2.2.2.2.1.3.0

A lengthy tree structure defined as a series of number would be very difficult to work with. The purpose of the MIB database is to provide a textual definition and translation between the numeric OID and human readable names for each of the parameters. When the proper MIBs for a device such as the M500 modem are loaded into the manager/browser or into a command line version of the net-SNMP manager/client programs, then much simpler names can be used. For example:

DATUM-M500-MODEM::modIfLevel.0

or more simply

modIfLevel.0

1.0.4 SNMP Agent (Server) Programs

The “agent” programs operate in the background as Linux “daemons” when started. These programs can be manually started and stopped (via /etc/init.d/snmpd or snmptrapd) or automatically started on boot up by configuration using the configwiz program. These programs are actually part of what is termed the “Master Agent”. A controlled device also commonly has “Sub-agents” which implement the specific controlled elements within the device. The SnIP as the controlled or “managed” element in the diagram above runs the Agent programs. There are 2 main ones which can run in the SnIP

1. “**snmpd**” is the standard net-SNMP agent daemon compiled with a standard set of MIBs built in. Those MIBs are for control of the standard Linux based processes, such as the Linux setup, the routing tables, etc.
2. “**m500-agent**”, A custom net-SNMP sub-agent which is written for control of the M500 series of satellite modems. When it is started in a system with a running snmpd it will automatically connect to the master agent, adding its capabilities to those built into the master. The m500-agent has its own MIB.

1.0.5 SNMP Manager (Client) Programs

Client end manager and helper applications (mainly in the /usr/bin directory for the SnIP) consists of multiple small programs. These programs are usually called as needed from the command line. Those installed in the SnIP include the following standard Net-SNMP programs. Other managers may have different or additional programs as part of their complement.

1. “**snmpbulkget**” is the management program used to get a defined bulk set of information from an agent.
2. “**snmpbulkwalk**” retrieve a sub-tree of values from an agent.
3. “**snmpdelta**” monitor delta differences in SNMP counter values
4. “**snmpdf**” display disk space usage on a network entity via SNMP.
5. “**snmpget**” is the management program used to get a single item of information from an agent.
6. “**snmpgetnext**” is the management program used to get the next MIB element from an agent relative to the last one requested.
7. “**snmpinform**” is the management program used to tell an agent who to send trap information to.
8. “**snmpnetstat**” get network status information from an agent.

9. “**snmpset**” is the management program used to set information on an agent. The set command only sets a single MIB item.
10. “**snmpstatus**” retrieves network statistics from an agent about the SNMP system itself.
11. “**snmptable**” retrieve an SNMP table from an agent and display it in tabular form.
12. “**snmptranslate**” translates one or more SNMP object identifier values from their symbolic (textual) forms into their numerical forms (or vice versa).
13. “**snmptrap**” fenerate a notification (trap) to report an event to the SNMP manager with the specified message.
14. “**snmpusm**” creates and maintains SNMPv3 users on a network entity.
15. “**snmpvacm**” creates and maintains SNMPv3 View-based Access Control entries on a network entity.
16. “**snmpwalk**” gets all information below the specified point in a tree from an agent.

1.0.6 SNMP Additional Resources

If you are unfamiliar with SNMP and its operation there is a wealth of information available on the web. A good starting point might be to Google for “SNMP Tutorial” and “SNMP Tools”.

There are also both free and commercial programs that allow one to browse the MIBs of controlled device agents. These programs run on Windows, Linux and Mac computers to provide either a textual or graphic display of information. There are also multiple open source tools of this sort available for Linux variants such as Fedora and Ubuntu among others. For Windows at least one MIB browser is available in a free version made by iReasoning.

2.0 SNMP Configuration

Proper SNMP requires configuring and enabling the SnIP Agent and also configuring and running an SNMP manager or “browser”.

2.1 SNMP SnIP Configuration

SnIP SNMP configuration consists of setting the correct variables for you system in the SNMP configuration files and enabling SNMP and or SNMP Trap operations from the configwiz program.

2.1.1 SNMP Configuration Basics

When enabled, the SNMP daemons wait and listen for messages on any interface port containing messages with the SNMP protocol. If the messages are addressed to this unit’s IP Address and are properly formed the agent will act upon the message and return a response.

2.1.1.2 Enable or Disable SNMP via Web User Interface

Log into a SnIP web address and select the SnIP >> SNMP page. The options available are to start or stop the SNMP master agent daemon and the SNMP Trap agent daemon immediately or to set the configwiz program so that the selected daemons with start automatically on each reboot. The modem sub-agent named “m500-agent” is automatically started and stopped with the master agent. These are the equivalent of the two methods shown below.

2.1.1.3 Enable or Disable SNMP via configwiz

From a terminal session (console or telnet or ssh) you can control whether SNMP or SNMP Trap daemons will start on SnIP bootup. The configwiz options (1) and (2) select the SNMP master agent and the SNMP Trap agents respectively. Save the changes and on the next reboot the selected agents will start automatically.

