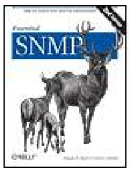


Essential SNMP, 2nd Edition

By Douglas Mauro, Kevin Schmidt

....
Publisher: **O'Reilly**
Pub Date: **September 2005**
ISBN: **0-596-00840-6**
Pages: **460**

Overview

Simple Network Management Protocol (SNMP) provides a "simple" set of operations that allows you to more easily monitor and manage network devices like routers, switches, servers, printers, and more. The information you can monitor with SNMP is wide-ranging--from standard items, like the amount of traffic flowing into an interface, to far more esoteric items, like the air temperature inside a router. In spite of its name, though, SNMP is not especially simple to learn.

O'Reilly has answered the call for help with a practical introduction that shows how to install, configure, and manage SNMP. Written for network and system administrators, the book introduces the basics of SNMP and then offers a technical background on how to use it effectively. *Essential SNMP* explores both commercial and open source packages, and elements like OIDs, MIBs, community strings, and traps are covered in depth. The book contains five new chapters and various updates throughout. Other new topics include:

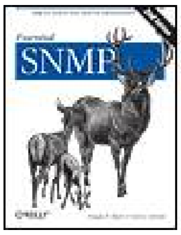
- Expanded coverage of SNMPv1, SNMPv2, and SNMPv3
- Expanded coverage of SNMPc
- The concepts behind network management and change management
- RRDTool and Cricket
- The use of scripts for a variety of tasks
- How Java can be used to create SNMP applications
- Net-SNMP's Perl module

The bulk of the book is devoted to discussing, with real examples, how to use SNMP for system and network administration tasks. Administrators will come away with ideas for writing scripts to help them manage their networks, create managed objects, and extend the operation of SNMP agents.

Once demystified, SNMP is much more accessible. If you're looking for a way to more easily manage your network, look no further than *Essential SNMP, 2nd Edition*.

NEXT

PREV NEXT



Essential SNMP, 2nd Edition

By Douglas Mauro, Kevin Schmidt

....

Publisher: **O'Reilly**

Pub Date: **September 2005**

ISBN: **0-596-00840-6**

Pages: **460**

Table of Contents | Index

PREV NEXT

PREV NEXT

Essential SNMP, Second Edition

by Douglas R. Mauro and Kevin J. Schmidt

Copyright © 2005 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (safari.oreilly.com). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Editors:	Michael Loukides and Debra Cameron
Production Editor:	Darren Kelly
Cover Designer:	Ellie Volckhausen
Interior Designer:	David Futato
Printing History:	
July 2001:	First Edition.
September 2005:	Second Edition.

ISBN: 0-596-00840-6

[M]

PREV NEXT

PREV NEXT

Preface

The Simple Network Management Protocol (SNMP) is an Internet-standard protocol for managing devices on IP networks. Many kinds of devices support SNMP, including routers, switches, servers, workstations, printers, modem racks, and uninterruptible power supplies (UPSs). The ways you can use SNMP range from the mundane to the exotic: it's fairly simple to use SNMP to monitor the health of your routers, servers, and other pieces of network hardware, but you can also use it to control your network devices, page someone, or take other automatic actions if problems arise. The information you can monitor ranges from relatively simple and standardized items, like the amount of traffic flowing into or out of an interface, to more esoteric hardware-and vendor-specific items, like the air temperature inside a router.

Given that there are already a number of books about SNMP in print, why write another one? Although there are many books on SNMP, there's a lack of books aimed at the practicing network or system administrator. Many books cover how to implement SNMP or discuss the protocol at a fairly abstract level, but none really answers the network administrator's most basic questions: how can I best put SNMP to work on my network? How can I make managing my network easier?

We provide a brief overview of the SNMP protocol in Chapters 2 and 3 and then spend a few chapters discussing issues such as hardware requirements and the sorts of tools that are available for use with SNMP. However, the bulk of this book is devoted to discussing, with real examples, how to use SNMP for system and network administration tasks.

Most newcomers to SNMP ask some or all of the following questions:

- What exactly is SNMP?
- How can I, as a system or network administrator, benefit from SNMP?
- What is a MIB?
- What is an OID?
- What is a community string?
- What is a trap?
- I've heard that SNMP is insecure. Is this true?
- Do any of my devices support SNMP? If so, how can I tell if they are configured properly?
- How do I go about gathering SNMP information from a device?
- I have a limited budget for purchasing network management software. What sort of free/open source software is available?
- Is there an SNMP Perl module that I can use to write cool scripts?
- Can I use Java^{TM} to work with SNMP?

This book answers all these questions and more. Our goal is to demystify SNMP and make it more accessible to a wider range of users.

PREV NEXT

Audience for This Book

This book is intended for system and network administrators who could benefit from using SNMP to manage their equipment but who have little or no experience with SNMP or SNMP applications. In our experience, almost any network, no matter how small, can benefit from using SNMP. If you're a Perl programmer, this book will give you some ideas about how to write scripts that use SNMP to help manage your network. If you're not a Perl user, you can use many of the other tools we present, ranging from Net-SNMP (an open source collection of command-line tools) to Hewlett-Packard's OpenView (a high-end, high-priced network management platform).

PREV NEXT

Organization

Chapter 1, Introduction to SNMP and Network Management, provides a nontechnical overview of network management with SNMP. We introduce the different versions of SNMP, managers and agents, network management concepts, and change management techniques.

Chapter 2, SNMPv1 and SNMPv2, discusses the technical details of SNMP versions 1 and 2. We look at the Structure of Management Information (SMI) and the Management Information Base (MIB) and discuss how SNMP actually workshow management information is sent and received over the network.

Chapter 3, SNMPv3, discusses SNMP version 3, which is now a full standard that provides robust security for SNMP.

Chapter 4, NMS Architectures, helps you to think through strategies for deploying SNMP.

Chapter 5, Configuring Your NMS, provides a basic understanding of what to expect when installing NMS software by looking at two NMS packages, HP's OpenView and Castle Rock's SNMPc.

Chapter 6, Configuring SNMP Agents, describes how to configure several SNMP agents for Unix and Windows, including the Net-SNMP agent. To round out the chapter, we discuss how to configure the embedded agents on two network devices: the Cisco SNMP agent and the APC Symetra SNMP agent.

Chapter 7, Polling and Setting, shows how you can use command-line tools and Perl to gather (poll) SNMP information and change (set) the state of a managed device.

Chapter 8, Polling and Thresholds, discusses how to configure OpenView and SNMPc to gather SNMP information via polling. This chapter also discusses RMON configuration on a Cisco router.

Chapter 9, Traps, examines how to send and receive traps using command-line tools, Perl, OpenView, and other management applications.

Chapter 10, Extensible SNMP Agents, shows how several popular SNMP agents can be extended. Extensible agents provide end users with a means to extend the operation of an agent without having access to the agent's source code.

Chapter 11, Adapting SNMP to Fit Your Environment, is geared toward Perl-savvy system administrators. We provide Perl scripts that demonstrate how to perform some common system administration tasks with SNMP.

Chapter 12, MRTG, introduces one of the most widely used open source SNMP applications, the Multi Router Traffic Grapher (MRTG). MRTG provides network administrators with web-based usage graphs of router interfaces and can be configured to graph many other kinds of data.

Chapter 13, RRDtool and Cricket, introduces RRDtool and Cricket. Used together, these tools provide graphing techniques like those in MRTG, but with added flexibility.

Chapter 14, Java and SNMP, discusses how to use Java to build SNMP applications.

Appendix A, Using Input and Output Octets, discusses how to use OpenView to graph input and output octets.

Appendix B, More on OpenView's NNM, discusses how to graph external data with Network Node Manager (NNM), add menu items to NNM, configure user profiles, and use NNM as a centralized communication interface.

Appendix C, Net-SNMP Tools, summarizes the usage of the Net-SNMP command-line tools.

Appendix D, SNMP RFCs, provides an authoritative list of the various RFC numbers that pertain to SNMP.

Appendix E, SNMP Support for Perl, is a good summary of the SNMP Perl module used throughout the book along with an introduction to the Net-SNMP Perl module.

Appendix F, Network Management Software, presents an overview of network management software by category.

Appendix G, Open Source Monitoring Software, introduces some commonly used open source network management and monitoring tools.

Appendix H, Network Troubleshooting Primer, provides a primer on tools that can aid in network troubleshooting.

PREV NEXT

PREV NEXT

What's New in This Edition

This second edition has been thoroughly revised and expanded. It includes the following new features:

- Chapter 1 includes coverage of the concepts behind network management and change management.
- Chapter 2 provides packet traces of the various SNMP operations.
- Chapter 3 provides coverage of SNMPv3. This chapter was an appendix in the first edition; it has been expanded to a full chapter.
- SNMPc coverage has been expanded in Chapters 5 and 9.
- Chapter 11 explains the use of scripts for a variety of tasks. This chapter has doubled in size to include many new scripts. You'll find scripts for service monitoring techniques for SMTP, POP3, HTTP, and DNS, a Perl-based SNMP agent, switch port control, usage of the Cisco Ping MIB, and a section on wireless access point (WAP) monitoring.
- Chapter 13, new in this edition, discusses RRDtool and Cricket.
- Chapter 14, also new in this edition, is devoted to showing how Java can be used to create SNMP applications.
- Appendix E provides a brief overview of Net-SNMP's

Perl module.

- Appendix G provides details on the most commonly used open source tools for network management and monitoring.
- Appendix H introduces the most commonly used network troubleshooting tools.

PREV NEXT

PREV NEXT

Example Programs

All the example programs in this book are available from this book's web page at http://www.oreilly.com/catalog/esnmp2/.

PREV NEXT

PREV NEXT

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact O'Reilly for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Essential SNMP*, Second Edition, by Douglas R. Mauro and Kevin J. Schmidt. Copyright 2005 O'Reilly Media, Inc., 0-596-00840-6."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

PREV NEXT

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Used for object IDs, URLs, filenames, and directory names. It is also used for emphasis and for the first use of technical terms.

`Constant width`

Used for examples, object definitions, literal values, textual conventions, and datatypes. It is also used to show source code, the contents of files, and the output of commands.

`Constant width bold`

Used in interactive examples to show commands or text that would be typed literally by the user. It is also used to emphasize when something, usually in source code or file-contents examples, has been added to or changed from a previous example.

`Constant width italic`

Used for replaceable parameter names in command syntax.

	Indicates a tip, suggestion, or general note.

	Indicates a warning or caution.

Comments and Questions

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (fax)

There is a web page for this book, which lists errata, code examples, reviews, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/esnmp2/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about books, conferences, software, Resource Centers, and the O'Reilly Network, see the O'Reilly web site at:

http://www.oreilly.com

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, it means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top technology books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at http://safari.oreilly.com.

PREV NEXT

Acknowledgments for the Second Edition

Deb Cameron deserves a big thank you for shepherding this second edition from beginning to end. Her diligence and effort helped keep us on track. Dr. Robert Minch, professor at Boise State University, provided valuable suggestions for the second edition. Bobby Krupczak, Ph.D., once again provided feedback on the Concord SystemEDGE agent. Frank Fock was kind enough to provide comments on the Java and SNMP chapter. Max Baker provided the idea for the channel-setting algorithm presented in Chapter 11. Jim Boney graciously volunteered the use of his Cisco routers. Castle Rock Computing was gracious enough to provide us with a copy of SNMPc for the second edition of this book; special thanks go to Castle Rock's John Maytum for coordinating our access to SNMPc.

We are grateful for input from Jason Briggs, Bill Horsfall, and Jason Weiss, who reviewed new material for this second edition under a very tight schedule.

Douglas

For years I worked as a system and network administrator and often faced the question, "How are things running?" This is what led me to SNMP and eventually to the idea for this book. Of course, I would like to thank Kevin for his hard work and dedication. Special thanks go to three special people in my life: my wife, Amy, and our children, Kari and Matthew, for putting up with my long absences while I was writing in the computer room. Thanks also go to my family and friends, who provided support and encouragement.

Kevin

Working on the second edition has been a great joy. The first edition has been out for almost four years, and in this time I have thought about what I wanted to add if someday O'Reilly wanted a second edition written. So, a thank you goes to O'Reilly for giving me the chance to update this book. I would like to thank Douglas for allowing me to once again work on the book with him. Finally, I would like to thank Danette, my loving and generous wife, for allowing me the time I needed to complete this project. Without her support, I would not have made it through the process.

PREV NEXT

PREV NEXT

Acknowledgments for the First Edition

It would be an understatement to say that this book was a long time in the making. It would never have been published without the patience and support of Michael Loukides. Thanks Mike! We would also like to thank the individuals who provided us with valuable technical review feedback and general help and guidance: Mike DeGraw-Bertsch at O'Reilly; Donald Cooley at Global Crossing; Jacob Kirsch at Sun Microsystems, Inc.; Bobby Krupczak, Ph.D., at Concord Communications; John Reinhardt at Road Runner; Patrick Bailey and Rob Sweet at Netrail; and Jürgen Schönwälder at the Technical University of Braunschweig. Rob Romano, a talented graphic artist at O'Reilly, deserves a thank you for making the figures throughout the book look great. Finally, thanks to Jim Sumser, who took the project over in its final stages, and to Rachel Wheeler, the production editor, for putting this book together.

PREV NEXT

PREV NEXT

Chapter 1. Introduction to SNMP and Network Management

In today's complex network of routers, switches, and servers, it can seem like a daunting task to manage all the devices on your network and make sure they're not only up and running but also performing optimally. This is where the Simple Network Management Protocol (SNMP) can help. SNMP was introduced in 1988 to meet the growing need for a standard for managing Internet Protocol (IP) devices. SNMP provides its users with a "simple" set of operations that allows these devices to be managed remotely.

This book is aimed toward system administrators who would like to begin using SNMP to manage their servers or routers, but who lack the knowledge or understanding to do so. We try to give you a basic understanding of what SNMP is and how it works; beyond that, we show you how to put SNMP into practice, using a number of widely available tools. Above all, we want this to be a practical booka book that helps you keep track of what your network is doing.

This chapter introduces SNMP, network management , and change management. Obviously, SNMP is the focus of this book, but having an understanding of general network management concepts will make you better prepared to use SNMP to manage your network.

PREV NEXT

PREV NEXT

1.1. What Is SNMP?

The core of SNMP is a simple set of operations (and the information these operations gather) that gives administrators the ability to change the state of some SNMP-based device. For example, you can use SNMP to shut down an interface on your router or check the speed at which your Ethernet interface is operating. SNMP can even monitor the temperature on your switch and warn you when it is too high.

SNMP usually is associated with managing routers, but it's important to understand that it can be used to manage many types of devices. While SNMP's predecessor, the Simple Gateway Management Protocol (SGMP) was developed to manage Internet routers, SNMP can be used to manage Unix systems, Windows systems, printers, modem racks, power supplies, and more. Any device running software that allows the retrieval of SNMP information can be managed. This includes not only physical devices but also software, such as web servers and databases.

