DPS Telecom
"Your Partners in Network Alarm Monitoring"

"How to Read and Understand the SNMP MIB..."

Instantly Understand Any SNMP Device...

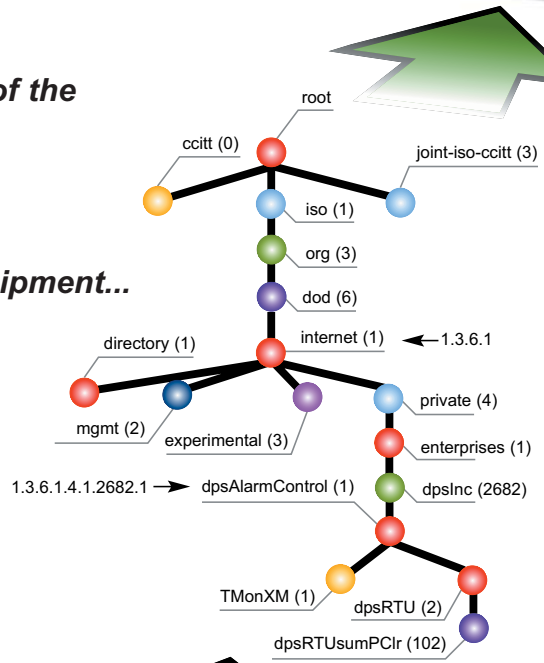
Is your SNMP knowledge good enough? What if you could ***instantly understand*** the capabilities of any SNMP device?

Reading the SNMP MIB file is the **best way** to do that, and **this guide teaches you how**. This SNMP MIB white paper is **a must-read for anyone who works with SNMP...**

Read this complete guide now:

- ***Understand the purpose and function of the MIB...***
- ***Learn how to read the MIB...***
- ***Use the MIB to evaluate any SNMP equipment...***

T/Mon NOC



NetGuardian 832A

Version 3.1
Released February 5, 2008

Written by Marshall DenHartog

www.dpstelecom.com · 1-800-622-3314

About the Author

Marshall DenHartog has over ten years' experience working with SNMP, including designing private MIB extensions, creating SNMP systems for multiple platforms, and developing SNMP-based monitoring for several nationwide networks.

DenHartog's experience with both the theoretical and practical sides of SNMP have equipped him to write a straightforward guide to the SNMP Management Information Base.

Contents

What is the MIB?

The MIB, or Management Information Base, is an ASCII text file that describes SNMP network elements as a list of data objects. Think of it as a dictionary of the SNMP language — every object referred to in an SNMP message must be listed in the MIB.

What does the MIB do?

The fundamental purpose of the MIB is to translate numerical strings into human-readable text. When an SNMP device sends a Trap or other message, it identifies each data object in the message with a number string called an object identifier, or OID. (OIDs are defined more fully later in this paper.)

The MIB provides a text label called for each OID. Your SNMP manager uses the MIB as a codebook for translating the OID numbers into a human-readable display.

Why do I need the MIB?

Your SNMP manager needs the MIB in order to process messages from your devices. Without the MIB, the message is just a meaningless string of numbers.

How do I get the MIB into my SNMP manager?

Your SNMP manager imports the MIB through a software function called compiling. Compiling converts the MIB from its raw ASCII format into a binary format the SNMP manager can use.

Why is the MIB important?

Because as far as SNMP managers and agents are concerned, if a component of a network device isn't described in the MIB, it doesn't exist.

For example, let's say you have an SNMP RTU with a built-in temperature sensor. You think you'll get temperature alarms from this device — but you never do, no matter how hot it gets. Why not? You read the RTU's MIB file and find out that it only lists discrete points, and not the temperature sensor. Since the sensor isn't described in the MIB, the RTU can't send Traps with temperature data.

Why do I need to understand the MIB?

As you can see, the MIB is your best guide to the real capabilities of an SNMP device. Just looking at the physical components of a device won't tell you what kind of Traps you can get from it. You might think it's strange that a manufacturer would add a component to a device and not describe it in the MIB. But the fact is, a lot of devices have sketchy MIBs that don't fully support all their functions.