2.1.1.4 Enable or Disable SNMP Manually for immediate use

From a terminal session (console or telnet or ssh) you can control start or stop the SNMP Master agent or SNMP Trap daemons. These commands are **not** persistent and a reboot will not return to the same state unless it is saved via the configwiz method above.

/etc/init.d/snmpd start *or* ***service snmpd start***

Or to stop

/etc/init.d/snmpd stop *or* ***service snmpd stop***

The standard Net-SNMP Master agent in the SnIP reads its configuration setting from a file named **"snmpd.conf"** located in the **"/etc/snmp/"** directory. It has some default settings which will allow use and access to all of the Linux and m500 modem parameters using the default community string of **"public"** for read only operations and **"private"** for read-write operations. The user is responsible for setting up the configuration file as needed for your operating system by editing and/or transferring to the SnIP.

2.1.2 snmpd Master Agent – Built-in Module List

The master Agent has a lot of information MIBs pre-configured into it at compile time for standard networking and Linux computer parameters. You can see the list of these using the command

"***net-snmp-config --snmpd-module-list***"

```
 mibII/system_mib
 mibII/sysORTable
 mibII/at
 mibII/interfaces mibII/ip
 mibII/snmp_mib
 mibII/tcp
 mibII/icmp
 mibII/udp
 mibII/vacm_vars
 mibII/setSerialNo
 ucd-snmp/memory
 ucd-snmp/vmstat
 ucd-snmp/proc
 ucd-snmp/versioninfo
 ucd-snmp/pass
 ucd-snmp/pass_persist
 ucd-snmp/disk
 ucd-snmp/loadave ucd
-snmp/extensible
 agent/extend
 ucd-snmp/errormib ucd
-snmp/file ucd
-snmp/dlmod
 ucd-snmp/proxy
 ucd-snmp/logmatch
 snmpv3/snmpEngine
 snmpv3/snmpMPDStats
 snmpv3/usmStats
 snmpv3/usmConf
 snmpv3/usmUser
 notification/snmpNotifyTable
 notification/snmpNotifyFilterTable
 notification/snmpNotifyFilterProfileTable
 target/snmpTargetAddrEntry
 target/snmpTargetParamsEntry
```

Note:

Testing SNMP on a Single SnIP

The SnIP contains all of the manager and agent end programs. It can query itself for information using the device address of "**localhost**" in place of an IP Address in most of the commands shown in this document.

To see if a particular agent daemon is currently running on the SnIP, use the "**ps**" command from a terminal session. It will show the snmpd and m500-agent daemons in its list if they are running.

You can also use the SnIP web server page SNMP to show the current status of the SNMP daemons.

```
 target/target
 target/target_counters
 agent/nsTransactionTable
 agent/nsModuleTable
 agent/nsDebug
 agent/nsCache
 agent/nsLogging
 agentx/master
 agentx/subagent
 utilities/override
 host/hr_system
 host/hr_storage
 host/hr_device
 host/hr_other
 host/hr_proc
 host/hr_network
 host/hr_print
 host/hr_disk
 host/hr_partition
 host/hr_filesys
 host/hr_swrun
 host/hr_swinst
 smux/smux
 util_funcs
 mibII/var_route
 mibII/kernel_linux
 mibII/ipAddr
 mibII/route_write
 mibII/tcpTable
 mibII/udpTable
 mibII/vacm_context
 mibII/vacm_conf
 utilities/execute header_complex
 agentx/protocol
 agentx/client
 agentx/master_admin agentx/agentx_config
 if-mib/data_access/interface_common
 if-mib/data_access/interface_linux
 if-mib/data_access/interface_ioctl
[snip-61:0 ~]
```

2.1.3 m500-agent Sub Agent – Built-in Parameter List

The module list above does not show the modem MIB entries which are built into the “**m500-agent**” program which is a sub-agentx entity. This sub-agent currently contains the following parameters accessible via SNMP. This list is expected to change in the future.

Name	Type	Example Value
UnitID.0	STRING	Mike's Test 1
unitModel.0	STRING	M500L
unitSerialNo.0	Gauge32	13509
unitVersion.0	STRING	1.18.040
unitFeatureSet.0	STRING	M505