Another aspect of network management is network monitoring ; that is, monitoring an entire network as opposed to individual routers, hosts, and other devices. Remote Network Monitoring (RMON) was developed to help us understand how the network itself is functioning, as well as how individual devices on the network are affecting the network as a whole. It can be used to monitor not only LAN traffic, but WAN interfaces as well. We discuss RMON in more detail later in this chapter and in Chapter 2.

1.1.1. RFCs and SNMP Versions

The Internet Engineering Task Force (IETF) is responsible for defining the standard protocols that govern Internet traffic, including SNMP. The IETF publishes Requests for Comments (RFCs), which are specifications for many protocols that exist in the IP realm. Documents enter the standards track first as *proposed* standards, then move to*draft* status. When a final draft is eventually approved, the RFC is given *standard* statusalthough there are fewer completely approved standards than you might think. Two other standards-track designations, *historical* and *experimental* , define (respectively) a document that has been replaced by a newer RFC and a document that is not yet ready to become a standard. The following list includes all the current SNMP versions and the IETF status of each (see Appendix D for a full list of the SNMP RFCs):

- SNMP Version 1 (SNMPv1) is the initial version of the SNMP protocol. It's defined in RFC 1157 and is a historical IETF standard. SNMPv1's security is based on communities, which are nothing more than passwords: plain-text strings that allow any SNMP-based application that knows the strings to gain access to a device's management information. There are typically three communities in SNMPv1: *read-only*, *read-write* and *trap*. It should be noted that while SNMPv1 is historical, it is still the primary SNMP implementation that many vendors support.

- SNMP version 2 (SNMPv2) is often referred to as community-string-based SNMPv2. This version of SNMP is technically called SNMPv2c, but we will refer to it throughout this book simply as SNMPv2. It's defined in RFC 3416, RFC 3417, and RFC 3418.

- SNMP version 3 (SNMPv3) is the latest version of SNMP. Its main contribution to network management is security. It adds support for strong authentication and private communication between managed entities. In 2002, it finally made the transition from draft standard to full standard. The following RFCs define the standard: RFC 3410, RFC 3411, RFC 3412, RFC 3413, RFC 3414, RFC 3415, RFC 3416, RFC 3417, RFC 3418, and RFC 2576. Chapter 3 provides a thorough treatment of SNMPv3 and Chapter 6 goes through the SNMPv3 agent configuration for Net-SNMP and Cisco. While it is good news that SNMPv3 is a full standard, vendors are notoriously slow at adopting new versions of a protocol. While SNMPv1 has been transitioned to historical, the vast majority of vendor implementations of SNMP are SNMPv1 implementations. Some large infrastructure vendors like Cisco have supported SNMPv3 for quite some time, and we will undoubtedly begin to see more vendors move to SNMPv3 as customers insist on more secure means of managing networks.

The official site for RFCs is http://www.ietf.org/rfc.html. One of the biggest problems with RFCs, however, is finding the one you want. It is a little easier to navigate the RFC index at Ohio State University (http://www.cse.ohio-state.edu/cs/Services/rfc/index.html).

1.1.2. Managers and Agents

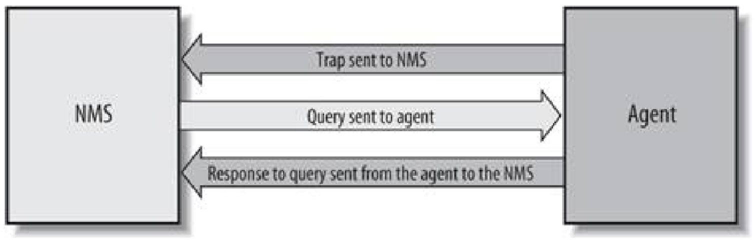
In the previous sections, we've vaguely referred to SNMP-capable devices and network management stations. Now it's time to describe what these two things really are. In the world of SNMP, there are two kind of entities: managers agents . A *manager* is a server running some kind of software system that can handle management tasks for a network. Managers are often referred to as Network Management Stations (NMSs).[*] An NMS is responsible for polling and receiving traps from agents in the network. A *poll*, in the context of network management, is the act of querying an agent (router, switch, Unix server, etc.) for some piece of information. This information can be used later to determine if some sort of catastrophic event has occurred. A *trap* is a way for the agent to tell the NMS that something has happened. Traps are sent asynchronously, not in response to queries from the NMS. The NMS is further responsible for performing an action[†] based upon the information it receives from the agent. For example, when your T1 circuit to the Internet goes down, your router can send a trap to your NMS. In turn, the NMS can take some action, perhaps paging you to let you know that something has happened.

[*] See Appendix F for a listing of some popular NMS applications.

[†] Note that the NMS is preconfigured to perform this action.

The second entity, the *agent*, is a piece of software that runs on the network devices you are managing. It can be a separate program (a daemon, in Unix language), or it can be incorporated into the operating system (for example, Cisco's IOS on a router, or the low-level operating system that controls a UPS). Today, most IP devices come with some kind of SNMP agent built in. The fact that vendors are willing to implement agents in many of their products makes the system administrator's or network manager's job easier. The agent provides management information to the NMS by keeping track of various operational aspects of the device. For example, the agent on a router is able to keep track of the state of each of its interfaces: which ones are up, which ones are down, etc. The NMS can query the status of each interface and take appropriate action if any of them are down. When the agent notices that something bad has happened, it can send a trap to the NMS. This trap originates from the agent and is sent to the NMS, where it is handled appropriately. Some devices will send a corresponding "all clear" trap when there is a transition from a bad state to a good state. This can be useful in determining when a problem situation has been resolved. Figure 1-1 shows the relationship between the NMS and an agent.

Figure 1-1. Relationship between an NMS and an agent



It's important to keep in mind that polls and traps can happen at the same time. There are no restrictions on when the NMS can query the agent or when the agent can send a trap.

1.1.3. The Structure of Management Information and MIBs

The Structure of Management Information (SMI) provides a way to define managed objects and their behavior. An agent has in its possession a list of the objects that it tracks. One such object is the operational status of a router interface (for example, *up*, *down*, or *testing*). This list collectively defines the information the NMS can use to determine the overall health of the device on which the agent resides.

The Management Information Base (MIB) can be thought of as a database of managed objects that the agent tracks. Any sort of status or statistical information that can be accessed by the NMS is defined in a MIB. The SMI provides a way to define managed objects while the MIB is the definition (using the SMI syntax) of the objects themselves. Like a dictionary, which shows how to spell a word and then gives its meaning or definition, a MIB defines a textual name for a managed object and explains its meaning. Chapter 2 goes into more technical detail about MIBs and the SMI.

An agent may implement many MIBs, but all agents implement a particular MIB called MIB-II [*] (RFC 1213). This standard defines variables for things such as interface statistics (interface speeds, MTU, octets[*] sent, octets received, etc.) as well as various other things pertaining to the system itself (system location, system contact, etc.). The main goal of MIB-II is to provide general TCP/IP management information. It doesn't cover every possible item a vendor may want to manage within its particular device.

[*] MIB-I is the original version of this MIB, but it is no longer referred to since MIB-II enhances it.

[*] An octet is an 8-bit quantity, which is the fundamental unit of transfer in TCP/IP networks.

What other kinds of information might be useful to collect? First, many draft and proposed standards have been developed to help manage things such as frame relay, ATM, FDDI, and services (mail, Domain Name System (DNS), etc.). A sampling of these MIBs and their RFC numbers includes:

- ATM MIB (RFC 2515)
- Frame Relay DTE Interface Type MIB (RFC 2115)
- BGP Version 4 MIB (RFC 1657)
- RDBMS MIB (RFC 1697)
- RADIUS Authentication Server MIB (RFC 2619)
- Mail Monitoring MIB (RFC 2789)
- DNS Server MIB (RFC 1611)

But that's far from the entire story, which is why vendors, and individuals, are allowed to define MIB variables for their own use.[†] For example, consider a vendor that is bringing a new router to market. The agent built into the router will respond to NMS requests (or send traps to the NMS) for the variables defined by the MIB-II standard; it probably also implements MIBs for the interface types it provides (e.g., RFC 2515 for ATM and RFC 2115 for Frame Relay). In addition, the router may have some significant new features that are worth monitoring but are not covered by any standard MIB. So, the vendor defines its own MIB (sometimes referred to as a *proprietary MIB*) that implements managed objects for the status and statistical information of its new router.

[†] This topic is discussed further in the next chapter.

Simply loading a new MIB into your NMS does not necessarily allow you to retrieve the data/values/objects, etc., defined within that MIB. You need to load only those MIBs supported by the agents from which you're requesting queries (e.g., snmpget, snmpwalk). Feel free to load additional MIBs for future device support, but don't panic when your device doesn't answer (and possibly returns errors for) these unsupported MIBs.

1.1.4. Host Management

Managing host resources (disk space, memory usage, etc.) is an important part of network management. The distinction between traditional system administration and network management has been disappearing over the last decade and is now all but gone. As Sun Microsystems puts it, "The network is the computer." If your web server or mail server is down, it doesn't matter whether your routers are running correctlyyou're still going to get calls. The Host Resources MIB (RFC 2790) defines a set of objects to help manage critical aspects of Unix and Windows systems.[*]

[*] Any operating system running an SNMP agent can implement Host Resources; it's not confined to agents running on Unix and Windows systems.

Some of the objects supported by the Host Resources MIB include disk capacity, number of system users, number of running processes, and software currently installed. Today, more and more people are relying on service-oriented web sites. Making sure your backend servers are functioning properly is as important as monitoring your routers and other communications devices.

Unfortunately, some agent implementations for these platforms do not implement this MIB since it's not required.

1.1.5. A Brief Introduction to Remote Monitoring (RMON)

Remote Monitoring Version 1 (RMONv1, or RMON) is defined in RFC 2819; an enhanced version of the standard, called RMON Version 2 (RMONv2), is defined in RFC 2021. RMONv1 provides the NMS with packet-level statistics about an entire LAN or WAN. RMONv2 builds on RMONv1 by providing network- and application-level statistics. These statistics can be gathered in several ways. One way is to place an RMON probe on every network segment you want to monitor. Some Cisco routers have limited RMON capabilities built in, so you can use their functionality to perform minor RMON duties. Likewise, some 3Com switches implement the full RMON specification and can be used as full-blown RMON probes.

The RMON MIB was designed to allow an actual RMON probe to run in an offline mode that allows the probe to gather statistics about the network it's watching without requiring an NMS to query it constantly. At some later time, the NMS can query the probe for the statistics it has been gathering. Another feature that most probes implement is the ability to set thresholds for various error conditions and, when a threshold is crossed, alert the NMS with an SNMP trap. You can find a little more technical detail about RMON in the next chapter.

PREV NEXT

1.2. The Concept of Network Management

SNMP is really about network management. Network management is a discipline of its own, but before learning about the details of SNMP in Chapter 2, it's helpful to have an overview of network management itself.

What is network management? Network management is a general concept that employs the use of various tools, techniques, and systems to aid human beings in managing various devices, systems, or networks. Let's take SNMP out of the picture right now and look at a model for network management called *FCAPS*, or Fault Management, Configuration Management, Accounting Management, Performance Management, and Security Management. These conceptual areas were created by the International Organization for Standardization (ISO) to aid in the understanding of the major functions of network management systems. Let's briefly look at each of these now.

1.2.1. Fault Management

The goal of fault management is to detect, log, and notify users of systems or networks of problems. In many environments, downtime of any kind is not acceptable.

Fault management dictates that these steps for fault resolution be followed:

1. Isolate the problem by using tools to determine symptoms.
2. Resolve the problem.
3. Record the process that was used to detect and resolve the problem.

While step 3 is important, it is often not used. Neglecting step 3 has the unwanted effect of causing new engineers to follow steps 1 and 2 in the dark when they could have consulted a database of troubleshooting tips.

1.2.2. Configuration Management

The goal of configuration management is to monitor network and system configuration information so that the effects on network operation of various versions of hardware and software elements can be tracked and managed.

Any system may have a number of interesting and pertinent configuration parameters that engineers may be interested in capturing, including:

- Version of operating system, firmware, etc.
- Number of network interfaces and speeds, etc.
- Number of hard disks
- Number of CPUs
- Amount of RAM

This information generally is stored in a database of some kind. As configuration parameters change for systems, this database is updated. An added benefit to having this data store is that it can aid in problem resolution.

1.2.3. Accounting Management

The goal of accounting management is to ensure that computing and network resources are used fairly by all groups or individuals who access them. Through this form of regulation, network problems can be minimized since resources are divided based on capacities.

1.2.4. Performance Management

The goal of performance management is to measure and report on various aspects of network or system performance.

Let's look at the steps involved in performance management:

1. Performance data is gathered.
2. Baseline levels are established based on analysis of the data gathered.
3. Performance thresholds are established. When these thresholds are exceeded, it is indicative of a problem that requires attention.

One example of performance management is service monitoring. For example, an Internet service provider (ISP) may be interested in monitoring its email service response time. This includes sending emails via SMTP and getting email via POP3. See Chapter 11 for examples of how to do this.

1.2.5. Security Management

The goal of security management is twofold. First, we wish to control access to some resource, such as a network and its hosts. Second, we wish to help detect and prevent attacks that can compromise networks and hosts. Attacks against networks and hosts can lead to denial of service and, even worse, allow hackers to gain access to vital systems that contain accounting, payroll, and source code data.

Security management encompasses not only network security systems but also physical security. Physical security includes card access and video surveillance systems. The goal here is to ensure that only authorized individuals have physical access to vulnerable systems.

Today, network security management is accomplished through the use of various tools and systems designed specifically for this purpose. These include:

- Firewalls
- Intrusion Detection Systems (IDSs)
- Intrusion Prevention Systems (IPSs)
- Antivirus systems
- Policy management and enforcement systems

Most if not all of today's network security systems can integrate with network management systems via SNMP.

PREV NEXT

1.3. Applying the Concepts of Network Management

Being able to apply the concepts of network management is as important as learning how to use SNMP. This section of the chapter provides insights into some of the issues surrounding network management.

1.3.1. Business Case Requirements

The endeavor of network management involves solving a business problem through an implementation of some sort. A business case is developed to understand the impact of implementing some sort of task or function. It looks at how, for example, network administrators do their day-to-day jobs. The basic idea is to reduce costs and increase effectiveness. If the implementation doesn't save a company any money while providing more effective services, there is almost no need to implement a given solution.

1.3.2. Levels of Activity

Before applying management to a specific service or device, you must understand the four possible levels of activity and decide what is appropriate for that service or device:

Inactive

No monitoring is being done, and, if you did receive an alarm in this area, you would ignore it.