When you're planning your SNMP monitoring, you need to be able to read MIBs so you can have a realistic idea of what capabilities you have. When you're evaluating new SNMP equipment, examine its MIB file carefully before you purchase.

How do I look at a MIB?

A MIB file is just ASCII text, so you can view it in any word processor or text editor, such as Microsoft Notepad. Some manufacturers provide precompiled MIBs in binary format, but those aren't readable. You want the raw ASCII version of the MIB file.

Note: MIB files are sometimes provided as Unix text files. Unix text format is significantly different from DOS/Windows text format. DOS/Windows text files have a carriage return and a line feed at the end of each line; Unix files only have a line feed. If you want to view MIB files on a Windows PC, ask your vendor for a DOS-formatted version, or you can use a conversion utility to convert between text formats.

Will I need to edit the MIB?

Generally speaking, no. MIB files aren't really designed to be edited by the end user. Theoretically, you could edit the text descriptions of managed objects to be more user-friendly, but it's better to use your SNMP manager's presentation software to create a useful display.

How do I read the MIB?

To read a MIB file, you have to understand just a little about how the MIB is structured. Don't worry — you don't have to master MIB notation in order to get useful information from the MIB. In this paper we're going to cover just the essentials you need to know to discover the telemetry capabilities of SNMP devices.

What does a MIB look like?

For an example, here are the first few lines of the standard DPS Telecom MIB file:

```
DPS-MIB-V38 DEFINITIONS ::= BEGIN
IMPORTS
      DisplayString
            FROM RFC1213-MIB
      OBJECT-TYPE
            FROM RFC-1212
      enterprises
            FROM RFC1155-SMI;
```

```
dpsInc OBJECT IDENTIFIER ::= {enterprises  2682}
dpsAlarmControl OBJECT IDENTIFIER ::= {dpsInc 1}
tmonXM OBJECT IDENTIFIER ::= {dpsAlarmControl 1}
tmonIdent OBJECT IDENTIFIER ::= {tmonXM 1}
tmonIdentManufacturer OBJECT-TYPE
      SYNTAX  DisplayString
      ACCESS  read-only
      STATUS  mandatory
      DESCRIPTION "The TMON/XM Unit manufacturer."
      ::= {tmonIdent  1}
tmonIdentModel OBJECT-TYPE
      SYNTAX  DisplayString
      ACCESS  read-only
      STATUS  mandatory
      DESCRIPTION "The TMON/XM model designation."
```

Wow! What language is that?

The MIB is written in ASN.1 notation. (The initials stand for Abstract Syntax Notation 1.) ASN.1 is a standard notation maintained by the ISO (International Organization for Standardization) and used in everything from the World Wide Web to aviation control systems.

A full description of ASN.1 is completely beyond the scope of this white paper — standard references to ASN.1 run up to 600 pages. For our purposes, there are only a few things to understand about ASN.1:

1. It's human-readable.
2. It's specifically designed for communication between dissimilar computer systems, so it's the same for every machine.
3. It's extensible, so it can be used for describing almost anything.
4. Once a term is defined in ASN.1, it can be used as a building block for making other terms. This is very important for understanding MIB structure — you'll see why later on.

How ASN.1 builds new terms out of existing terms

ASN.1 defines each term as a sequence of components, some of which may be sequences themselves. To give a simplified example, here's how you might describe a letter in ASN.1:

```
Letter ::= SEQUENCE     {
      opening     OCTET STRING,
      body        OCTET STRING,
      closing     OCTET STRING,
      address     AddressType
}
```

Note that while most of the elements in this sequence are defined using a primitive element (the "octet string," which is the equivalent of a byte), the address is simply defined as a

text string, "AddressType." You can do this because AddressType is defined in another sequence, like so:

```
AddressType ::= SEQUENCE      {
      name          OCTET STRING,
      number              INTEGER,
      street              OCTET STRING,
      city          OCTET STRING,
      state         OCTET STRING,
      zipCode       INTEGER
}
```

For a computer parsing the sequence "Letter," AddressType will be read as an instruction to insert the octet string and integer structures listed in the sequence that defines AddressType

What terms are defined in the MIB?

The elements defined in the MIB can be extremely broad (for example, all objects created by private businesses) or they can be extremely specific (like a particular Trap message generated by a specific alarm point on an RTU.)