Name	Type	Example Value
unitFecAType.0	Gauge32	2
unitFecASwVersion.0	Gauge32	110
unitFecBType.0	Gauge32	0
unitFecBSwVersion.0	Gauge32	0
unitInstalledIntfOption.0	Gauge32	3
modAlm.0	INTEGER	alarmOff(0)
demodAlm.0	INTEGER	alarmOn(1)
refUnitAlm.0	INTEGER	alarmOff(0)
clkAlm.0	INTEGER	alarmOff(0)
rdnAlm.0	INTEGER	alarmOff(0)
rdnNoBackupAlm.0	INTEGER	alarmOff(0)
rdnBackupinAlm.0	INTEGER	alarmOff(0)
overTempAlm.0	INTEGER	alarmOff(0)
modHdrAlm.0	INTEGER	alarmOff(0)
demHdrAlm.0	INTEGER	alarmOff(0)
fecAFailAlm.0	INTEGER	alarmOff(0)
fecBFailAlm.0	INTEGER	alarmOff(0)
intOptAlm.0	INTEGER	alarmOff(0)
sndDtaActAlm.0	INTEGER	alarmOff(0)
rcvDtaActAlm.0	INTEGER	alarmOff(0)
txEnable.0	INTEGER	disabled(0)
rxLocked.0	INTEGER	disabled(0)
redundancy.0	INTEGER	disabled(0)
onLine.0	INTEGER	enabled(1)
unitRdnEnable.0	INTEGER	enableInt(1)
unitRdnOnLine.0	INTEGER	OnLine(1)
unitRdnSwitchOn.0	INTEGER	AnyAlarm(0)
unitRdnHoldDelay.0	INTEGER	6 tenths of seconds
unitRdnSendConfig.0	INTEGER	idle(0)
unitRdnSwitchToBackup.0	INTEGER	idle(0)
modCxrState.0	INTEGER	disabled(0)
modTestState.0	INTEGER	normal(0)
cxrAlm.0	INTEGER	alarmOff(0)
apcAlm.0	INTEGER	alarmOff(0)
refAlm.0	INTEGER	alarmOff(0)
dLckAlm.0	INTEGER	alarmOff(0)
lvlAlm.0	INTEGER	alarmOff(0)

Name	Type	Example Value
loAlm.0	INTEGER	alarmOff(0)
stpAlm.0	INTEGER	alarmOff(0)
sysAlm.0	INTEGER	alarmOff(0)
ocxoAlm.0	INTEGER	alarmOn(1)
bucPwrAlm.0	INTEGER	alarmOff(0)
cxrEnable.0	INTEGER	disabled(0)
cxrOnline.0	INTEGER	sending(1)
rtsMute.0	INTEGER	muteOff(0)
modTestModulation.0	INTEGER	normal(0)
modSymbolRate.0	INTEGER	2676459 Hz
modCxrALC.0	INTEGER	-64 V in tenths
modLoAFC.0	INTEGER	113 V in tenths
modStepAFC.0	INTEGER	116 V in tenths
modIfFrequency.0	INTEGER	1665000000 Hz
modRfFrequency.0	INTEGER	14252000 kHz
modLoFrequency.0	INTEGER	13050 MHz
modIfOffset.0	INTEGER	0 Hz
modIfLevel.0	INTEGER	-350 dBm in tenths
modIfCarrierAUPC.0	INTEGER	disabled(0)
modIfOutput.0	INTEGER	disabled(0)
modIfSpectrumDirection.0	INTEGER	normal(0)
modIfSpectrumFilter.0	INTEGER	IESS(0)
modIfCarrierEnable.0	INTEGER	disabled(0)
modIfCarrierMuteConfig.0	INTEGER	auto(0)
modIfModulation.0	INTEGER	QPSK(1)
commitIfConfigModemSettings.0	STRING	No errors
modDataBitRate.0	Unsigned32	4000000
modDataFecType.0	INTEGER	TPC(4)
modDataFecOption.0	INTEGER	0
modDataFecCodeRate.0	STRING	3/4-4k
modDataDiffEncoder.0	INTEGER	disabled(0)
modDataScrambler.0	INTEGER	Auto(1)
modDataClockSource.0	INTEGER	internal(0)
modDataReadSolomonMode.0	INTEGER	Disabled(0)
modFECTypeOptionCrate.0	INTEGER	404
commitModemDataConfigSettings.0	STRING	No error
demodCxrState.0	Gauge32	enabled(1)

Name	Type	Example Value
demodTestState.0	INTEGER	normal(0)
dtaAlm.0	INTEGER	alarmOn(1)
bckAlm.0	INTEGER	alarmOff(0)
lvlDemodAlm.0	INTEGER	alarmOn(1)
ebAlm.0	INTEGER	alarmOff(0)
agcAlm.0	INTEGER	alarmOn(1)
refDemodAlm.0	INTEGER	alarmOff(0)
loDemodAlm.0	INTEGER	alarmOff(0)
stpDemodAlm.0	INTEGER	alarmOff(0)
sysDemodAlm.0	INTEGER	alarmOff(0)
lnbPwrAlm.0	INTEGER	alarmOn(1)
cxrStatus.0	INTEGER	locked(1)
demodEbNo.0	INTEGER	0 dB in tenths
demodIfOffset.0	INTEGER	0 Hz
rcvCxrLevel.0	INTEGER	-822 dBm in tenths
rcvEstBer.0	INTEGER	-1 Scale Factor by 10E-10
rcvEstSer.0	INTEGER	-1
ifLoopBack.0	INTEGER	disabled(0)
demodSymbolRate.0	INTEGER	2548977 Hz
demodCxrAGC.0	INTEGER	100 V in tenths
demodLoAFC.0	INTEGER	66 V in tenths
demodStepAFC.0	INTEGER	67 V in tenths
demodIDcOffset.0	INTEGER	13 V in tenths
demodQDcOffset.0	INTEGER	9 V in tenths
demodIfFrequency.0	INTEGER	1565000000 Hz
demodRfFrequency.0	INTEGER	12225000 kHz
demodLoFrequency.0	INTEGER	10750 MHz
demodIfSweepRange.0	Unsigned32	30000
demodIfCarrierLevel.0	INTEGER	-822 dBm in tenths
demodIfEbNo.0	INTEGER	0 dB in tenths
demodIfModulation.0	INTEGER	QPSK(1)
demodIfSpectrumDirection.0	INTEGER	normal(0)
demodIfSpectrumFilter.0	INTEGER	IESS(0)
demodIfReceiveCarrierStatus.0	INTEGER	2
commitIfConfigDemodSettings.0	STRING	No error
demodDataBitRate.0	Unsigned32	3809477
demodDataFecType.0	INTEGER	TPC(4)