Reactive

No monitoring is being done; you react to a problem if it occurs.

Interactive

You monitor components but must interactively troubleshoot them to eliminate side-effect alarms and isolate a root cause.

Proactive

You monitor components, and the system provides a root-cause alarm for the problem at hand and initiates predefined automatic restoral processes where possible to minimize downtime.

1.3.3. Reporting of Trend Analysis

The ability to monitor a service or system proactively begins with trend analysis and reporting . Chapters 12 and 13 describe two tools that are capable of aiding in trend reporting. In general, the goal of trend analysis is to identify when systems, services, or networks are beginning to reach their maximum capacity, with enough lead time to do something about it before it becomes a real problem for end users. For example, you may discover a need to add more memory to your database server or upgrade to a newer version of some application server software that adds a performance boost. Doing so before it becomes a real problem can help your users avoid frustration and possibly keep you employed.

1.3.4. Response Time Reporting

If you are responsible for managing any sort of server (HTTP, SMTP, etc.), you know how frustrating it can be when users come knocking on your door to say that the web server is slow or that surfing the Internet is slow. Response time reporting measures how various aspects of your network (including systems) are performing with respect to responsiveness. Chapter 11 shows how to monitor services with SNMP.

1.3.5. Alarm Correlation

Alarm correlation deals with narrowing down many alerts and events into a single alert or several events that depict the real problem. Another name for this is root-cause analysis. The idea is simple, but it tends to be difficult in practice. For example, when a web server on your network goes down, and you are managing all devices between you and the server (including the switch the server is on and the router), you may get any number of alerts including ones for the server being down, the switch being down, or the router being down, depending on where the real failure is.

Let's say the router is the real issue (for example, an interface card died). You really only need to know that the router is down. Network management systems can often detect when some device or network is unreachable due to varying reasons. The key in this situation is to correlate the server, switch, and router down events into a single high-level event detailing that the router is down. This high-level event can be made up of all the entities and their alarms that are affected by the router being down, but you want to shield an operator from all of these until he is interested in looking at them. The real problem that needs to be addressed is the router's failure. Keeping this storm of alerts and alarms away from the operator helps with overall efficiency and improves the trouble resolution capabilities of the staff.

Clearing alarms is also important. For example, once the router is back up and running, presumably it's going to send an SNMP message that it has come back to life, or maybe a network management system will discover that it's back up and create an alarm to this effect. This notion of state transition, from bad to good, is common. It helps operators know that something is indeed up and operational. It also helps with trending. If you see that a certain device is constantly unreliable, you may want to investigate why.

1.3.6. Trouble Resolution

The key to trouble resolution is knowing that what you are looking at is valuable and can help you resolve the problem. As such, alarms and alerts should aid an operator in resolving the problem. For example, when your router goes down, a cryptic message like "router down" is not helpful. If possible, alerts and alarms should provide the operator with enough detail so that she can effectively troubleshoot and resolve the problem.

PREV NEXT

PREV NEXT

1.4. Change Management

Change management deals with, well, managing change. In other words, you need to plan for both scheduled and emergency changes to your network. Not doing so can cause networks and systems to be unreliable at best and can upset the very people you work for at worst. The following sections provide a high-level overview of change management techniques. The following techniques are recommended by Cisco. See the end of this section for the URL to this paper and others on the topic of network management.

1.4.1. Planning for Change

Change planning is a process that identifies the risk level of a change and builds change planning requirements to ensure that the change is successful. The key steps for change planning are as follows:

- Assign all potential changes a risk level prior to scheduling the change.
- Document at least three risk levels with corresponding change planning requirements. Identify risk levels for software and hardware upgrades, topology changes, routing changes, configuration changes, and new deployments. Assign higher risk levels to nonstandard add, move, or change types of activity.
- The high-risk change process you document needs to include lab validation, vendor review, peer review, and detailed configuration and design documentation.
- Create solution templates for deployments affecting multiple sites. Include information about physical layout, logical design, configuration, software versions, acceptable hardware chassis and modules, and deployment guidelines.
- Document your network standards for configuration, software version, supported hardware, and DNS. Additionally, you may need to document things like device naming conventions, network design details, and

services supported throughout the network.

1.4.2. Managing Change

Change management is a process that approves and schedules the change to ensure the correct level of notification with minimal user impact. The key steps for change management are as follows:

- Assign a change controller who can run change management review meetings, receive and review change requests, manage change process improvements, and act as a liaison for user groups.
- Hold periodic change review meetings with system administration, application development, network operations, and facilities groups as well as general users.
- Document change input requirements, including change owner, business impact, risk level, reason for change, success factors, backout plan, and testing requirements.
- Document change output requirements, including updates to DNS, network map, template, IP addressing, circuit management, and network management.
- Define a change approval process that verifies validation steps for higher-risk change.
- Hold postmortem meetings for unsuccessful changes to determine the root cause of change failure.
- Develop an emergency change procedure that ensures that an optimal solution is maintained or quickly restored.

1.4.3. High-Level Process Flow for Planned Change Management

The steps you'll need to follow during a network change are represented in Figure 1-2.[*] The following sections briefly discuss each box in the flow.

[*] Reprinted by permission from Cisco's "Change Management: Best Practices White Paper," Document ID 22852, http://www.cisco.com/warp/public/126/chmgmt.shtml.

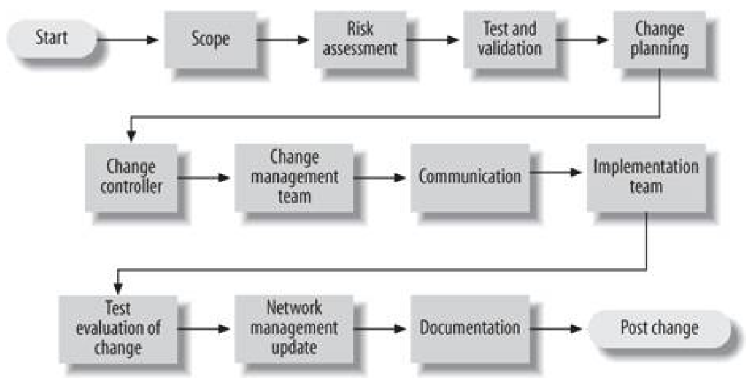
1.4.3.1. Scope

Scope is the who, what, where, and how for the change. In other words, you need to detail every possible impact point for the change, especially its impact on people.

1.4.3.2. Risk assessment

Everything you do to or on a network, when it comes to change, has an associated risk. The person requesting the change needs to establish the risk level for the change. It is best to experiment in a lab setting if you can before you go live with a change. This can help identify problems and aid in risk evaluation.

Figure 1-2. Process flow for planned change management



1.4.3.3. Test and validation

With any proposed change, you want to make sure you have all of your bases covered. Rigorous testing and validation can help with this. Depending upon the associated risk, various levels of validation may need to be performed. For example, if the change has the potential to impact a great many systems, you may wish to test the change in a lab setting. If the change doesn't work, you may also need to document backout procedures.

1.4.3.4. Change planning

For a change to be successful, you must plan for it. This includes requirements gathering, ordering software or hardware, creating documentation, and coordinating human resources.

1.4.3.5. Change controller

Basically, a change controller is a person who is responsible for coordinating all details of the change process.

1.4.3.6. Change management team

You should create a change management team that includes representation from network operations, server operations, application support, and user groups within your organization. The team should review all change requests and approve or deny each request based on completeness, readiness, business impact, business need, and any other conflicts.

The change management team does not investigate the technical accuracy of the change; technical experts who better understand the scope and technical details should complete this phase of the change process.

1.4.3.7. Communication

Many organizations, even small ones, fail to communicate their intentions. Make sure you keep people who may be affected up-to-date on the status of the changes.

1.4.3.8. Implementation team

You should create an implementation team consisting of individuals with the technical expertise to expedite a change. The implementation team should also be involved in the planning phase to contribute to the development of the project checkpoints, testing, backout criteria, and backout time constraints. This team should guarantee adherence to organizational standards, update DNS and network management tools, and maintain and enhance the tool set used to test and validate the change.

1.4.3.9. Test evaluation of change

Once the change has been made, you should begin testing it. Hopefully you already have a set of tests documented that can be used to validate the change. Make sure you allow yourself enough time to perform the tests. If you must back out the change, make sure you test this scenario, too.

1.4.3.10. Network management update

Be sure to update any systems like network management tools, device configurations, network configurations, DNS entries, etc., to reflect the change. This may include removing devices from the management systems that no longer exist, changing the SNMP trap destination your routers use, and so forth.

1.4.3.11. Documentation

Always update documentation that becomes obsolete or incorrect when a change occurs. Documentation may end up being used by a network administrator to solve a problem. If it isn't up-to-date, he cannot be effective in his duties.

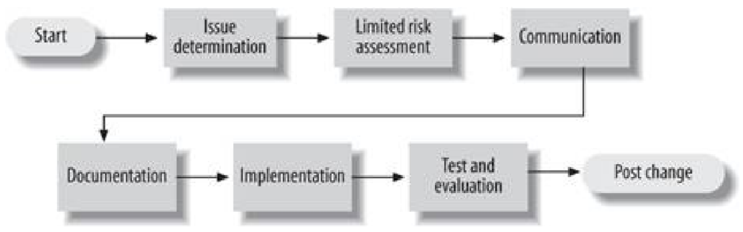
1.4.4. High-Level Process Flow for Emergency Change Management

In the real world, change often comes at 2 a.m. when some critical system is down. But with some effort, your on-the-fly change doesn't have to cause heartburn for you and others in the company. Documentation means a lot more during emergency changes than it does in planned changes. In the heat of the moment, things can get lost or forgotten. Accurately recording the steps and procedures taken will ensure that troubles can be resolved in the future. If you have to, take short notes while the process is unfolding. Later, write it up formally; the important thing is to remember to do it.

Figure 1-3 shows the process flow for emergency changes.[*]

[*] Reprinted by permission from Cisco's "Change Management: Best Practices White Paper," Document ID 22852, http://www.cisco.com/warp/public/126/chmgmt.shtml.

Figure 1-3. Emergency change process



1.4.4.1. Issue determination

Knowing what needs to change is generally not difficult to determine in an emergency. The key is to take one step at a time and not rush things. Yes, time is critical, but rushing can cause mistakes to be made or even bring about a resolution that doesn't fix the real issue. In some cases, the outage can be unnecessarily prolonged.

1.4.4.2. Limited risk assessment

Risk assessment is performed by the network administrator on duty, with advice from other support personnel. Her experience will guide her in how the change is classified from a risk perspective. For example, changing the version of software on a router has much greater impact than changing a device's IP address.

1.4.4.3. Communication and documentation

If at all possible, users should be notified of the change. In an emergency situation, it isn't always possible. Also, be sure to communicate any changes with the change manager. The manager will wish to add to any metrics he keeps on changes. Ensuring that documentation is up-to-date cannot be stressed enough. Having out-of-date documentation means that the staff cannot accurately troubleshoot network and systems problems in the future.

1.4.4.4. Implementation

If the process of assigning risk and documentation occurs prior to the implementation, the actual implementation should be straightforward. Beware of the potential for change coming from multiple support personnel without their knowing about each other's changes. This scenario can lead to increased potential downtime and misinterpretation of the problem.

1.4.4.5. Test and evaluation

Be sure to test the change. Generally, the person who implemented the change also tests and evaluates it. The primary goal is to determine whether the change had the desired effect. If it did not, the emergency change process must be restarted.

1.4.5. Before and After SNMP

Now that you have an idea about what SNMP and network management are, we should look at the before and after pictures for implementing these concepts and technologies. Let's say that you have a network of 100 machines running various operating systems. Several machines are fileservers, a few others are print servers, another is running software that verifies credit card transactions (presumably from a web-based ordering system), and the rest are personal workstations. In addition, various switches and routers help keep the network going. A T1 circuit connects the company to the Internet, and a private connection runs to the credit card verification system.

What happens when one of the fileservers crashes? If it happens in the middle of the workweek, the people using it will notice and the appropriate administrator will be called to fix it. But what if it happens after everyone has gone home, including the administrators, or over the weekend?

What if the private connection to the credit card verification system goes down at 10 p.m. on Friday and isn't restored until Monday morning? If the problem was faulty hardware and it could have been fixed by swapping out a card or replacing a router, thousands of dollars in web site sales could have been lost for no reason. Likewise, if the T1 circuit to the Internet goes down, it could adversely affect the amount of sales generated by individuals accessing your web site and placing orders.

These are obviously serious problemsproblems that can conceivably affect the survival of your business. This is where SNMP comes in. Instead of waiting for someone to notice that something is wrong and locate the person responsible for fixing the problem (which may not happen until Monday morning, if the problem occurs over the weekend), SNMP allows you to monitor your network constantly, even when you're not there. For example, it will notice if the number of bad packets coming through one of your router's interfaces is gradually increasing, suggesting that the router is about to fail. You can arrange to be notified automatically when failure seems imminent so that you can fix the router before it actually breaks. You can also arrange to be notified if the credit card processor appears to get hungyou may even be able to fix it from home. And if nothing goes wrong, you can return to the office on Monday morning knowing there won't be any surprises.

There might not be quite as much glory in fixing problems before they occur, but you and your management will rest more easily. We can't tell you how to translate that into a higher salarysometimes it's better to be the guy who rushes in and fixes things in the middle of a crisis, rather than the guy who makes sure the crisis never occurs. But SNMP does enable you to keep logs that prove your network is running reliably and show when you took action to avert an impending crisis.

1.4.6. Staffing Considerations

Implementing a network management system can mean adding more staff to handle the increased load of maintaining and operating such an environment. At the same time, adding this type of monitoring should, in most cases, reduce the workload of your system administration staff. You will need:

- Staff to maintain the management station. This includes ensuring the management station is configured to properly handle events from SNMP-capable devices.

- Staff to maintain the SNMP-capable devices. This includes making sure that workstations and servers can communicate with the management station.
- Staff to watch and fix the network. This group is usually called a Network Operations Center (NOC) and is staffed 24/7. An alternative to 24/7 staffing is to implement rotating pager duty, where one person is on call at all times, but not necessarily present in the office. Pager duty works only in smaller networked environments in which a network outage can wait for someone to drive into the office and fix the problem.

There is no way to predetermine how many staff members you will need to maintain a management system. The size of the staff will vary depending on the size and complexity of the network you're managing. Some of the larger Internet backbone providers have 70 or more people in their NOCs and others have only one.

PREV NEXT

1.5. Getting More Information

Getting a handle on SNMP may seem like a daunting task. The RFCs provide the official definition of the protocol, but they were written for software developers, not network administrators, so it can be difficult to extract the information you need from them. Fortunately, many online resources are available. A good place to look is the SimpleWeb (http://www.simpleweb.org). SNMP Link (http://www.SNMPLink.org) is another good site for information. *The Simple Times*, an online publication devoted to SNMP and network management, is also useful. You can find all the issues ever published[*] at http://www.simple-times.org. SNMP Research is a commercial SNMP vendor. Aside from selling advanced SNMP solutions, its web site contains a good amount of free information about SNMP. The company's web site is http://www.snmp.com.