Each element in the MIB is given an object identifier, or OID. An OID is a number that uniquely identifies an element in the SNMP universe. Each OID is associated with a human-readable text label.

What is the function of an OID?

The OIDs identify the data objects that are the subjects of an SNMP message. When your SNMP device sends a Trap or a GetResponse, it transmits a series of OIDs, paired with their current values.

The location of the OID within the overall SNMP packet is shown in Figure 1.

What does an OID look like?

Here's an example: 1.3.6.1.4.1.2681.1.2.102

OK ... but what does it mean?

The OID is a kind of address. It locates this particular element within the entire SNMP universe. The OID describes a tree structure, as shown in Figure 2, and each number separated by a decimal point represents a branch on that tree.

The first few numbers identify the domain of the organization that issued the OID, followed by numbers that identify objects within the domain. Imagine if your home address started "Universe, Milky Way Galaxy ..." and ended with your house number. In a similar way, each OID begins at the root level of the OID domain and gradually becomes more specific.

Each element of the OID also has a human-readable text designation. From left to right, our sample OID reads:

1 (iso): The International Organization for Standardization,

This RTU Grows with Your Network

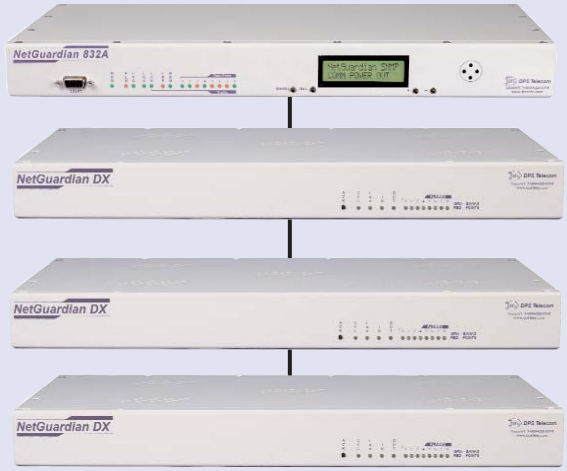
When you're planning your alarm monitoring, think about the future. You don't want to get locked into an alarm system that's inadequate for your future needs — but you don't want to spend too much for alarm capacity you won't immediately use, either.

The NetGuardian 832A remote telemetry unit expands its capacity as your needs change. Install a NetGuardian at your remote site now, and get exactly the right coverage for your current needs.

Then, as your remote site grows, you can extend your alarm monitoring capabilities by adding NetGuardian DX Expansion units. Each NetGuardian DX adds 48 more alarm points, and you can daisy-chain up to three NetGuardian DXs off each NetGuardian 832A base unit.

http://www.dpstelecom.com/ng-832a

Unit	Capacity
Base NG 832	32
1 DX	80
2 DX	128
3 DX	176



NetGuardian DX: Expand your alarm monitoring capacity with NetGuardian DX Expansion Units.

one of the two organizations that assign OID domains.