Name	Type	Example Value
demodDataFecOption.0	INTEGER	0
demodDataFecCodeRate.0	STRING	3/4-4k
demodDataDiffEncoder.0	INTEGER	disabled(0)
demodDataDeScrambler.0	INTEGER	Auto(1)
demodDataClockSource.0	INTEGER	rcv-clock(0)
demodDataReadSolomonMode.0	INTEGER	Disabled(0)
demodFECTypeOptionCrate.0	INTEGER	404
commitDemodDataConfigSettings.0	STRING	No error
interfaceType.0	INTEGER	Ethernet-SnIP(8)
intfOnLine.0	INTEGER	enabled(1)
intfRxLocked.0	INTEGER	disabled(0)
intfRedundancy.0	INTEGER	disabled(0)
terrestrialLoopback.0	INTEGER	disabled(0)
satelliteLoopback.0	INTEGER	disabled(0)
intfAlm.0	INTEGER	alarmOff(0)
intfFail.0	INTEGER	alarmOff(0)
intfReset.0	INTEGER	alarmOff(0)

2.1.3.1 m500-agent Sub Agent – BUC/LNB Built-in Parameter List

BUC and LNB Objects
The following SNMP objects are automatically added when the modem/SnIP connected to is an L-Band version of the M500 series modems (M500L type). These objects are contained within the standard MIB named "*DATUM-M500-MODEM-MIB*".

Name	Type	Example Value
BUCCurrent.0	INTEGER	1600 mA
BUCVoltage.0	INTEGER	478 V in 0.1 V steps
BUCTxFrequency.0 [See Note 1]	INTEGER	0 kHz
BUCPwrAlm.0	INTEGER	alarmOff(0)
BUCDCPower.0	INTEGER	disable(0)
BUC10MHzReference.0	INTEGER	disabled(0)
BUCHighCurrentLimit.0	INTEGER	600 I in 10 mA steps
BUCLowCurrentLimit.0	INTEGER	5 I in 10 mA steps
BUCLowVoltageLimit.0	INTEGER	80 V in 0.1 V steps
BUCLOFrequency.0 [See Note 1]	INTEGER	0 MHz
LNBCurrent.0	INTEGER	224 mA
LNBRxFrequency.0 [See Note 1]	INTEGER	0 kHz
LNBPwrAlm.0	INTEGER	alarmOff(0)
LNBDCPower.0	INTEGER	Off(0)

Name	Type	Example Value
LNB10MHzReference.0	INTEGER	disabled(0)
LNBHighCurrentLimit.0	INTEGER	500 mA
LNBLowCurrentLimit.0	INTEGER	20 mA
LNBLOFrequency.0 [See Note 1]	INTEGER	0 MHz

[Note 1]Currently the BUC and LNB RF frequencies and LO frequencies are coupled to those in the L-Band Modem front panel settings. Use of these parameters either on the modem front panel or in SNMP causes the modem's standard IF frequency to display as RF. That causes problems in SNMP and can result in modified IF frequency values which can result in lost communications. The modem values for the BUC and LNB LO frequency should always be set to “0” if SNMP is used. Virtual LO and RF frequencies for transmit and receive are provided in the modLoFrequency, modRfFrequency, demodLoFrequency and demodRfFrequency parameters.

2.2 SNMP Browser/Manager Configuration

The Browser or Manager program is what queries the SnIP for information. It also needs a copy of the MIBs for equipment that it will want access to, plus knowledge of the community string and security arrangements for each piece of that equipment. For access to the SnIP a standard set of MIBs built into most net-SNMP browsers can be used since it is basically a Linux device. For the unique Datum Systems' M500 series modems the MIBs required are named:

1. DATUM-REG.txt – The base Datum Systems Registration MIB.
2. DATUM-M500-MODEM-MIB.txt – The M500 series modem MIB.
3. DATUM-MODEM-TRAPS-MIB.txt – The available traps for Datum System's modems.
4. DATUM-TRAP-CONFIG.txt – The MIB which allows configuring Traps.