[*] At this writing, the current issue is quite old, published in December 2002.

Another great resource is Usenet news. The newsgroup most people frequent is *comp.dcom.net-management*. Another good newsgroup is *comp.protocols.snmp*. Groups such as these promote a community of information sharing, allowing seasoned professionals to interact with individuals who are not as knowledgeable about SNMP or network management. Google has a great interface for searching Usenet news group at http://groups.google.com.

There is an SNMP FAQ, available in two parts at http://www.faqs.org/faqs/snmp-faq/part1/ and http://www.faqs.org/faqs/snmp-faq/part2/.

Cisco has some very good papers on network management, including "Network Management Basics" (http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/nmbasics.htm and "Change Management," from which Figure 1-2 and Figure 1-3 were drawn. Also, Douglas W. Stevenson's article, "Network Management: What It Is and What It Isn't," available at http://www.itmweb.com/essay516.htm, provides important background material for all students of network management.

With that background in mind, Chapter 2 delves much deeper into the details of SNMP.

PREV NEXT

PREV NEXT

Chapter 2. SNMPv1 and SNMPv2

In this chapter, we start to look at SNMP in detail, specifically covering features found in SNMPv1 and SNMPv2 (we'll allude to SNMPv3 occasionally but we describe its features in detail in Chapter 3). By the time you finish this chapter, you should understand how SNMP sends and receives information, what SNMP communities are, and how to read MIB files. We'll also look in more detail at the three MIBs that were introduced in Chapter 1, namely MIB-II, Host Resources, and RMON.

PREV NEXT

PREV NEXT

2.1. SNMP and UDP

SNMP uses the User Datagram Protocol (UDP) as the transport protocol for passing data between managers and agents. UDP, defined in RFC 768, was chosen over the Transmission Control Protocol (TCP) because it is connectionless; that is, no end-to-end connection is made between the agent and the NMS when datagrams (packets) are sent back and forth. This aspect of UDP makes it unreliable since there is no acknowledgment of lost datagrams at the protocol level. It's up to the SNMP application to determine if datagrams are lost and retransmit them if it so desires. This is typically accomplished with a simple timeout. The NMS sends a UDP request to an agent and waits for a response. The length of time the NMS waits depends on how it's configured. If the timeout is reached and the NMS has not heard back from the agent, it assumes the packet was lost and retransmits the request. The number of times the NMS retransmits packets is also configurable.

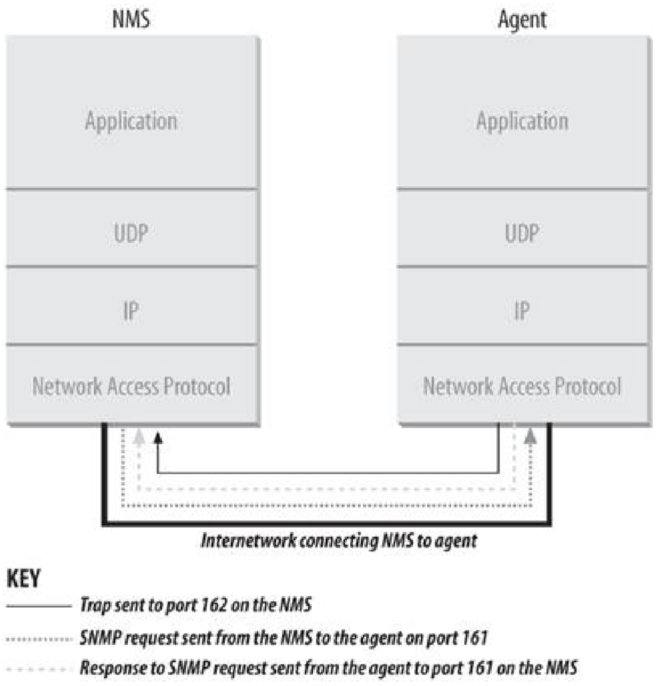
At least as far as regular information requests are concerned, the unreliable nature of UDP isn't a real problem. At worst, the management station issues a request and never receives a response. For traps, the situation is somewhat different. If an agent sends a trap and the trap never arrives, the NMS has no way of knowing that it was ever sent. The agent doesn't even know that it needs to resend the trap because the NMS is not required to send a response back to the agent acknowledging receipt of the trap.

The upside to the unreliable nature of UDP is that it requires low overhead , so the impact on your network's performance is reduced. SNMP has been implemented over TCP, but this is more for special-case situations in which someone is developing an agent for a proprietary piece of equipment. In a heavily congested and managed network, SNMP over TCP is a bad idea. It's also worth realizing that TCP isn't magic and that SNMP is designed for working with networks that are in troubleif your network never failed, you wouldn't need to monitor it. When a network is failing, a protocol that tries to get the data through but gives up if it can't is almost certainly a better design choice than a protocol that floods the network with retransmissions in its attempt to achieve reliability.

SNMP uses UDP port 161 for sending and receiving requests and port 162 for receiving traps from managed devices. Every device that implements SNMP must use these port numbers as the defaults, but some vendors allow you to change the default ports in the agent's configuration. If you change these defaults, the NMS must be made aware of the changes so that it can query the device on the correct ports.

Figure 2-1 shows the TCP/IP protocol suite , which is the basis for all TCP/IP communication. Today, any device that wishes to communicate on the Internet (e.g., Windows systems, Unix servers, Cisco routers, etc.) must use this protocol suite. This model is often referred to as a *protocol stack* since each layer uses the information from the layer directly below it and provides a service to the layer directly above it.

Figure 2-1. TCP/IP communication model and SNMP


NMS
Agent
Application
UDP
IP
Network Access Protocol
Application
UDP
IP
Network Access Protocol
Internetwork connecting NMS to agent
KEY
Trap sent to port 162 on the NMS
SNMP request sent from the NMS to the agent on port 161
Response to SNMP request sent from the agent to port 161 on the NMS

When either an NMS or an agent wishes to perform an SNMP function (e.g., a request or trap), the following events occur in the protocol stack :

Application

First, the actual SNMP application (NMS or agent) decides what it's going to do. For example, it can send an SNMP request to an agent, send a response to an SNMP request (this would be sent from the agent), or send a trap to an NMS. The application layer provides services to an end user, such as an operator requesting status information for a port on an Ethernet switch.

UDP

The next layer, UDP, allows two hosts to communicate with one another. The UDP header contains, among other things, the destination port of the device to which it's sending the request or trap. The destination port will either be 161 (query) or 162 (trap).

IP

The IP layer tries to deliver the SNMP packet to its intended destination, as specified by its IP address.

Media Access Control (MAC)

The final event that must occur for an SNMP packet to reach its destination is for it to be handed off to the physical network, where it can be routed to its final destination. The MAC layer is composed of the actual hardware and device drivers that put your data onto a physical piece of wire, such as an Ethernet card. The MAC layer is also responsible for receiving packets from the physical network and sending them back up the protocol stack so that they can be processed by the application layer (SNMP, in this case).

This interaction between SNMP applications and the network is not unlike that between two pen pals. Both have messages that need to be sent back and forth to one another. Let's say you decide to write your pen pal a letter asking if she would like to visit you over the summer. By deciding to send the invitation, you've acted as the SNMP application. Filling out the envelope with your pen pal's address is equivalent to the function of the UDP layer, which records the packet's destination port in the UDP header; in this case, it's your pen pal's address. Placing a stamp on the envelope and putting it in the mailbox for the mailman to pick up is equivalent to the IP layer's function. The final act occurs when the mailman comes to your house and picks up the letter. From here, the letter will be routed to its final destination, your pen pal's mailbox. The MAC layer of a computer network is equivalent to the mail trucks and airplanes that carry your letter on its way. When your pen pal receives the letter, she will go through the same process to send you a reply.

PREV NEXT

PREV NEXT

2.2. SNMP Communities

SNMPv1 and SNMPv2 use the notion of communities to establish trust between managers and agents. An agent is configured with three community names : read-only, read-write, and trap. The community names are essentially passwords; there's no real difference between a community string and the password you use to access your account on the computer. The three community strings control different kinds of activities. As its name implies, the read-only community string lets you read data values but doesn't let you modify the data. For example, it allows you to read the number of packets that have been transferred through the ports on your router but doesn't let you reset the counters. The read-write community string is allowed to read and modify data values; with the read-write community string, you can read the counters, reset their values, and even reset the interfaces or do other things that change the router's configuration. Finally, the trap community string allows you to receive traps (asynchronous notifications) from the agent.

Most vendors ship their equipment with default community strings , typically *public* for the read-only community string and*private* for the read-write community string. It's important to change these defaults before your device goes live on the network. (You may get tired of hearing this because we say it many times, but it's absolutely essential.) When setting up an SNMP agent, you will want to configure its trap destination, which is the address to which it will send any traps it generates. In addition, since SNMP community strings are sent in clear text, you can configure an agent to send an SNMP authentication-failure trap when someone attempts to query your device with an incorrect community string.

Among other things, authentication-failure traps can be very useful in determining when an intruder might be trying to gain access to your network.

Because community strings are essentially passwords, you should use the same rules for selecting them as you use for Unix or Windows user passwords: no dictionary words, spouse names, etc. An alphanumeric string with mixed upper- and lowercase letters is generally a good idea. As mentioned earlier, the problem with SNMP's authentication is that community strings are sent in plain text, which makes it easy for people to intercept them and use them against you. SNMPv3 addresses this by allowing, among other things, secure authentication and communication between SNMP devices.

There are ways to reduce your risk of attack. IP firewalls or filters minimize the chance that someone can harm any managed device on your network by attacking it through SNMP. You can configure your firewall to allow UDP traffic from only a list of known hosts. For example, you can allow UDP traffic on port 161 (SNMP requests) into your network only if it comes from one of your NMSs. The same goes for traps; you can configure your router so that it allows UDP traffic on port 162 to your NMS only if it originates from one of the hosts you are monitoring. Firewalls aren't 100% effective, but simple precautions such as these do a lot to reduce your risk.

It is important to realize that if someone has read-write access to any of your SNMP devices, he can gain control of those devices by using SNMP (for example, he can set router interfaces, switch ports down, or even modify your routing tables). One way to protect your community strings is to use a Virtual Private Network (VPN) to make sure your network traffic is encrypted. Another way is to change your community strings often. Changing community strings isn't difficult for a small network, but for a network that spans city blocks or more and has dozens (or hundreds or thousands) of managed hosts, changing community strings can be a problem. An easy solution is to write a simple Perl script that uses SNMP to change the community strings on your devices.

PREV NEXT

2.3. The Structure of Management Information

So far, we have used the term *management information* to refer to the operational parameters of SNMP-capable devices. However, we've said very little about what management information actually contains or how it is represented. The first step toward understanding what kind of information a device can provide is to understand how this data is represented within the context of SNMP. The Structure of Management Information Version 1 (SMIv1, RFC 1155) does exactly that: it defines precisely how managed objects [*] are named and specifies their associated datatypes. The Structure of Management Information Version 2 (SMIv2, RFC 2578) provides enhancements for SNMPv2. We'll start by discussing SMIv1, and we will discuss SMIv2 in the next section.[†]

[*] For the remainder of this book, *management information* will be referred to as *managed objects.* Similarly, a single piece of management information (such as the operational status of a router interface) will be known as a *managed object.*

[†] It's worth noting that the version of SMI being used does not relate to the version of SNMP being used.

The definition of managed objects can be broken down into three attributes:

Name

The name, or object identifier (OID), uniquely defines a managed object. Names commonly appear in two forms: numeric and "human readable." In either case, the names are long and inconvenient. In SNMP applications, a lot of work goes into helping you navigate through the namespace conveniently.

Type and syntax

A managed object's datatype is defined using a subset of Abstract Syntax Notation One (ASN.1). ASN.1 is a way of specifying how data is represented and transmitted between managers and agents, within the context of SNMP. The nice thing about ASN.1 is that the notation is machine independent. This means that a PC running Windows 2000 can communicate with a Sun SPARC machine and not have to worry about things such as byte ordering.

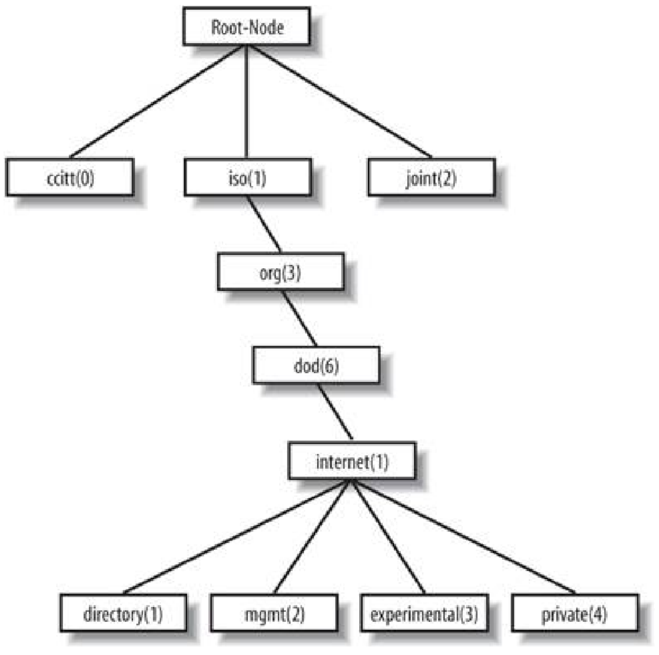
Encoding

A single instance of a managed object is encoded into a string of octets using the Basic Encoding Rules (BER). BER defines how the objects are encoded and decoded so that they can be transmitted over a transport medium such as Ethernet.

2.3.1. Naming OIDs

Managed objects are organized into a treelike hierarchy . This structure is the basis for SNMP's naming scheme. An object ID is made up of a series of integers based on the nodes in the tree, separated by dots (.). Although there's a human-readable form that's friendlier than a string of numbers, this form is nothing more than a series of names separated by dots, each representing a node of the tree. You can use the numbers themselves, or you can use a sequence of names that represent the numbers. Figure 2-2 shows the top few levels of this tree. (We have intentionally left out some branches of the tree that don't concern us here.)

Figure 2-2. SMI object tree



In the object tree, the node at the top of the tree is called the root, anything with children is called a subtree,[*] and anything without children is called a leaf node. For example, Figure 2-2's root, the starting point for the tree, is called Root-Node. Its subtree is made up of *ccitt(0)*, *iso(1)*, and *joint(2)*. In this illustration, *iso(1)* is the only node that contains a subtree; the other two nodes are both leaf nodes. *ccitt(0)* and *joint(2)* do not pertain to SNMP, so they will not be discussed in this book.[*]

[*] Note that the term *branch* is sometimes used interchangeably with *subtree*.