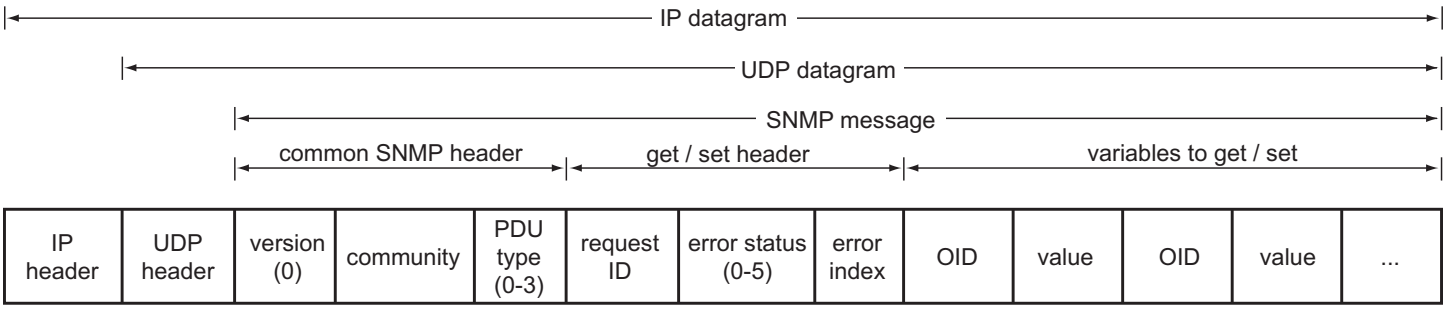


Figure 1. The OID identifies managed objects that can have assigned values

3 (org): An ISO-recognized organization.
6 (dod): U.S. Department of Defense, the agency originally responsible for the Internet.
1 (internet): Internet OID.
4 (private): Private organizations.
1 (enterprises): Business enterprises.
2682 (dpsInc): DPS Telecom.
1 (dpsAlarmControl): DPS alarm and control devices.
2 (dpsRTU): DPS remote telemetry unit.
102 (dpsRTUsumPClr): A Trap generated when all the alarm points on an RTU are clear.

When I look at my MIB files, I don't see long strings of numbers like that

That's because each element of an OID only needs to be defined once. Remember, in ASN.1 notation, once a term is defined, it can be used as a building block to define other terms. The last number of an OID — its most specific element — refers back to the more general elements defined earlier in the MIB.

Here's how the last four elements in our sample OID are defined in the DPS Telecom MIB:

```
dpsInc OBJECT IDENTIFIER ::= {enterprises  2682}
dpsAlarmControl OBJECT IDENTIFIER ::= {dpsInc 1}
dpsRTU OBJECT IDENTIFIER ::= {dpsAlarmControl 2}
dpsRTUsumPClr TRAP-TYPE
      ENTERPRISE  dpsRTU
      VARIABLES { sysDescr,        sysLocation,
dpsRTUDateTime }
      DESCRIPTION  "Generated  when  all  points
clear."
      ::= 102
```

Look at how each term is defined as the term that came immediately before it in the OID, plus one more element. For example, dpsInc is enterprises (1.3.6.1.4) plus one more element, called 2682. The next term, dpsAlarmControl, is dpsInc (1.3.6.1.4.2682), plus one more element, called 1. And so on. Each term in the OID is defined as an extension of earlier terms, going all the way back to the root term, iso.

How to Get Better Visibility of Your SNMP Alarms

There's a big difference between basic alarm monitoring and intelligent alarm management. Any basic system will give you some kind of notification of an alarm. But simple status reports don't provide effective full visibility of your network.

Automated Correction

Your staff can't hover around a screen watching for alarms with their full attention 24/7. A simple system cannot get alarm information to the people who can correct problems quick enough to make a difference. And some problems require immediate action far faster than any human being can respond.

Intelligent Notification

An intelligent alarm management system won't just tell personnel there's a problem; it will locate the problem, provide instructions for corrective action, route alarm information directly to the people who need it, and, if possible, correct the problem automatically. Advanced features like these can make the difference between a minor incident and major downtime.

If you want these features, you need the T/Mon NOC Remote Alarm Monitoring System. T/Mon is a multiprotocol, multifunction alarm master with advanced features like programmable custom alarms, automatic alarm correction, e-mail and pager alarm notification and more.

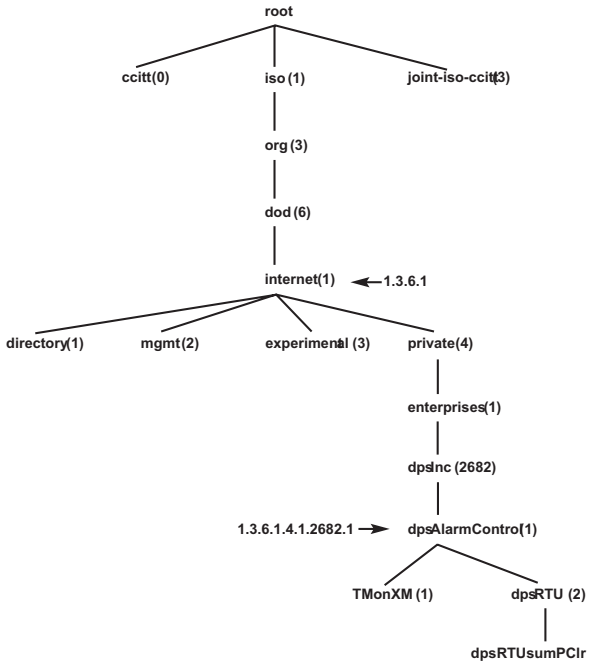


Figure 2. A branch of the MIB object identifier tree

An OID is meaningless unless every element it refers to is specifically called out and identified in the MIB. So when you're compiling your MIB files on your SNMP manager, you need to supply not only the OIDs defined by your equipment vendor, but also OIDs for public entities: iso, org, dod, internet, and so on.