Note that these files have an extension of “.txt” but some applications prefer the extension of “.mib” and they can be changed at will. Each Master or browser uses its own procedures for loading and using MIBs and you should refer to that documentation. The Net-SNMP suite of Master processes is common to Linux and those will be briefly described below in Section 3 with examples to show their use. Our original .mib versions are created in Linux and use the Linux new line conventions which may not display correctly using a Windows program such as “Notepad”. We attempt to reformat the “.txt” versions of these programs with the standard Windows new line characters.

Assuming Net-SNMP installed on a Linux computer, the MIBs are most commonly maintained in the computer's **“/usr/share/snmp/mibs**” directory. This directory may be in **“/usr/share/mibs/netsnmp**” on versions of Net-SNMP after 5.4.2.1. Download the current “DATUM-REG.txt” and “DATUM-M500-MODEM-MIB.txt” (and the trap MIBs if desired) from the DatumSystems.com website and place the files in that directory. Alternately you can always get the MIBs from the SnIP itself, as they are maintained in the SnIP's **“/usr/share/snmp/mibs**” directory. They can be accessed using scp, tftp, ftp, WinSCP or rsync. To put the mibs into a Linux computer you will probably need root access or use of “sudo” to access that directory. SNMP commands such as walk, get, set, etc may scan the MIBs located there for your given parameter name. But the proper method is to tell Net-SNMP which MIBs to load beyond its standard set. A typical sequence to load the Datum Systems M500 modem MIBs might be:

```
export MIBS=+DATUM-GLOBAL-REG
export MIBS=+DATUM-M500-MODEM
export MIBS=+DATUM-MODEM-TRAPS
```

Note that these MIB names are **not** the names of the files containing the MIBs, but the name of the top level MIB definition within that file. Net-SNMP looks into the files to locate the MIB definition names.

After loading the MIBs you can then issue commands using the shorthand names of parameters. For example to get the current Demodulator Rcv IF Frequency you could use “snmpget” as in:

```
$ snmpget -v 2c -c public 192.168.15.64 demodIfFrequency.0
```

SNMP is also possible without loading the MIB, but you will have to know the full “OID” address of the modem parameters. They start at 1.3.6.1.4.1.31978.3.1.2 as described above.

2.2.1 Ubuntu 10.10 or others using net-SNMP 5.4.3 and above

If you have recently updated your Linux computer and the new version of net-SNMP is 5.4.3 or above then file locations may have changed and net-SNMP may not work correctly with the SnIP. The following changes to your Linux computer may be needed to recover a working system.

In Ubuntu 10.10 Net-SNMP was upgraded from Version 5.4.2.1 to 5.4.3 and all of the standard mibs were removed. The default location was also moved to a different directory, now in ***/usr/share/mibs/netsnmp*** or ietf or iana.
A) To get back the standard mibs you need to install a new package

```
sudo apt-get install snmp-mibs-downloader
```

The package installation normally also downloads the standard mibs and puts them into 3 folders under /usr/share/mibs. There is also a resulting program named "download-mibs" but you apparently don't need to use it since the actual download is done by the install.

B) You need to modify the snmp.conf configuration file by removing the comment # on the last line which reads “#mibs :”

```
sudo gedit /etc/snmp/snmp.conf
```

The last line should now be -

```
mibs :
```

C) Then you need to move the Datum System's mibs from the /usr/share/snmp/mibs folder to one of the new ones in /usr/share/mibs, for example in the netsnmp folder. They may work equally well in another folder.

```
sudo cp /usr/share/snmp/mibs/DATUM*.txt /usr/share/mibs/netsnmp/
```

You could also get these MIB files from the SnIP itself if you did not have them in the snmp/mibs folder.

3.0 Basics for Using SNMP – Commands and Examples

The SnIP is mainly intended as an Agent, returning responses to SNMP commands. It can also act as a manager, getting information from other SnIPs or devices, but that application is not covered here except to note that the capability exists.

The following commands and examples are representative of operations and responses from a Linux computer running standard Net-SNMP, which is the only base system we can describe for non-proprietary SNMP functions. For operating any commercial or proprietary browser or NMS you should consult their own documentation. Before these operations will succeed you must have checked the computer’s package manager to determine if Net_SNMP is loaded and if not then install it.

Before being able to access SNMP information from the SnIP you must know its “community” string that is defined for specific areas in the SNMP configuration file located in and named **“/etc/snmp/snmpd.conf”**. This file is located in the SnIP itself. It has been set up with a base configuration to allow you to check the functioning of SNMP on the SnIP, but should be configured properly by the user for system, security and control methods desired. That subject is beyond the scope of this document.

A typical command typed at the console of an SNMP controller might be something like the following. The $ is the prompt from the command line in Linux and is not entered as part of a command.

```
$ snmpget -v1 192.168.15.35 -c public iso.3.6.1.2.1.1.9.1.3.6
```

Which would mean: Get the public community value of the numbered iso parameter shown from the SnIP at the IP Address shown using version 1 protocol. A response in this case might be:

```
iso.3.6.1.2.1.1.9.1.3.6 =STRING: "View-based Access Control Model for SNMP."
```

The SnIP is set up by default with the “***community string***” value of “**public**” for read operations. This next example will walk through the full tree for the “SNMPv2-MIB” which is normally a predefined MIB loaded into Net-SNMP by default and is also part of the standard MIBs configured into the SnIP's SNMP master agent.

```
$ snmpwalk –v 2c -c public localhost iso.3.6.1.2.1.1
```

This will produce a huge amount of information containing hundreds of lines of responses.