[*] The *ccitt* subtree is administered by the International

Telegraph and Telephone Consultative Committee (CCITT) ; the *joint* subtree is administered jointly by the ISO and CCITT. As we said, neither branch has anything to do with SNMP.

For the remainder of this book, we will focus on the *iso(1).org(3).dod(6).internet(1)* subtree, which is represented in OID form as *1.3.6.1* or as *iso.org.dod.internet*. Each managed object has a numerical OID and an associated textual name. The dotted-decimal notation is how a managed object is represented internally within an agent; the textual name, like an IP domain name, saves humans from having to remember long, tedious strings of integers.

The *directory* branch currently is not used. The *management* branch, or *mgmt*, defines a standard set of Internet management objects. The *experimental* branch is reserved for testing and research purposes. Objects under the *private* branch are defined unilaterally, which means that individuals and organizations are responsible for defining the objects under this branch. Here is the definition of the *internet* subtree, as well as all four of its subtrees:

```
internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

The first line declares *internet* as the OID *1.3.6.1*, which is defined (the `::=` is a definition operator) as a subtree of *iso.org.dod*, or *1.3.6*. The last four declarations are similar, but they define the other branches that belong to *internet*. For the *directory* branch, the notation `{ internet 1 }` tells us that it is part of the *internet* subtree and that its OID is *1.3.6.1.1*. The OID for *mgmt* is *1.3.6.1.2*, and so on.

There is currently one branch under the *private* subtree. It's used to give hardware and software vendors the ability to define their own private objects for any type of hardware or software they want managed by SNMP. Its SMI definition is:

```
enterprises   OBJECT IDENTIFIER ::= { private 1 }
```

The Internet Assigned Numbers Authority (IANA) currently manages all the private enterprise number assignments for individuals, institutions, organizations, companies, etc.[†] A list of all the current private enterprise numbers can be obtained from http://www.iana.org/assignments/enterprise-numbers. As an example, Cisco Systems's private enterprise number is 9, so the base OID for its private object space is defined as *iso.org.dod.internet.private.enterprises.cisco*, or *1.3.6.1.4.1.9*. Cisco is free to do as it wishes with this private branch. It's typical for companies such as Cisco that manufacture networking equipment to define their own private enterprise objects. This allows for a richer set of management information than can be gathered from the standard set of managed objects defined under the *mgmt* branch.

[†] The term *private enterprise* will be used throughout this book to refer to the *enterprises* branch.

Companies aren't the only ones who can register their own private enterprise numbers . Anyone can do so, and it's free. The web-based form for registering private enterprise numbers can be found at http://www.isi.edu/cgi-bin/iana/enterprise.pl. After you fill in the form, which asks for information such as your organization's name and contact information, your request should be approved in about a week. Why would you want to register your own number? When you become more conversant in SNMP, you'll find things you want to monitor that aren't covered by any MIB, public or private. With your own enterprise number, you can create your own private MIB that allows you to monitor exactly what you want. You'll need to be somewhat clever in extending your agents so that they can look up the information you want, but it's very doable.

2.3.2. Defining OIDs

The SYNTAX attribute provides for definitions of managed objects through a subset of ASN.1. SMIv1 defines several datatypes that are paramount to the management of networks and network devices. It's important to keep in mind that these datatypes are simply a way to define what kind of information a managed object can hold. The types we'll be discussing are similar to those that you'd find in a computer programming language like C. Table 2-1 lists the supported datatypes for SMIv1.

Table 2-1. SMIv1 datatypes

Datatype	Description
INTEGER	A 32-bit number often used to specify enumerated types within the context of a single managed object. For example, the operational status of a router interface can be *up*, *down*, or *testing*. With enumerated types, 1 would represent up, 2 down, and 3 testing. The value zero (0) must not be used as an enumerated type, according to RFC 1155.
OCTET STRING	A string of zero or more octets (more commonly known as bytes) generally used to represent text strings, but also sometimes used to represent physical addresses.
Counter	A 32-bit number with minimum value 0 and maximum value 2^{32} - 1 (4,294,967,295). When the maximum value is reached, it wraps back to zero and starts over. It's primarily used to track information such as the number of octets sent and received on an interface or the number of errors and discards seen on an interface. A Counter is monotonically increasing, in that its values should never decrease during normal operation. When an agent is rebooted, all Counter values should be set to zero. Deltas are used to determine if anything useful can be said for successive queries of Counter values. A delta is computed by querying a Counter at least twice in a row and taking the difference between the query results over some time interval.
OBJECT IDENTIFIER	A dotted-decimal string that represents a managed object within the object tree. For example, *1.3.6.1.4.1.9* represents Cisco Systems' private enterprise OID.
NULL	Not currently used in SNMP.

Datatype	Description
SEQUENCE	Defines lists that contain zero or more other ASN.1 datatypes.
SEQUENCE OF	Defines a managed object that is made up of a SEQUENCE of ASN.1 types.
IpAddress	Represents a 32-bit IPv4 address. Neither SMIv1 nor SMIv2 discusses 128-bit IPv6 addresses .
NetworkAddress	Same as the IpAddress type, but can represent different network address types.
Gauge	A 32-bit number with minimum value 0 and maximum value 2^{32} - 1 (4,294,967,295). Unlike a Counter, a Gauge can increase and decrease at will, but it can never exceed its maximum value. The interface speed on a router is measured with a Gauge.
TimeTicks	A 32-bit number with minimum value 0 and maximum value 2^{32} - 1 (4,294,967,295). TimeTicks measures time in hundredths of a second. Uptime on a device is measured using this datatype.
Opaque	Allows any other ASN.1 encoding to be stuffed into an OCTET STRING.

The goal of all these object types is to define managed objects. In Chapter 1, we said that a MIB is a logical grouping of managed objects as they pertain to a specific management task, vendor, etc. The MIB can be thought of as a specification that defines the managed objects a vendor or device supports. Cisco, for instance, has literally hundreds of MIBs defined for its vast product line. For example, its Catalyst device has a separate MIB from its 7000 series router. Both devices have different characteristics that require different management capabilities. Vendor-specific MIBs are typically distributed as human-readable text files that can be inspected (or even modified) with a standard text editor such as *vi*.

> Most modern NMS products maintain a compact form of all the MIBs that define the set of managed objects for all the different types of devices they're responsible for managing. NMS administrators typically compile a vendor's MIB into a format the NMS can use. Once a MIB has been loaded or compiled, administrators can refer to managed objects using either the numeric or human-readable object ID.

It's important to know how to read and understand MIB files . The following example is a stripped-down version of MIB-II (anything preceded by is a comment):

```
RFC1213-MIB DEFINITIONS ::= BEGIN

     IMPORTS
          mgmt, NetworkAddress, IpAddress, Counter, Gauge,
          TimeTicks
               FROM RFC1155-SMI
          OBJECT-TYPE
               FROM RFC 1212;

     mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }

-- groups in MIB-II

     system       OBJECT IDENTIFIER ::= { mib-2 1 }
     interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
     at           OBJECT IDENTIFIER ::= { mib-2 3 }
```

```
ip          OBJECT IDENTIFIER ::= { mib-2 4 }
icmp        OBJECT IDENTIFIER ::= { mib-2 5 }
tcp         OBJECT IDENTIFIER ::= { mib-2 6 }
udp         OBJECT IDENTIFIER ::= { mib-2 7 }
egp         OBJECT IDENTIFIER ::= { mib-2 8 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp        OBJECT IDENTIFIER ::= { mib-2 11 }

-- the Interfaces table

-- The Interfaces table contains information on the entity's
-- interfaces. Each interface is thought of as being
-- attached to a 'subnetwork.' Note that this term should
-- not be confused with 'subnet,' which refers to an
-- addressing-partitioning scheme used in the Internet
-- suite of protocols.

ifTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "A list of interface entries. The number of entries is
         given by the value of ifNumber."
    ::= { interfaces 2 }

ifEntry OBJECT-TYPE
    SYNTAX  IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "An interface entry containing objects at the subnetwork
         layer and below for a particular interface."
    INDEX   { ifIndex }
    ::= { ifTable 1 }

IfEntry ::=
    SEQUENCE {
        ifIndex
            INTEGER,
        ifDescr
```

```
   DisplayString,
ifType
   INTEGER,
ifMtu
   INTEGER,
ifSpeed
   Gauge,
ifPhysAddress
   PhysAddress,
ifAdminStatus
   INTEGER,
ifOperStatus
   INTEGER,
ifLastChange
   TimeTicks,
ifInOctets
   Counter,
ifInUcastPkts
   Counter,
ifInNUcastPkts
   Counter,
ifInDiscards
   Counter,
ifInErrors
   Counter,
ifInUnknownProtos
   Counter,
ifOutOctets
   Counter,
ifOutUcastPkts
   Counter,
ifOutNUcastPkts
   Counter,
ifOutDiscards
   Counter,
ifOutErrors
   Counter,
ifOutQLen
   Gauge,
ifSpecific
   OBJECT IDENTIFIER
```

```
            }

        ifIndex OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                "A unique value for each interface. Its value ranges
                 between 1 and the value of ifNumber. The value for
                 each interface must remain constant at least from one
                 reinitialization of the entity's network management
                 system to the next reinitialization."

            ::= { ifEntry 1 }

        ifDescr OBJECT-TYPE
            SYNTAX  DisplayString (SIZE (0..255))
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                "A textual string containing information about the
                 interface. This string should include the name of
                 the manufacturer, the product name, and the version
                 of the hardware interface."
            ::= { ifEntry 2 }

END
```

The first line of this file defines the name of the MIBin this case, RFC1213-MIB. (RFC 1213 is the RFC that defines MIB-II; many of the MIBs we refer to are defined by RFCs.) The format of this definition is always the same. The IMPORTS section of the MIB is sometimes referred to as the *linkage* section. It allows you to import datatypes and OIDs from other MIB files using the IMPORTS clause. This MIB imports the following items from RFC1155-SMI (RFC 1155 defines SMIv1, which we discussed earlier in this chapter):

- mgmt
- NetworkAddress

- IpAddress
- Counter
- Gauge
- TimeTicks

It also imports OBJECT-TYPE from RFC 1212, the *Concise MIB Definition* which defines how MIB files are written. Each group of items imported using the IMPORTS clause uses a FROM clause to define the MIB file from which the objects are taken.

The OIDs that will be used throughout the remainder of the MIB follow the linkage section. This group of lines sets up the top level of the *mib-2* subtree. *mib-2* is defined as *mgmt* followed by *.1*. We saw earlier that *mgmt* was equivalent to *1.3.6.1.2*. Therefore, *mib-2* is equivalent to *1.3.6.1.2.1*. Likewise, the *interfaces* group under *mib-2* is defined as { mib-2 2 }, or *1.3.6.1.2.1.2*.

After the OIDs are defined, we get to the actual object definitions. Every object definition has the following format:

```
<name> OBJECT-TYPE
   SYNTAX <datatype>
   ACCESS <either read-only, read-write, write-only, or not-accessible>
   STATUS <either mandatory, optional, or obsolete>
   DESCRIPTION
      "Textual description describing this particular managed object."
   ::= { <Unique OID that defines this object> }
```

The first managed object in our subset of the MIB-II definition is *ifTable*, which represents a table of network interfaces on a managed device (note that object names are defined using mixed case, with the first letter in lowercase). Here is its definition using ASN.1 notation:

```
ifTable
 OBJECT-TYPE
   SYNTAX  SEQUENCE OF IfEntry
   ACCESS  not-accessible
   STATUS  mandatory
   DESCRIPTION
```

```
        "A list of interface entries. The number of entries is given by
         the value of ifNumber."
    ::= { interfaces 2 }
```

The SYNTAX of *ifTable* is SEQUENCE OF IfEntry. This means that *ifTable* is a table containing the columns defined in *IfEntry*. The object is not-accessible which means that there is no way to query an agent for this object's value. Its status is mandatory, which means an agent must implement this object in order to comply with the MIB-II specification. The DESCRIPTION describes exactly what this object is. The unique OID is *1.3.6.1.2.1.2.2* or *iso.org.dod.internet.mgmt.mib-2.interfaces.2*.

Let's now look at the SEQUENCE definition from the MIB file earlier in this section, which is used with the SEQUENCE OF type in the *ifTable* definition:

```
IfEntry ::=
    SEQUENCE {
        ifIndex
            INTEGER,
        ifDescr
            DisplayString,
        ifType
            INTEGER,
        ifMtu
            INTEGER,
        .
        .
        .
        ifSpecific
            OBJECT IDENTIFIER
    }
```

Note that the name of the sequence (*IfEntry*) is mixed case, but the first letter is capitalized, unlike the object definition for *ifTable*. This is how a sequence name is defined. A sequence is simply a list of columnar objects and their SMI datatypes, which defines a conceptual table. In this case, we expect to find variables defined by *ifIndex*, *ifDescr*, *ifType* etc. This table can contain any number of rows; it's up to the agent to manage the rows that reside in the table. It is possible for an NMS to add rows to a table. This operation is covered in "The set Operation," later in this chapter.

Now that we have *IfEntry* to specify what we'll find in any row of the table, we can look back to the definition of *ifEntry* (the actual rows of the table) itself:

```
ifEntry OBJECT-TYPE
    SYNTAX  IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
        "An interface entry containing objects at the subnetwork layer
         and below for a particular interface."
    INDEX   { ifIndex }
    ::= { ifTable 1 }
```

ifEntry defines a particular row in the*ifTable*. Its definition is almost identical to that of *ifTable*, except we have introduced a new clause, INDEX. The index is a unique key used to define a single row in the *ifTable*. It's up to the agent to make sure the index is unique within the context of the table. If a router has six interfaces, the *ifTable* will have six rows in it. *ifEntry*'s OID is *1.3.6.1.2.1.2.2.1*, or *iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry*. The index for *ifEntry* is *ifIndex*, which is defined as:

```
ifIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "A unique value for each interface. Its value ranges between
         1 and the value of ifNumber. The value for each interface
         must remain constant at least from one reinitialization of the
         entity's network management system to the next reinitialization."
       ::= { ifEntry 1 }
```

The *ifIndex* object is read-only, which means we can see its value, but we cannot change it. The final object our MIB defines is *ifDescr*, which is a textual description for the interface represented by that particular row in the *ifTable*. Our MIB example ends with the END clause, which marks the end of the MIB. In the actual MIB-II files, each object listed in the *IfEntry* sequence has its own object definition. In this version of the MIB we list only two of them, in the interest of conserving space.

PREV NEXT

PREV NEXT

2.4. Extensions to the SMI in Version 2

SMIv2 extends the SMI object tree by adding the *snmpV2* branch to the *internet* subtree, adding several new datatypes and making a number of other changes. Figure 2-3 shows how the *snmpV2* objects fit into the bigger picture; the OID for this new branch is *1.3.6.1.6.3.1.1*, or *iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObjects* SMIv2 also defines some new datatypes, which are summarized in Table 2-2.

Figure 2-3. SMIv2 registration tree for SNMPv2

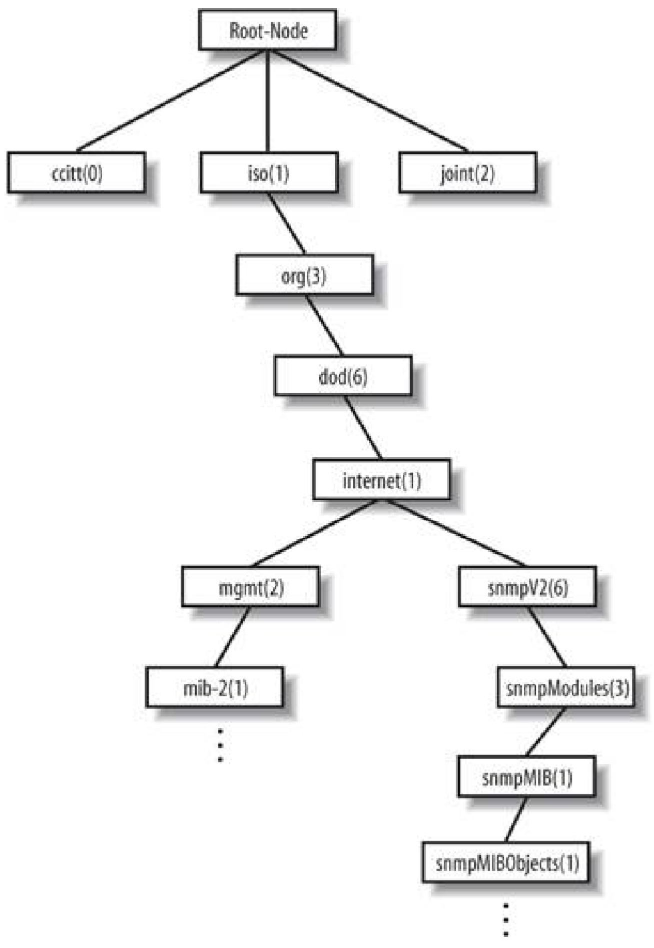


Table 2-2. New datatypes for SMIv2

Datatype	Description
Integer32	Same as an INTEGER.

Datatype	Description
Counter32	Same as a Counter.
Gauge32	Same as a Gauge.
Unsigned32	Represents decimal values in the range of 0 to 2^{32} - 1, inclusive.
Counter64	Similar to Counter32, but its maximum value is 18,446,744,073,709,551,615. Counter64 is ideal for situations in which a Counter32 may wrap back to 0 in a short amount of time.
BITS	An enumeration of nonnegative named bits.

The definition of an object in SMIv2 has changed slightly from SMIv1. There are some new optional fields, giving you more control over how an object is accessed, allowing you to augment a table by adding more columns, and letting you give better descriptions. Here's the syntax of an object definition for SMIv2 . The changed parts are in bold:

```
<name> OBJECT-TYPE
  SYNTAX <datatype>
  UnitsParts <Optional, see below>
  MAX-ACCESS <See below>
  STATUS <See below>
  DESCRIPTION
    "Textual description describing this particular managed object."
  AUGMENTS { <name of table> }
  ::= { <Unique OID that defines this object> }
```

Table 2-3 briefly describes the object definition enhancements made in SMIv2.

Table 2-3. SMIv2 object definition enhancements

Object definition enhancement	Description
UnitsParts	A textual description of the units (i.e., seconds, milliseconds, etc.) used to represent the object.
MAX-ACCESS	An OBJECT-TYPE's ACCESS can be MAX-ACCESS in SNMPv2 . The valid options forMAX-ACCESS are read-only, read-write, read-create, not-accessible, and accessible-for-notify.
STATUS	This clause has been extended to allow the current, obsolete, and deprecated keywords. current in SNMPv2 is the same as mandatory in an SNMPv1 MIB.
AUGMENTS	In some cases, it is useful to add a column to an existing table. The AUGMENTS clause allows you to extend a table by adding one or more columns, represented by some other object. This clause requires the name of the table the object will augment.

SMIv2 defines a new trap type called NOTIFICATION-TYPE, which we will discuss in "SNMP Notification" later in this chapter. SMIv2 also introduces new textual conventions that allow managed objects to be created in more abstract ways. RFC 2579 defines the textual conventions used by SNMPv2, which are listed in Table 2-4.

Table 2-4. Textual conventions for SMIv2

Textual convention	Description
DisplayString	A string of NVT ASCII characters. A DisplayString can be no more than 255 characters in length.
PhysAddress	A media- or physical-level address, represented as an OCTET STRING.
MacAddress	Defines the media-access address for IEEE 802 (the standard for LANs) in canonical[*] order. (In everyday language, this means the Ethernet address.) This address is represented as six octets.
TruthValue	Defines both true and false Boolean values.
TestAndIncr	Used to keep two management stations from modifying the same managed object at the same time.
AutonomousType	An OID used to define a subtree with additional MIB-related definitions.
VariablePointer	A pointer to a particular object instance, such as *ifDescr* for interface *3*. In this case, the VariablePointer would be the OID *ifDescr.3*.
RowPointer	A pointer to a row in a table. For example, *ifIndex.3* points to the third row in *ifTable*.
RowStatus	Used to manage the creation and deletion of rows in a table, since SNMP has no way of doing this via the protocol itself. RowStatus can keep track of

Textual convention	Description
	the state of a row in a table as well as receive commands for creation and deletion of rows. This textual convention is designed to promote table integrity when more than one manager is updating rows. The following enumerated types define the commands and state variables: active(1), notInService(2) notReady(3), createAndGo(4), createAndWait(5), and anddestroy(6).
TimeStamp	Measures the amount of time elapsed between the device's system uptime and some event or occurrence.
TimeInterval	Measures a period of time in hundredths of a second. TimeInterval can take any integer value from 0-2147483647.
DateAndTime	An OCTET STRING used to represent date and time information.
StorageType	Defines the type of memory an agent uses. The possible values are other(1), volatile(2), nonVolatile(3), permanent(4), and readOnly(5).
TDomain	Denotes a kind of transport service.
TAddress	Denotes the transport service address. TAddress is defined to be from 1-255 octets in length.

[*] *Canonical order means that the address should be represented with the least-significant bit first*.

2.5. A Closer Look at MIB-II

MIB-II is a very important management group because every device that supports SNMP must also support MIB-II. Therefore, we will use objects from MIB-II in our examples throughout this book. We won't go into detail about every object in the MIB; we'll simply define the subtrees. The section of *RFC1213-MIB* that defines the base OIDs for the *mib-2* subtree looks like this:

```
mib-2        OBJECT IDENTIFIER ::= { mgmt 1 }
system       OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
at           OBJECT IDENTIFIER ::= { mib-2 3 }
ip           OBJECT IDENTIFIER ::= { mib-2 4 }
icmp         OBJECT IDENTIFIER ::= { mib-2 5 }
tcp          OBJECT IDENTIFIER ::= { mib-2 6 }
udp          OBJECT IDENTIFIER ::= { mib-2 7 }
egp          OBJECT IDENTIFIER ::= { mib-2 8 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp         OBJECT IDENTIFIER ::= { mib-2 11 }
```

mib-2 is defined as *iso.org.dod.internet.mgmt.1*, or *1.3.6.1.2.1*. From here, we can see that the *system* group is `mib-2 1`, or *1.3.6.1.2.1.1*, and so on. Figure 2-4 shows the MIB-II subtree of the *mgmt* branch.

Figure 2-4. MIB-II subtree

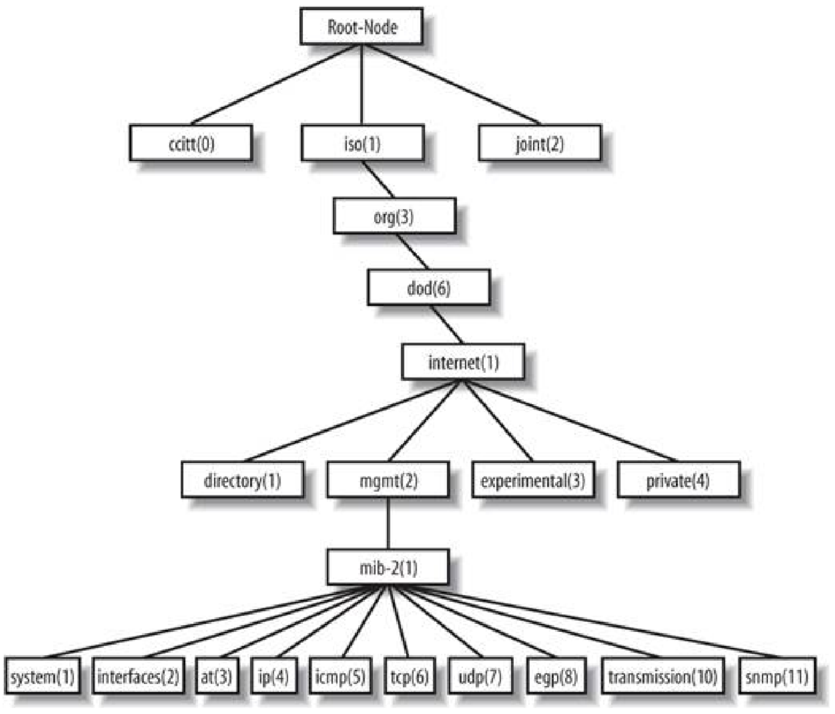


Table 2-5 briefly describes each management group defined in MIB-II. We don't go into great detail about each group since you can pull down RFC 1213 and read the MIB yourself.

Table 2-5. Brief description of the MIB-II groups

Subtree name	OID	Description
system	*1.3.6.1.2.1.1*	Defines a list of objects that pertain to system operation, such as the system uptime, system contact, and system name.
interfaces	*1.3.6.1.2.1.2*	Keeps track of the status of each interface on a managed entity. The *interfaces* group monitors which interfaces are up or down and tracks such things as octets sent and received, errors and discards, etc.
at	*1.3.6.1.2.1.3*	The address translation (*at*) group is deprecated and is provided only for backward compatibility.
ip	*1.3.6.1.2.1.4*	Keeps track of many aspects of IP, including IP routing.
icmp	*1.3.6.1.2.1.5*	Tracks things such as ICMP errors, discards, etc.
tcp	*1.3.6.1.2.1.6*	Tracks, among other things, the state of the TCP connection (e.g., *closed*, *listen*, *synSent*
udp	*1.3.6.1.2.1.7*	Tracks UDP statistics, datagrams in and out, etc.
egp	*1.3.6.1.2.1.8*	Tracks various statistics about the Exterior Gateway Protocol (EGP) and keeps an EGP neighbor table.
transmission	*1.3.6.1.2.1.10*	No objects are currently defined for this group, but other media-specific MIBs are defined using this subtree.
snmp	*1.3.6.1.2.1.11*	Measures the performance of the underlying SNMP implementation on the managed entity and tracks things such as the number of SNMP packets sent and received.

PREV NEXT

2.6. SNMP Operations

We've discussed how SNMP organizes information, but we've left out how we actually go about gathering management information. Now we're going to take a look under the hood to see how SNMP does its thing.

The Protocol Data Unit (PDU) is the message format that managers and agents use to send and receive information. Each of the following SNMP operations has a standard PDU format:

- get
- getnext
- getbulk (SNMPv2 and SNMPv3)
- set
- getresponse
- trap
- notification (SNMPv2 and SNMPv3)
- inform (SNMPv2 and SNMPv3)
- report (SNMPv2 and SNMPv3)

In addition to running actual command-line tools, we will also provide packet dumps of the SNMP operations. For those of you who like looking at packet dumps, this will give you an inside look at what the packet structure is for each command. The packet dumps themselves were taken using the command-line version of Ethereal (http://www.ethereal.com
a look at each operation now. All of the get and set operations were captured with the following command:

```
$ /usr/sbin/tethereal -i lo -x -V -F libpcap -f "port 161"
```

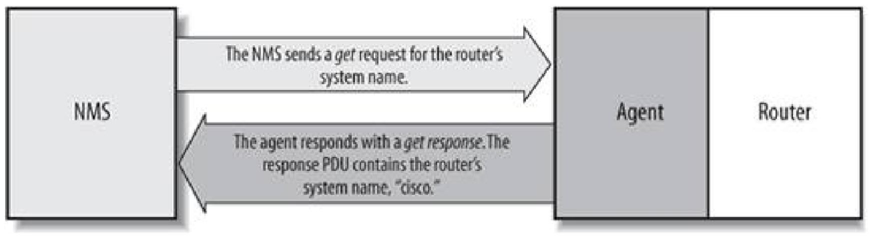
Traps and notifications were captured with this command:

```
$ /usr/sbin/tethereal -i lo -x -V -F libpcap -f "port 162"
```

2.6.1. The get Operation

The get request is initiated by the NMS, which sends the request to the agent. The agent receives the request and processes it to the best of its ability. Some devices that are under heavy load, such as routers, may not be able to respond to the request and will have to drop it. If the agent is successful in gathering the requested information, it sends a getresponse back to the NMS, where it is processed. This process is illustrated in Figure 2-5

Figure 2-5. get request sequence


NMS
The NMS sends a get request for the router's system name.
The agent responds with a get response. The response PDU contains the router's system name, "cisco."
Agent
Router

How did the agent know what the NMS was looking for? One of the items in the get request is a variable binding. A *variable binding*, or *varbind*, is a list of MIB objects that allows a request's recipient to see what the originator wants to know. Variable bindings can be thought of as *OID=value* pairs that make it easy for the originator (the NMS, in this case) to pick out the information it needs when the recipient fills the request and sends back a response. Let's look at this operation in action:

```
$ snmpget cisco.ora.com public .1.3.6.1.2.1.1.6.0
system.sysLocation.0 = ""
```

> All the Unix commands presented in this chapter come from the Net-SNMP agent package (formerly the UCD-SNMP available Unix and Windows agent. You can download the package from http://net-snmp.sourceforge.net. [Appendix C](#) summarizes the commands in this package.

Several things are going on in this example. First, we're running a command on a Unix host. The command is called snmpget. Its main job is to facilitate the gathering of management data using a get request. We've given it three arguments on the command line: the name of the device we would like to query ([cisco.ora.com](#)), the read-only community string (and the OID we would like gathered (*.1.3.6.1.2.1.1.6.0*). If we look back at that *1.3.6.1.2.1.1* is the *system* group, but there are two more integers at the end of the OID: and *.0*. The *.6* is actually the MIB variable that we wish to query; its human-readable name is *sysLocation*. In this case, we would like to see what the system location is set to on the Cisco router. As you can see by the response (`system.sysLocation.0 = ""`), the system location on this router currently is not set to anything. Also note that the response from snmpget is in variable binding format, *OID=value*.

There is one more thing to look at. Why does the MIB variable have a In SNMP, MIB objects are defined by the convention *x.y*, where *x* is the actual OID of the managed object (in our example, *1.3.6.1.2.1.1.6*) and *y* is the instance identifier. For *objects* (that is, objects that aren't defined as a row in a table), *y* is always 0. In the case of a table, the instance identifier lets you select a specific row of the table; 1 is the first row, 2 is the second row, etc. For example, consider the *ifTable* object we looked at earlier in this chapter. When looking up values in *ifTable*, we would use a nonzero instance identifier to select a particular row in the table (in this case, a particular network interface).

> Graphical NMS applications , which include most commercial packages, do not use command-line programs to retrieve management information. We use these commands to give you a feel for how the retrieval commands work and what they typically return. The information a graphical NMS retrieves and its retrieval process are identical to these command-line programs; the NMS just lets you formulate queries and displays the results using a more convenient GUI.

The get command is useful for retrieving a single MIB object at a time. Trying to manage anything in this manner can be a waste of time, though. This is where the getnext command comes in. It allows you to retrieve more than one object from a device, over a period of time.

Now let's look at an SNMP packet as seen with Ethereal's command-line tool tethereal. Given the following command:

```
$ snmpget -v 1 -c public 127.0.0.1 sysContact.0
```

we get the following two datagram traces from tethereal:

```
Frame 1 (85 bytes on wire, 85 bytes captured)
    Arrival Time: Sep 20, 2004 13:46:15.041115000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 85 bytes
    Capture Length: 85 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 71
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7ca4 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: 34066 (34066), Dst Port: snmp (161)
    Source port: 34066 (34066)
    Destination port: snmp (161)
    Length: 51
    Checksum: 0xfe46 (incorrect, should be 0xbbea)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET (0)
    Request Id: 0x20a71b4c
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: NULL

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 47 00 00 40 00 00 11 7c a4 7f 00 00 01 7f 00   .G..@...|.......
0020  00 01 85 12 00 a1 00 33 fe 46 30 29 02 01 00 04   .......3.F0)....
0030  06 70 75 62 6c 69 63 a0 1c 02 04 20 a7 1b 4c 02   .public.... ..L.
0040  01 00 02 01 00 30 0e 30 0c 06 08 2b 06 01 02 01   .....0.0...+....
0050  01 04 00 05 00                                    .....

Frame 2 (144 bytes on wire, 144 bytes captured)
    Arrival Time: Sep 20, 2004 13:46:15.071891000
    Time delta from previous packet: 0.030776000 seconds
    Time since reference or first frame: 0.030776000 seconds
    Frame Number: 2
    Packet Length: 144 bytes
    Capture Length: 144 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 130
    Identification: 0x031d (797)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x794c (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34066 (34066)
    Source port: snmp (161)
    Destination port: 34066 (34066)
    Length: 110
    Checksum: 0xfe81 (incorrect, should be 0xdf61)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x20a71b4c
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: STRING: Root <root@localhost> (configure /etc/snmp/snmp.local.conf)

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 82 03 1d 40 00 00 11 79 4c 7f 00 00 01 7f 00   ....@...yL......
0020  00 01 00 a1 85 12 00 6e fe 81 30 64 02 01 00 04   .......n..0d....
0030  06 70 75 62 6c 69 63 a2 57 02 04 20 a7 1b 4c 02   .public.W.. ..L.
0040  01 00 02 01 00 30 49 30 47 06 08 2b 06 01 02 01   .....0I0G..+....
0050  01 04 00 04 3b 52 6f 6f 74 20 3c 72 6f 6f 74 40   ....;Root <root@
0060  6c 6f 63 61 6c 68 6f 73 74 3e 20 28 63 6f 6e 66   localhost> (conf
0070  69 67 75 72 65 20 2f 65 74 63 2f 73 6e 6d 70 2f   igure /etc/snmp/
0080  73 6e 6d 70 2e 6c 6f 63 61 6c 2e 63 6f 6e 66 29   snmp.local.conf)
```

There are two frames, each labeled appropriately. Frame 1 is initiated by the client. Frame 2 is the agent's response. Ethereal is nice in that it tells us the version of SNMP in use, and the error code (defined later in this chapter in Table 2-6 and Table 2-7). Giving the following command:

```
$ snmpget -v 2c -c public 127.0.0.1 sysContact.0
```

we see the following output from tethereal:

```
Frame 1 (85 bytes on wire, 85 bytes captured)
    Arrival Time: Sep 20, 2004 13:46:26.129733000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 85 bytes
    Capture Length: 85 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 71
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7ca4 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: 34066 (34066), Dst Port: snmp (161)
    Source port: 34066 (34066)
    Destination port: snmp (161)
    Length: 51
    Checksum: 0xfe46 (incorrect, should be 0xbb8f)
```

```
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GET (0)
    Request Id: 0x175f7f93
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: NULL

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 47 00 00 40 00 00 11 7c a4 7f 00 00 01 7f 00   .G..@...|.......
0020  00 01 85 12 00 a1 00 33 fe 46 30 29 02 01 01 04   .......3.F0)....
0030  06 70 75 62 6c 69 63 a0 1c 02 04 17 5f 7f 93 02   .public....._...
0040  01 00 02 01 00 30 0e 30 0c 06 08 2b 06 01 02 01   .....0.0...+....
0050  01 04 00 05 00                                    .....

Frame 2 (144 bytes on wire, 144 bytes captured)
    Arrival Time: Sep 20, 2004 13:46:26.129926000
    Time delta from previous packet: 0.000193000 seconds
    Time since reference or first frame: 0.000193000 seconds
    Frame Number: 2
    Packet Length: 144 bytes
    Capture Length: 144 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 130
    Identification: 0x031e (798)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x794b (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34066 (34066)
    Source port: snmp (161)
    Destination port: 34066 (34066)
    Length: 110
    Checksum: 0xfe81 (incorrect, should be 0xdf06)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x175f7f93
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: STRING: Root <root@localhost> (configure /etc/snmp/snmp.local.conf)

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 82 03 1e 40 00 00 11 79 4b 7f 00 00 01 7f 00   ....@...yK......
0020  00 01 00 a1 85 12 00 6e fe 81 30 64 02 01 01 04   .......n..0d....
0030  06 70 75 62 6c 69 63 a2 57 02 04 17 5f 7f 93 02   .public.W..._...
0040  01 00 02 01 00 30 49 30 47 06 08 2b 06 01 02 01   .....0I0G..+....
0050  01 04 00 04 3b 52 6f 6f 74 20 3c 72 6f 6f 74 40   ....;Root <root@
0060  6c 6f 63 61 6c 68 6f 73 74 3e 20 28 63 6f 6e 66   localhost> (conf
0070  69 67 75 72 65 20 2f 65 74 63 2f 73 6e 6d 70 2f   igure /etc/snmp/
0080  73 6e 6d 70 2e 6c 6f 63 61 6c 2e 63 6f 6e 66 29   snmp.local.conf)
```

The datagram traces look similar to the SNMPv1 traces. Again, we see the version of SNMP in use, namely 2C.

2.6.2. The getnext Operation

The getnext operation lets you issue a sequence of commands to retrieve a group of values from a MIB. In other words, for each MIB object we want to retrieve, a separate getnext request and getresponse are generated. The getnext command traverses a subtree in lexicographic order. Since an OID is a sequence of integers, it's easy for an agent to start at the root of its SMI object tree and work its way down until it finds the OID it is looking for. This form of searching is called depth-first. When the NMS receives a response from the agent for the getnext command it just issued, it issues another getnext command. It keeps doing this until the agent returns an error, signifying that the end of the MIB has been reached and there are no more objects left to get.

If we look at another example, we can see this behavior in action. This time we'll use a command called snmpwalk. This command simply facilitates the getnext procedure for us. It's invoked just like the snmpget command, except this time we specify which branch to start at (in this case, the *system* group):

```
$ snmpwalk cisco.ora.com public system
system.sysDescr.0 = "Cisco IOS Software, C2600 Software (C2600-IPBASE-M),
Version 12.3(8)T3, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2004 by Cisco Systems, Inc.
Compiled Tue 20-Jul-04 17:03 by eaarmas"
system.sysObjectID.0 = OID: enterprises.9.1.19
system.sysUpTime.0 = Timeticks: (27210723) 3 days, 3:35:07.23
system.sysContact.0 = ""
system.sysName.0 = "cisco.ora.com"
system.sysLocation.0 = ""
system.sysServices.0 = 6
```

The getnext sequence returns seven MIB variables. Each object is part of the
as it's defined in RFC 1213. We see a system object ID, the amount of time the system has been up, the contact person, etc.

Given that you've just looked up some object, how does getnext figure out which object to look up next? getnext is based on the concept of the lexicographic ordering of the MIB's object tree. This order is made much simpler because every node in the tree is assigned a number. To understand what this means, let's start at the root of the tree and walk down to the *system* node.

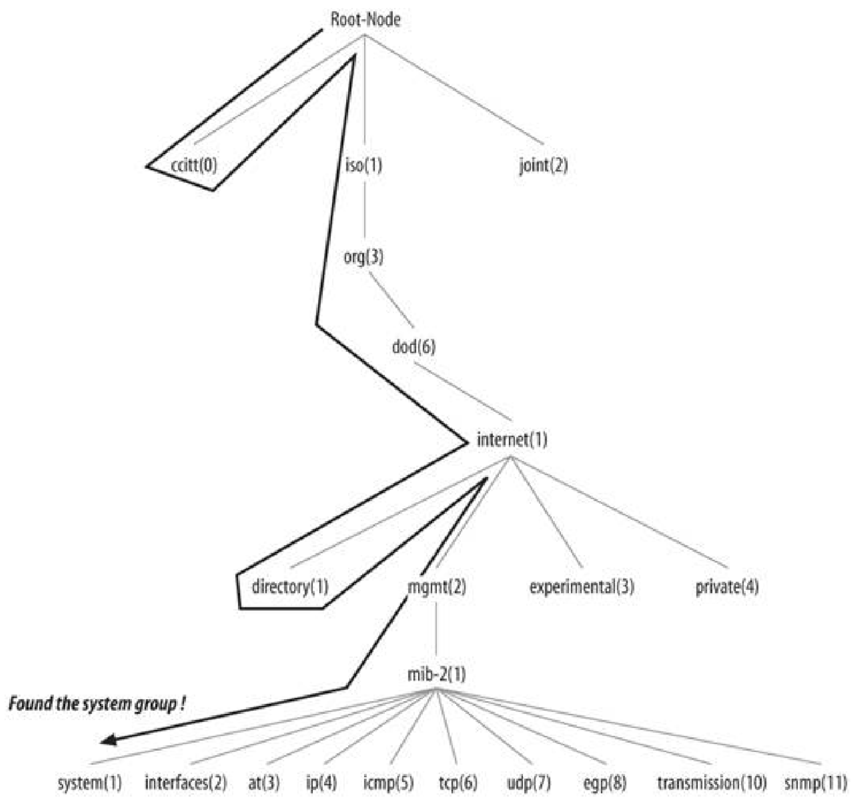
To get to the *system* group (OID *1.3.6.1.2.1.1*), we start at the root of the object tree and work our way down. Figure 2-6 shows the logical progression from the root of the tree all the way to the *system* group. At each node in the tree, we visit the lowest numbered branch. Thus, when we're at the root node, we start by visiting *ccitt*. This node has no nodes underneath it, so we move to the *iso* node. Since *iso* does have a child, we move to that node, process continues until we reach the *system* node. Since each branch is made up of ascending integers (*ccitt(0) iso(1) join(2)*, for example), the agent has no problem traversing this tree structure all the way down to the *system(1)* group. If we were to continue this walk, we'd proceed to *system.1* (*system.sysLocation*), *system.2*, and the other objects in the group. Next, we'd go to *interfaces(2)*, and so on.

Now let's look at what Ethereal sees. Given the following command:

```
$ snmpwalk -v 1 -c public 127.0.0.1 system
```

we get the following output from tethereal:

Figure 2-6. Walking the MIB tree



```
Frame 1 (82 bytes on wire, 82 bytes captured)
    Arrival Time: Sep 20, 2004 13:46:53.598461000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 82 bytes
    Capture Length: 82 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253
(10.0.1.253)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 68
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x62b0 (correct)
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: 34069 (34069), Dst Port: snmp (161)
    Source port: 34069 (34069)
    Destination port: snmp (161)
    Length: 48
    Checksum: 0x183b (incorrect, should be 0xa7f4)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x00ec4809
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1 (SNMPv2-SMI::mib-2)
    Value: NULL

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 44 00 00 40 00 00 11 62 b0 0a 00 01 fd 0a 00   .D..@...b.......
0020  01 fd 85 15 00 a1 00 30 18 3b 30 26 02 01 00 04   .......0.;0&....
0030  06 70 75 62 6c 69 63 a1 19 02 04 00 ec 48 09 02   .public......H..
0040  01 00 02 01 00 30 0b 30 09 06 05 2b 06 01 02 01   .....0.0...+....
0050  05 00                                             ..