So every MIB file needs to describe the entire OID tree?

Fortunately, no. The upper levels of the OID tree — the parts that define the general OID structure — are defined in a series of standard reference MIB files called RFCs.

(The initials stand for Request for Comment. The RFCs that define SNMP OIDs are part of a larger group of RFC documents that define the Internet as a whole.)

The RFC MIB defines a basic dictionary of terms that vendors use to write their own equipment-specific MIBs. So a vendor-created MIB doesn't have to define the entire OID tree. The vendor's MIB file only has to define the unique OIDs that describe the vendor's equipment.

At the beginning of every MIB file is an IMPORTS line that calls out the terms used in the MIB and the RFC MIB that defines those terms.

Let's take another look at the very beginning of the DPS Telecom MIB:

```
DPS-MIB-V38 DEFINITIONS ::= BEGIN
IMPORTS
      DisplayString
            FROM RFC1213-MIB
```

Alarm Master Choice: T/Mon NOC

T/Mon NOC has many features to make your alarms more meaningful, including:

1. **Detailed, plain English alarm descriptions** include severity, location and date/time stamp.
2. **Immediate notification of COS alarms**, including new alarms and alarms that have cleared
3. **Standing alarm list** is continuously updated.
4. **Text message windows** displaying specific instructions for the appropriate action for an alarm.
5. **Nuisance alarm filtering**, allowing your staff to focus its attention on serious threats.
6. **Pager and email notifications** sent directly to maintenance personnel, even if they're away from the NOC.
7. **Derived alarms and controls** that combine and correlate data from multiple alarm inputs and automatically control remote site equipment to correct complex threats.

For more information, check out T/Mon on the Web at **www.dpstelecom.com/tmon.**

How to avoid the most common cause of compile errors

Your SNMP manager can't compile your MIB files correctly unless it also compiles the RFC MIBs that your MIB files refer to. For example, to compile the DPS Telecom MIB, you need to compile RFC1213-MIB, RFC 1212 and RFC1155-SMI. Compile errors are often caused by missing RFC MIBs.

RFC MIBs are publicly available on the Internet, or your vendor can supply the RFC MIBs you need.

```
OBJECT-TYPE
      FROM RFC-1212
enterprises
      FROM RFC1155-SMI;
```

From this IMPORTS line we can read that the DPS MIB is written using three terms defined in other MIBs — DisplayString, OBJECT-TYPE and enterprises — and these terms are defined in the RFC MIBs listed.

All MIB files are written as extensions of the master RFCs. For this reason, you'll sometimes hear people say that there's only one MIB for all SNMP devices, and that individual MIB files are merely subsections of the unified Management Information Base.

That may be true in theory, but in real life, you only need to worry about the equipment you use, the MIBs that support your equipment, and the RFCs that support those MIBs.

So I'm reading the MIB. What information am I looking for?

You don't need to carefully read over every last line of the MIB file. For your purposes, you're only looking for particular items that will tell you what elements of the device you can monitor and control.

A well-written MIB will be divided into sections. Sections will be identified by comment lines. (In MIB notation, comments lines are identified by two hyphens.) So if you find a line that reads something like:

```
-- TRAP definitions
```

You know you've found what you're looking for.

There are also text labels that identify the MIB objects you're interested in. For example, in SNMP v1 MIBs, Traps are identified by the text label "TRAP-TYPE." If you know the text labels for the kinds of objects you're looking for, you can scan the MIB in a series of Ctrl-F searches.