Assuming that the M500 Modem MIBs have been loaded as described above, commands can be issued to view parameters or blocks of parameters as shown in the following examples:

Get the Demodulator IF Frequency setting from the modem/SnIP at IP address 192.168.15.64:

```
$ snmpget -v 2c -c public 192.168.15.64 demodIfFrequency.0
```

Set the Modulator Data Bit Rate to 2.048 Mbps

```
$ snmpset -v 2c -c private 192.168.15.64 modDataBitRate.0 u 2048000

DATUM-M500-MODEM::modDataBitRate.0 = Gauge32: 2048000
```

Note that in these examples the name of the parameter must be followed by a “.0”. The response from the second example for “set” is shown below the command. Also note that an Unsigned32 value is returned as a Guage32 in this particular version of Net-SNMP, but the “u” type in the set command is required. Also note in the “Set” example that we had to use the read-write community value of “private” for the command to work.

Net-SNMP attempts to “parse” the names given and searches through the available MIBs for matching entries. So for example a “Walk” command can find all items beginning with (and in some cases containing) a name. Names are not case sensitive either. So the following is an example of looking for all modulator alarms.

```
$ snmpwalk -v 2c -c public 192.168.15.64 modAlarms

DATUM-M500-MODEM::cxrAlm.0 = INTEGER: alarmOff(0)
DATUM-M500-MODEM::apcAlm.0 = INTEGER: alarmOff(0)
```

```
DATUM-M500-MODEM::refAlm.0 = INTEGER: alarmOff(0)
DATUM-M500-MODEM::dLckAlm.0 = INTEGER: alarmOff(0)
DATUM-M500-MODEM::lvlAlm.0 = INTEGER: alarmOff(0)
DATUM-M500-MODEM::loAlm.0 = INTEGER: alarmOff(0)
DATUM-M500-MODEM::stpAlm.0 = INTEGER: alarmOff(0)
DATUM-M500-MODEM::sysAlm.0 = INTEGER: alarmOff(0)
DATUM-M500-MODEM::ocxoAlm.0 = INTEGER: alarmOn(1)
DATUM-M500-MODEM::bucPwrAlm.0 = INTEGER: alarmOff(0)
```

The same response would have resulted from using “**modAlm**” or “**modal**” or “**moda**”. Similarly you could request all demodIF objects by walking over “**demodif**”. Note also that the “**walk**” command does not use the “.0” on the name end as get and set require.

For reading just the BUC or LNB values from an L-Band modem using snmpwalk you must use the terms “lbuc” and “llnb” for the SNMP parsing routines to properly find them.

3.1 Basics for Using SNMP – Best Practices and Cautions

Neither SNMP or the satellite modem is “magic”. SNMP is designed as a very general purpose tool mainly for networking and its parameters. There are cases where it does not match well with modem or link capabilities. It is also intended more for machine to machine communications rather than human readability.

3.1.1 SNMP and Complex Commands

A satellite modem is a complex piece of equipment and SNMP has a rigid set of rules that cannot easily accommodate some types of settings easily. A good example of this is the modulator or demodulator FEC settings.

There are 3 interactive settings for the FEC Type, FEC Option and FEC Code Rate which will return an error in attempting to set them in the wrong order. To avoid the problem the combination parameters have been added which are named “modFECTypeOptionCrate” and “demodFECTypeOptionCrate” for the mod and demod respectively. A 3 digit number is entered with these settings corresponding to the Type, Option and Code Rate in that order. These numbers are the same as the option numbers that would be selected for each parameter on the from panel. For example The setting value of “404” represents TPC (4), Advanced Option Mode (0) and Code rate 3/4-4k (4). The method for handling this complex case is handled differently on the modem front panel and on the web page configuration.

You should also be cautioned about specific settings that are “out of range” but the modem will make a best effort and set the value to the maximum (or minimum) possible. This is true of the mod and demod data bit rate and the lower level alarm limits on the the demodulator carrier level. The maximum or minimum for these values can vary considerably based on modulation mode, interfaces, installed options and FEC types, etc. So if a modulator bit rate is set to 12 Mbps but the maximum possible for the FEC type is 5 Mbps, then the modem will set the value to 5 Mbps. SNMP has no response error type to represent this case, and returns an error.

It is good practice to read the value of a parameter again when an error is reported after attempting to set it.

3.1.2 SNMP Bandwidth Caution

As you may have noticed doing some basic “walk” experiments, SNMP can return a significant amount of information easily. SNMP is also very inefficient from a bandwidth standpoint. Considering that much SNMP monitor traffic will be over a satellite link with limited bandwidth and resources, care should be exercised to reduce the amount of SNMP traffic to a minimum. If a management system is polling every possible parameter every second, then there may be little room for actual customer data on the link.

3.1.3 Far End SNMP Modem Control Caution

This might be a good place to say “think before you act”. Satellite links often connect to remote unmanned sites. It is extremely easy to change a setting on the far end of a link at a remote site and cause the link to go down, losing connectivity and control. Then someone will have to go to that site to restore it to operation.

The M500 series of modems are designed with the ability to automatically go to several predefined configurations if the receive carrier is lost for longer than a specified period of time. This can be part of a recovery strategy if thought out in advance and the consequences of signal loss are known to operations personnel. It could also lead to confusion if not used properly. See the M500 modem main manual for more information on the “ACR” or Automatic Configuration Recovery feature.

4.0 Troubleshooting

Does none of this seem to work for you? If you cannot get SNMP to work with the SnIP or the Modem then its time to do some basic checking and troubleshooting.

4.1 Basic Requirements for SNMP to Work and Tests

SNMP can be difficult for newcomers. All of the following must be in place for SNMP to work:

1. The SnIP must be at a valid IP Address and Mask for the connected network.
2. You must have a valid browser/manager computer to access the SnIP' SNMP agent.
3. The management computer must be accessible from the SnIP.
4. You must have the SnIP SNMP master agent running. The m500 sub agent must also be running for modem control.
5. The manager computer must be running Net-SNMP or an SNMP Browser program.
6. The SNMP browser/manager must have the appropriate MIBs loaded.

Test Connectivity:

One first test that will determine if 1, 2 and 3 conditions are met is to “ping” the SnIP from the management computer. If you do not get a response then the network and addresses are possibly incorrect.

Test Agents Running:

Next, log into the SnIP via telnet using the default user name and password of “root” and “datum” or others if you have changed them. Use the Linux “ps” command to view the running processes. You should see the entries for at least the “snmpd” daemon and the m500-agent as illustrated below. This verifies item 4.

```
  212 root      2768 S   /usr/sbin/snmpd -Lsd -Lf /dev/null -p /var/run/snmpd.
  217 root      1792 S   /root/m500-agent
```

The exact lines above may change with various versions of SNMP, but both the “snmpd” master agent and the “m500-agent” modem sub-agent should be present.

The m500-agent directory will probably vary depending on the location it is run from. If you do not see these items running then the configwiz has probably not been used to enable the daemons. You can also start the daemons manually from a telnet session using the command

```
/etc/init.d/snmpd start
```

or in a version 0.6.xx filesystem the slightly easier command can be used:

```
service snmpd start
```

Either of these methods of manually starting the SNMP agent and modem agent daemons should indicate success. You can check that they are running by using the “ps” command after which you should see the daemons listed as above.

The web interface also contains a page at SnIP>>SNMP which allows both immediate and persistent SNMP agent control.

Test Agent Functionality:

Even if the programs are running you can double check their functionality by testing the SnIP against itself. This requires logging into a console, Telnet or SSH session on the SnIP. Use any of the following commands to see if you get valid responses. Any of these 3 should work, returning responses assuming the SnIP is properly set up. The first uses the SnIP's own IP Address, the second uses the local loopback address and the third uses the default host name of “localhost”. A valid response will be a listing of outputs saying “Cannot find module (xxxx)” plus several lines of output for the values at the end.

```
snmpwalk -v2c -c public 192.168.15.64 iso.3.6.1.6.3.10
snmpwalk -v2c -c public 127.0.0.1 iso.3.6.1.6.3.10
snmpwalk -v2c -c public localhost iso.3.6.1.6.3.10
```

Later versions of the SNMP startup script will automatically load the basic default MIBs which are part of the SnIP filesystem. If there are some lines showing the tag “Cannot find xxx” issue use the command “export MIBS=ALL” and then re-issuing any of these commands. The “Cannot find” comes about because the text MIBs are not loaded into the manager, which would be difficult because the size of all these text MIBs is in the MegaByte range.

You can do further testing at this point to check the modem agent. This command should return a fairly short response.

```
snmpwalk -v2c -c public 127.0.0.1 iso.3.6.1.4.1.31978.3.1.2.2.1.1
```

and prove that the m500-agent is running and returning responses. Now we are really sure that item 4 is satisfied.

Test Manager/Browser Functionality:

If you've made it this far and you cannot get information in the browser or Net-SNMP from the SnIP then you should double check the item 5 requirement by looking at the browser configuration again. Make sure you are using the correct IP Addresses and community string in the configuration. For Net-SNMP this is on the command line for each command. For browsers this is typically a GUI configuration item. For example iReasoning's browser has a configuration setup at Tools>>Options and the Agents tab which is used to set up the version and read and write community strings for each IP Address.

Test Manager/Browser MIB Load:

The last item to check is that the MIBs are loaded in the browser or Net-SNMP. Read Section 2.2 above again and insure that the proper MIBs exist and are loaded. Full browsers normally show which MIBs are loaded as part of their window presentation. With Net-SNMP the standard MIBs loaded by default should allow operation with the SnIP.