The MIB objects you need to know

From the perspective of a telemetry manager, what you need to know from the MIB is:

1. What other RFC MIBs you need to support this device
2. What event reports (Traps) the device can send to the SNMP manager
3. What information you can request from the device (the SNMP equivalent of an alarm poll)
4. What characteristics of the device you can control via SNMP

RFC MIBs

The first thing you should look for in the MIB is what RFC MIBs are required to support this device. The necessary RFCs will be called out in the IMPORTS line at the beginning of the MIB.

Traps: Event Reports

For telemetry purposes, the MIB elements you're most interested in are what Traps the device can send. Traps are often described as alarms, but it's better to think of them as event reports.

When a Trap is called out in the MIB, it means that the device is configured to generate a report whenever the element listed changes state. This doesn't mean that the event is necessarily important. Many Traps are merely status messages.

In SNMP v1 MIBs, Traps are always designated with the text label TRAP-TYPE. Here's an example from the MIB for the DPS Telecom NetGuardian RTU::

```
dpsRTUp8005Set TRAP-TYPE
      ENTERPRISE  dpsRTU
      VARIABLES { sysDescr,    sysLocation,
dpsRTUDateTime,
dpsRTUAPort,               dpsRTUCAddress,
dpsRTUADisplay, dpsRTUAPoint,
dpsRTUAPntDesc, dpsRTUAState }
      DESCRIPTION "Generated when discrete
point 5 is set."
      ::=  8005
```

Fortunately, you can ignore a lot of this gobbledygook.

I want to use a device feature that isn't described in the MIB. What can I do?

You can ask the vendor to extend the MIB to include this feature. DPS Telecom has extended its MIB to support client needs. But you need to understand that extending a MIB is actually a software development project. The MIB is not just a text file. It's also a software interface document to the embedded firmware of your SNMP device. Making additions to the MIB requires rewriting the device firmware.

This is a serious project, involving writing code, debugging it, and undergoing a thorough quality assurance process.

Here are the elements that you're interested in:

TRAP-TYPE: This tells you it's a Trap.

DESCRIPTION: This is a human-readable description of the Trap. It should give you a good basic indication of what the Trap signifies.

VARIABLES: This tells you actual information will be included in the Trap. When an actual Trap is sent, each of these variables will be paired with a numerical value that indicates its current state. A variable-and-value pair is called a variable binding.

The variables look pretty cryptic, but it's easy to find out what they mean. Each variable is a text label for an OID defined elsewhere in the MIB. You can do a

Ctrl-F search for any variable term and find its definition. For example, "dpsRTUAPort" is defined in the DPS MIB like this:

```
dpsRTUAPort OBJECT-TYPE
      SYNTAX  INTEGER
      ACCESS  read-only
      STATUS  mandatory
      DESCRIPTION "RTU port number."
      ::= {dpsRTUAlarmEntry 1}
```

Trap variables are your best guide to what alarms you'll get from an SNMP device. Depending on the device, the variables can be highly detailed or they can be vague summary alarms.

Object-Types: Data you can read and sometimes write

When reading the MIB, you'll also want to know what information you can directly request from the device, and what information you can send to the device. These functions are controlled by the SNMP commands GetRequest and SetRequest.

If you want to translate these commands into classic telemetry terms, you can roughly think of a GetRequest as an alarm poll and a SetRequest as a control command.

GetRequests and SetRequests operate on a type of element called an object-type. Object-types are called out in the MIB like this:

```
tmonAState OBJECT-TYPE
      SYNTAX  DisplayString (SIZE (8))
      ACCESS  read-only
      STATUS  mandatory
      DESCRIPTION "The current alarm state."
      ::= {tmonAlarmEntry 4}
```

There are many different kinds of object-types. The specific object-types you might find in a MIB depend on the type of device, what kind of components it has, what the functions of those components, are, etc.

You're probably not going to be interested in every object-type

Quick Primer on SNMP Messages

In SNMP v1, there are only 5 basic PDUs (program datagram units):

GetRequest: a manager-to-agent message requesting the current value of a managed object.

GetNext: a manager-to-agent message requesting the current value of the managed object one number after the one named in the request. (This is a way of walking down a table of values.)