5.0 SNMP Traps

SNMP Traps are background processes that run and monitor specified parameters. If the parameters fall outside of proper ranges then a “trap” is generated and the manager is notified in one of two ways. Either a trap message or a notification is sent to the manager/browser monitoring the system.

5.1 Configuring SNMP Traps

SNMP Traps and their documentation are still in development. As an alternative the standard modem polling routines are capable of capturing and sending SNMP compliant trap/notification messages in version 2c format. This capability requires modem software version 1.22 or above and SnIP filesystem software version 0.6.29 or above, or ipkg “modem-control” package version 0.6.29 applied to previous filesystem versions.

This trap generation method is very flexible and does not even require that SNMP be running in the SnIP for it to operate. Only that a standard MIB browser be running with the DATUM-GLOBAL-REG, DATUM-M500-MODEM, DATUM-MODEM-TRAPS and DATUM-TRAP-CONFIG-MIB loaded.

Only two of the listed traps in the DATUM-MODEM-TRAPS mib are used, the mod alarm and demod alarm. But the modem polling is capable of sensing many more alarm types, so when the trap is reported it uses the trap location string to indicate more detail about the specific failure. For example if a BUC fault is detected in an L-Band modem at a location of for example “Shenzhen” there there would likely be 2 traps received at the monitoring location. One would be a modem alarm with the location listed as “Modem Alarm On, Shenzhen” and the other would be “BUC Alarm On, Shenzhen”. When the alarm was cleared you would also expect 2 traps listing the BUC and modem Alarm Off. This is more powerful than many simple trap definitions.

5.1.1 SNMP Notification Setup Procedure

The setup to send notification alerts to a specific SNMP trap receiver involves the following steps:

1. Edit the “/etc/config/mdm-events” file to turn on alerts.
2. Edit the “/etc/config/alert.conf” file to set the destination, community string and location variables.
3. Restart the modem polling service.

This manual procedure is not difficult, but there is a new helper script to make the alert setup process very easy. The script is named “**setalerts**” and its use from any terminal command line is as follows:

- If you want to see usage instructions and the current settings of all the variables relating to alerts just type “**setalerts**” without any arguments and help plus current settings will be returned.
- When you are ready to turn alerts on you need to know the IP Address of the mib client browser that will receive the alerts formated as SNMP trap messages. It is also advisable to set a location for the modem so that a received alert will be identified as to its source. For example the modem may be at a station in Sao Paolo, Brazil and the trap receiver might be at 129.67.23.182.
- From a SnIP terminal session you want to tell the SnIP to turn on alerts and set those two variables. Here is the command:

```
setalerts -v on ip=”129.67.23.182” loc=”Sao Paolo, Brazil”
```

The -v is optional and sets the output to verbose mode. It is used here for a first time so that you can see each action taking place including the stop and start of the modem polling daemon if changes are made. The “on” argument tells it to turn alerts on, while “off” would have turned them off. The ip and loc keywords are for the destination IP Address and modem location strings respectively. The quotation marks are especially required for the location because it contains spaces. Items identical to current settings are ignored and IP Addresses are checked for validity.

Other keywords for arguments that can be set are:

- ***mask=number*** – this number represents those bits in the modem polling status that will be turned into alerts when their value changes. The default value of 0x1800001c is recommended. The initial “18” represent the LNB and BUC alarms for L-Band modems, while the “1c” represents the demod alarm, mod alarm and demod lock alarm respectively. Although the current modem trap definitions in the M500-
- ***cm=”community string”*** - the default for this is “public”, but you can use any value you have set up in your system.

keywords and arguments can be entered on the line in any order with the exception that the verbose -v option must be first if used. There can be no spaces between the keyword, equal sign and keyword value. A space must be used between each keyword/value and the next.

Manual Alert Configuration

If you wish to manually configure alerts here are the basic changes required n more detail :

A. Edit the mdm-events file, for example
nano /etc/config/mdm-events – Locate the line near the end of the file that says

```
   SEND_ALERTS=0
```

and change the 0 to “1” if you want to turn it on.

Locate the line near the end of the file that says

```
   alert_mask=0x1800001c
```

and change the value to represent the proper bits that you want to send a trap on. See the SnIP Modem Control Guide and the table in Section 1.1.7.1 which shows the contents of the Unit Polling Status command packet. The bits in Bytes 6 through 10 are those that can be detected as specified by the mask value.

Save the file using Ctrl O (and return to save as the same name), then Ctrl X.

B. Edit the alert.conf file, for example
nano /etc/config/alert.conf – Near the beginning of the file are 3 lines similar to the following:
that say:

```
  dest_ip="192.168.15.104"
  notif_comty="public"
  loc="Santa Cruz, CA"
```

Edit each of the values as desired and save the file using Ctrl O (and return to save as the same name), then Ctrl X.

C. Restart the modem polling daemon to have it reload the mdm-events file.

```
   service mdmpolld restart
```

or

```
   /etc/init.d/mdmpolld restart
```

--End of Document.