SetRequest: a manager-to-agent message that writes a new value to a managed object

GetResponse: an agent-to-manager message in response to a GetRequest or a SetRequest. In either case, the message reports the current value of the managed object named in the manager's request

Trap: an agent-to-manager message reporting a change in the value of a managed object

Learn about the MIB the Easy Way: Attend DPS Telecom Factory Training

Learn about the MIB and SNMP in-depth and hand-on, in a practical class that will teach you how to get the most from your network monitoring. At a DPS Factory Training Event, you'll learn how to turn SNMP theory into a practical plan for improving your network visibility.

It's the easiest and most complete way to learn SNMP alarm monitoring from the technicians who have designed hundred of successful SNMP monitoring implementations.

For Factory Training Events dates and information call **1-800-693-3314** today or see us on the Web at **www.dpstelecom.com/training**.

listed in the MIB, because you're not going to be interested in everything about the device's functions.

When searching for object-types, it's helpful to start with a plan of what functions of the device you want to manage. What information do you want to retrieve? What controls do you want to set? Knowing the device's functions and how you want to use them will help you narrow down what object-types you should look for in the MIB.

Access

The most important entry in an object-type description is the ACCESS line. This controls whether you can read and write the data described in the object-type.

There are three access settings: not-accessible, read-only and read-write.

Not-accessible means the object-type is there, but you can't request the data in a GetRequest.

Read-only means you can request the data in a GetRequest, but you can't write new data for the object-type in a SetRequest.

Read-write means you're free to retrieve the data in a GetRequest and write new data for the object-type in a SetRequest.

In the example of the alarm state object-type:

```
tmonAState OBJECT-TYPE
      SYNTAX  DisplayString (SIZE (8))
      ACCESS  read-only
      STATUS  mandatory
      DESCRIPTION "The current alarm state."
      ::= {tmonAlarmEntry 4}
```

The access here is read-only, because the alarm state is set by the alarm input on that alarm point.

Here's an example of an object-type with read-write access:

```
dpsRTUDateTime OBJECT-TYPE
      SYNTAX  DisplayString (SIZE (23))
      ACCESS  read-write
      STATUS  mandatory
      DESCRIPTION "The RTU system date and time."
      ::= {dpsRTUIdent  4}
```

Here the access is read-write, because this is a value that you can set from your SNMP manager. You can retrieve the current settings from the RTU's internal clock through a GetRequest. And if the clock needs to be reset, you can write new data in a SetRequest.

DPS Telecom Guarantees You Won't Fail — or Your Money Back

When you're choosing a network monitoring vendor, don't

7 Features That SNMP Managers Can't Match

1. Detailed alarm notifications in plain English that your staff will immediately understand and take action on.
2. Immediate notification of changes of state (COSs), including new alarms and alarms that have cleared.
3. A continuously updated list of all current standing alarms.
4. Text message windows displaying specific instructions for the appropriate action for an alarm.
5. Nuisance alarm filtering that eliminates meaningless status alarms and oscillations allowing your staff to focus its attention on serious threats.
6. Pager and e-mail notifications. Send alarm notifications directly to maintenance personnel, even if they're away from the NOC.
7. Derived alarms and controls that combine and correlate data from multiple alarm inputs and automatically control remote site equipment to correct complex threats.

The T/Mon NOC Remote Alarm Monitoring System provides total visibility of your network status and automatically notifies the right people to keep your network running.

Sign up for a Web demo of T/Mon NOC at
www.dpstelecom.com/webdemo

take chances. Be skeptical. Ask the hard questions. Above all, look for experience. Don't take a sales rep's word that his company can do custom development. Ask how many systems they've worked with, how many protocols they can integrate to SNMP, and check for client testimonials.

DPS Telecom has created hundreds of successful SNMP monitoring implementations for telecoms, utility telecoms, and transportation companies. (Check out **www.dpstelecom.com/case-studies** for some examples.) DPS Telecom monitoring solutions are proven performers under real-world conditions.

You're never taking any risk when you work with DPS Telecom. Your SNMP monitoring solution is backed by a 30-day, no-risk, money-back guarantee. Test your DPS monitoring solution at your site for 30 days. If you're dissatisfied for any reason, just send it back for a full refund.

What to Do Next

Before you make a decision about your SNMP monitoring, there's a lot more you need to know. There's dangers you want to avoid — and there's also opportunities to improve your remote site maintenance that you don't want to miss.

Call or email Rick Dodd at **1-800-622-3314** or **rdodd@dpstele.com** and ask for a free, live Web demonstration of SNMP monitoring solutions with the T/Mon NOC Remote Alarm Monitoring System. There's no obligation to buy — no high-pressure salesmen — just straightforward information to help you make the best decision about your network monitoring. You'll get complete information on hardware, software, specific applications, specifications, features and benefits . . . plus you'll be able to ask questions and get straight answers.

Call Rick at **1-800-622-3314** today to schedule your free Web demo of SNMP monitoring solutions — or

register on the Web at **www.dpstelecom.com/tmon-webdemo**.

The DPS Telecom **White Paper Series** offers a complete library of helpful advice and **survival guides** for every aspect of system monitoring and control.

www.dpstelecom.com/white-papers

Price is Only the First Part of Cost Justification — Make Sure Your Vendor Offers Guaranteed Results

By Bob Berry
Chief Executive Officer
DPS Telecom

In my experience, clients who think hard about cost justification have a more important concern than just price. They want to make sure that they're not spending their company's money on a system that doesn't work as advertised.

That's smart. You have to be careful when working with equipment vendors, especially on protocol mediation projects. Most vendors can't support all your legacy equipment, and they don't have the development capabilities to make integration work.

Some vendors will charge you large NRE (non-refundable engineering) fees up front for custom work, and give no guarantee that the resulting product will meet your performance requirements.

Personally, I think that's a lousy way to do business. I give all my clients a 30-day guarantee: **If my product doesn't completely satisfy you, return it for a full refund.** If I can't give you a solution, I don't want your money. If I'm doing custom work for you, I don't expect you to pay for it until I've proven that it works to your satisfaction.

Very few vendors will make that guarantee. But you need to demand the best level of service from your vendor to ensure that your SNMP alarm monitoring deployment is 100% successful.

Alarm Monitoring Solutions from DPS Telecom

Alarm Monitoring Masters

T/Mon NOC: Full-featured alarm master for up to 1 million alarm points. Features support for 25 protocols, protocol mediation, alarm forwarding, pager and email alarm notification, Web Browser access, multi-user access, standing alarm list, alarm history logging.

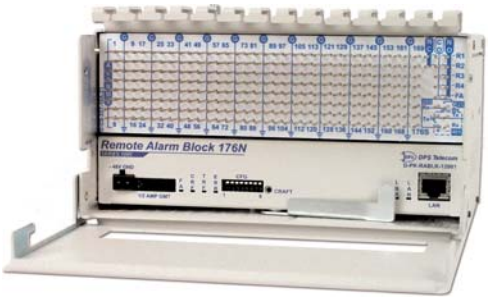
T/Mon SLIM: Light capacity regional alarm master. Supports up to 64 devices and 7,500 alarm points. Features pager and email alarm notification, Web Browser access, standing alarm list and alarm history logging.

NetGuardian 16S: With 16 serial ports, integrated local audiovisual notification, two separate NICs, powerful alarm collection and versatile alarm reporting via SNMP Trap, email and pager, he NetGuardian-16S can handle any alarm monitoring need

Remote Telemetry Units

NetGuardian 832A: RTU monitors 32 alarm points, 8 analog inputs, 8 control relays, 32 ping targets, 8 terminal server ports; reports to any SNMP manager, T/Mon NOC or T/Mon LT

NetGuardian 216: RTU monitors 16 alarm points, 2 analog inputs, 2 control relays, 1 terminal server port; reports to any SNMP manager, T/Mon NOC or T/Mon LT.



Remote Alarm Block 176N: Wire-wrap alarm block monitors 176 alarm points, 4 controls; reports to any SNMP manager, T/Mon NOC or T/Mon LT

NetGuardian 480: RTU monitors 80 alarm points, 4 control relays; reports to any SNMP manager, TL1 master, T/Mon NOC or T/Mon LT

www.dpstelecom.com
1-800-622-3314

DPS Telecom
"Your Partners in Network Alarm Monitoring"

"We protect your network like your business depends on it"