Frame 2 (160 bytes on wire, 160 bytes captured)
    Arrival Time: Sep 20, 2004 13:46:53.598662000
    Time delta from previous packet: 0.000201000 seconds
    Time since reference or first frame: 0.000201000 seconds
    Frame Number: 2
    Packet Length: 160 bytes
    Capture Length: 160 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253 (10.
0.1.253)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 146
    Identification: 0x031f (799)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x5f43 (correct)
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34069 (34069)
    Source port: snmp (161)
    Destination port: 34069 (34069)
    Length: 126
    Checksum: 0x1889 (incorrect, should be 0x3f0a)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x00ec4809
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-MIB::sysDescr.0)
    Value: STRING: Linux mailworks.guarded.net 2.4.21-4.EL #1 Fri Oct 3 18:13:58 EDT 2003
```

```
  i686

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   .............E.
0010  00 92 03 1f 40 00 00 11 5f 43 0a 00 01 fd 0a 00   ....@..._C......
0020  01 fd 00 a1 85 15 00 7e 18 89 30 74 02 01 00 04   .......~..0t....
0030  06 70 75 62 6c 69 63 a2 67 02 04 00 ec 48 09 02   .public.g....H..
0040  01 00 02 01 00 30 59 30 57 06 08 2b 06 01 02 01   .....0Y0W..+....
0050  01 01 00 04 4b 4c 69 6e 75 78 20 6d 61 69 6c 77   ....KLinux mailw
0060  6f 72 6b 73 2e 67 75 61 72 64 65 64 2e 6e 65 74   orks.guarded.net
0070  20 32 2e 34 2e 32 31 2d 34 2e 45 4c 20 23 31 20    2.4.21-4.EL #1
0080  46 72 69 20 4f 63 74 20 33 20 31 38 3a 31 33 3a   Fri Oct 3 18:13:
0090  35 38 20 45 44 54 20 32 30 30 33 20 69 36 38 36   58 EDT 2003 i686

Frame 3 (85 bytes on wire, 85 bytes captured)
  Arrival Time: Sep 20, 2004 13:46:53.682655000
  Time delta from previous packet: 0.083993000 seconds
  Time since reference or first frame: 0.084194000 seconds
  Frame Number: 3
  Packet Length: 85 bytes
  Capture Length: 85 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
  Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
  Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
  Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253 (10.0.
1.253)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..0. = ECN-Capable Transport (ECT): 0
    .... ...0 = ECN-CE: 0
  Total Length: 71
  Identification: 0x0001 (1)
  Flags: 0x04
    .1.. = Don't fragment: Set
    ..0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 0
  Protocol: UDP (0x11)
  Header checksum: 0x62ac (correct)
```

```
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: 34069 (34069), Dst Port: snmp (161)
    Source port: 34069 (34069)
    Destination port: snmp (161)
    Length: 51
    Checksum: 0x183e (incorrect, should be 0x9ee5)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x00ec480a
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-MIB::sysDescr.0)
    Value: NULL

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 47 00 01 40 00 00 11 62 ac 0a 00 01 fd 0a 00   .G..@...b.......
0020  01 fd 85 15 00 a1 00 33 18 3e 30 29 02 01 00 04   .......3.>0)....
0030  06 70 75 62 6c 69 63 a1 1c 02 04 00 ec 48 0a 02   .public......H..
0040  01 00 02 01 00 30 0e 30 0c 06 08 2b 06 01 02 01   .....0.0...+....
0050  01 01 00 05 00                                    .....

Frame 4 (95 bytes on wire, 95 bytes captured)
    Arrival Time: Sep 20, 2004 13:46:53.682855000
    Time delta from previous packet: 0.000200000 seconds
    Time since reference or first frame: 0.084394000 seconds
    Frame Number: 4
    Packet Length: 95 bytes
    Capture Length: 95 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253 (10.0.
1.253)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 81
    Identification: 0x0320 (800)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x5f83 (correct)
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34069 (34069)
    Source port: snmp (161)
    Destination port: 34069 (34069)
    Length: 61
    Checksum: 0x1848 (incorrect, should be 0xa08c)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x00ec480a
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.2.0 (SNMPv2-MIB::sysObjectID.0)
    Value: OID: SNMPv2-SMI::enterprises.8072.3.2.10

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 51 03 20 40 00 00 11 5f 83 0a 00 01 fd 0a 00   .Q. @..._.......
0020  01 fd 00 a1 85 15 00 3d 18 48 30 33 02 01 00 04   .......=.H03....
0030  06 70 75 62 6c 69 63 a2 26 02 04 00 ec 48 0a 02   .public.&....H..
0040  01 00 02 01 00 30 18 30 16 06 08 2b 06 01 02 01   .....0.0...+....
0050  01 02 00 06 0a 2b 06 01 04 01 bf 08 03 02 0a      .....+.........
```

To save space, we included only the first four frames, which are the first two getnext operations. As before, frames 1 and 3 are client requests and frames 2 and 4 are the agent's responses. Now let's look at SNMPv2's datagram traces (again we kept it short):

```
Frame 1 (82 bytes on wire, 82 bytes captured)
```

```
    Arrival Time: Sep 20, 2004 13:47:06.413352000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 82 bytes
    Capture Length: 82 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253 (10.0.
1.253)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 68
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x62b0 (correct)
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: 34069 (34069), Dst Port: snmp (161)
    Source port: 34069 (34069)
    Destination port: snmp (161)
    Length: 48
    Checksum: 0x183b (incorrect, should be 0xa1af)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x75cb182f
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1 (SNMPv2-SMI::mib-2)
    Value: NULL

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 44 00 00 40 00 00 11 62 b0 0a 00 01 fd 0a 00   .D..@...b.......
0020  01 fd 85 15 00 a1 00 30 18 3b 30 26 02 01 01 04   .......0.;0&....
0030  06 70 75 62 6c 69 63 a1 19 02 04 75 cb 18 2f 02   .public....u../.
0040  01 00 02 01 00 30 0b 30 09 06 05 2b 06 01 02 01   .....0.0...+....
0050  05 00                                             ..

Frame 2 (160 bytes on wire, 160 bytes captured)
    Arrival Time: Sep 20, 2004 13:47:06.413554000
    Time delta from previous packet: 0.000202000 seconds
    Time since reference or first frame: 0.000202000 seconds
    Frame Number: 2
    Packet Length: 160 bytes
    Capture Length: 160 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253 (10.0.
1.253)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 146
    Identification: 0x0342 (834)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x5f20 (correct)
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34069 (34069)
```

```
    Source port: snmp (161)
    Destination port: 34069 (34069)
    Length: 126
    Checksum: 0x1889 (incorrect, should be 0x38c5)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x75cb182f
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-MIB::sysDescr.0)
    Value: STRING: Linux mailworks.guarded.net 2.4.21-4.EL #1 Fri Oct 3 18:13:58 EDT 2003
    i686

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 92 03 42 40 00 00 11 5f 20 0a 00 01 fd 0a 00   ...B@..._ ......
0020  01 fd 00 a1 85 15 00 7e 18 89 30 74 02 01 01 04   .......~..0t....
0030  06 70 75 62 6c 69 63 a2 67 02 04 75 cb 18 2f 02   .public.g..u../.
0040  01 00 02 01 00 30 59 30 57 06 08 2b 06 01 02 01   .....0Y0W..+....
0050  01 01 00 04 4b 4c 69 6e 75 78 20 6d 61 69 6c 77   ....KLinux mailw
0060  6f 72 6b 73 2e 67 75 61 72 64 65 64 2e 6e 65 74   orks.guarded.net
0070  20 32 2e 34 2e 32 31 2d 34 2e 45 4c 20 23 31 20    2.4.21-4.EL #1
0080  46 72 69 20 4f 63 74 20 33 20 31 38 3a 31 33 3a   Fri Oct 3 18:13:
0090  35 38 20 45 44 54 20 32 30 30 33 20 69 36 38 36   58 EDT 2003 i686

Frame 3 (85 bytes on wire, 85 bytes captured)
    Arrival Time: Sep 20, 2004 13:47:06.495596000
    Time delta from previous packet: 0.082042000 seconds
    Time since reference or first frame: 0.082244000 seconds
    Frame Number: 3
    Packet Length: 85 bytes
    Capture Length: 85 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253 (10.0.
1.253)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 71
    Identification: 0x0001 (1)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x62ac (correct)
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: 34069 (34069), Dst Port: snmp (161)
    Source port: 34069 (34069)
    Destination port: snmp (161)
    Length: 51
    Checksum: 0x183e (incorrect, should be 0x98a0)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GET-NEXT (1)
    Request Id: 0x75cb1830
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-MIB::sysDescr.0)
    Value: NULL

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 47 00 01 40 00 00 11 62 ac 0a 00 01 fd 0a 00   .G..@...b.......
0020  01 fd 85 15 00 a1 00 33 18 3e 30 29 02 01 01 04   .......3.>0)....
0030  06 70 75 62 6c 69 63 a1 1c 02 04 75 cb 18 30 02   .public....u..0.
0040  01 00 02 01 00 30 0e 30 0c 06 08 2b 06 01 02 01   .....0.0...+....
0050  01 01 00 05 00                                    .....

Frame 4 (95 bytes on wire, 95 bytes captured)
    Arrival Time: Sep 20, 2004 13:47:06.495794000
    Time delta from previous packet: 0.000198000 seconds
    Time since reference or first frame: 0.082442000 seconds
    Frame Number: 4
    Packet Length: 95 bytes
    Capture Length: 95 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.0.1.253 (10.0.1.253), Dst Addr: 10.0.1.253 (10.0.
1.253)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 81
    Identification: 0x0343 (835)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x5f60 (correct)
    Source: 10.0.1.253 (10.0.1.253)
    Destination: 10.0.1.253 (10.0.1.253)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34069 (34069)
    Source port: snmp (161)
    Destination port: 34069 (34069)
    Length: 61
    Checksum: 0x1848 (incorrect, should be 0x9a47)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x75cb1830
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.2.0 (SNMPv2
-MIB::sysObjectID.0)
    Value: OID: SNMPv2-SMI::enterprises.8072.3.2.10
```

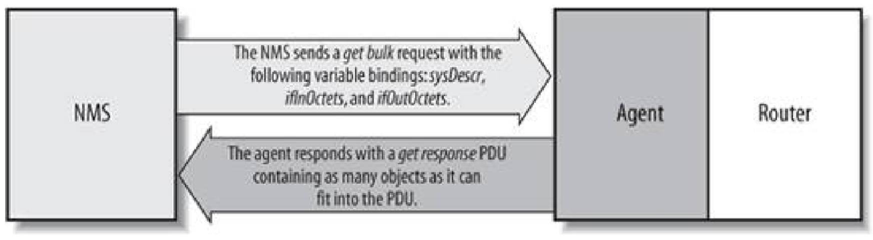
```
0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   .............E.
0010  00 51 03 43 40 00 00 11 5f 60 0a 00 01 fd 0a 00   .Q.C@..._'......
0020  01 fd 00 a1 85 15 00 3d 18 48 30 33 02 01 01 04   .......=.H03....
0030  06 70 75 62 6c 69 63 a2 26 02 04 75 cb 18 30 02   .public.&..u..0.
0040  01 00 02 01 00 30 18 30 16 06 08 2b 06 01 02 01   .....0.0...+....
0050  01 02 00 06 0a 2b 06 01 04 01 bf 08 03 02 0a      .....+.........
```

2.6.3. The getbulk Operation

SNMPv2 defines the getbulk operation, which allows a management application to retrieve a large section of a table at once. The standard get operation can attempt to retrieve more than one MIB object at once, but message sizes are limited by the agent's capabilities. If the agent can't return all the requested responses, it returns an error message with no data. The getbulk operation, on the other hand, tells the agent to send back as much of the response as it can. This means that incomplete responses are possible. Two fields must be set when issuing a getbulk command: nonrepeaters and max-repetitions. Nonrepeaters tell the getbulk command that the first *N* objects can be retrieved with a simple getnext operation. max-repetitions tells the getbulk command to attempt up to *M* getnext operations to retrieve the remaining objects. Figure 2-7 shows the getbulk command sequence.

In Figure 2-7, we're requesting three bindings:*sysDescr*, *ifInOctets*, and
number of variable bindings that we've requested is given by the formula
N is the number of nonrepeaters (i.e., scalar objects in the requestin this case, 1 because *sysDescr* is the only scalar object), *M* is max-repetitions (in this

Figure 2-7. getbulk request sequence


NMS
The NMS sends a get bulk request with the following variable bindings: sysDescr, ifInOctets, and ifOutOctets.
The agent responds with a get response PDU containing as many objects as it can fit into the PDU.
Agent
Router

case, we've set it arbitrarily to 3), and *R* is the number of nonscalar objects in the request (in this case, 2 because *ifInOctets* and *ifOutOctets* are both nonscalar). Plugging in the numbers from this example, we get 1 + (3 * 2) = 7, which is the total number of variable bindings that can be returned by this getbulk request.

The Net-SNMP package comes with a command for issuing getbulk queries
this command using all the parameters previously discussed, it will look like the following:

```
$ snmpbulkget -v2c -c public -Cn1 -Cr3 linux.ora.com sysDescr ifInOctets ifOutOctets
system.sysDescr.0 = " Linux snort 2.4.7-10 #1 Thu Sep 6 17:27:27 EDT 2001
i686 unknown "
interfaces.ifTable.ifEntry.ifInOctets.1 = 70840
interfaces.ifTable.ifEntry.ifOutOctets.1 = 70840
interfaces.ifTable.ifEntry.ifInOctets.2 = 143548020
interfaces.ifTable.ifEntry.ifOutOctets.2 = 111725152
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
```

Since getbulk is an SNMPv2 command, you have to tell snmpbulkget to use an SNMPv2 PDU with the -v2c option. nonrepeaters and max-repetitions are set with the -Cn1 and -Cr3 options. This sets nonrepeaters to 1 and max-repetitions to 3. Notice that the command returned seven variable bindings: one for *sysDescr* and three each for
ifOutOctets.

Now let's look at a trace. If we use the following command:

```
$ ./snmpbulkget -v2c -Cn1 -Cr2 127.0.0.1 -c public sysDescr sysContact
```

we get the following trace:

```
Frame 1 (97 bytes on wire, 97 bytes captured)
    Arrival Time: Sep 20, 2004 20:24:19.106374000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 97 bytes
    Capture Length: 97 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 83
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7c98 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: 34193 (34193), Dst Port: snmp (161)
    Source port: 34193 (34193)
    Destination port: snmp (161)
    Length: 63
    Checksum: 0xfe52 (incorrect, should be 0x0c90)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: GETBULK (5)
    Request Id: 0x0f15c607
    Non-repeaters: 1
    Max repetitions: 2
    Object identifier 1: 1.3.6.1.2.1.1.1 (SNMPv2-MIB::sysDescr)
    Value: NULL
    Object identifier 2: 1.3.6.1.2.1.1.4 (SNMPv2-MIB::sysContact)
    Value: NULL

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 53 00 00 40 00 00 11 7c 98 7f 00 00 01 7f 00   .S..@...|.......
0020  00 01 85 91 00 a1 00 3f fe 52 30 35 02 01 01 04   .......?.R05....
0030  06 70 75 62 6c 69 63 a5 28 02 04 0f 15 c6 07 02   .public.(.......
0040  01 01 02 01 02 30 1a 30 0b 06 07 2b 06 01 02 01   .....0.0...+....
0050  01 01 05 00 30 0b 06 07 2b 06 01 02 01 01 04 05   ....0...+.......
0060  00                                                .

Frame 2 (211 bytes on wire, 211 bytes captured)
    Arrival Time: Sep 20, 2004 20:24:19.151924000
    Time delta from previous packet: 0.045550000 seconds
    Time since reference or first frame: 0.045550000 seconds
    Frame Number: 2
    Packet Length: 211 bytes
    Capture Length: 211 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 197
    Identification: 0x0052 (82)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7bd4 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34193 (34193)
    Source port: snmp (161)
    Destination port: 34193 (34193)
    Length: 177
    Checksum: 0xfec4 (incorrect, should be 0x47bb)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: RESPONSE (2)
    Request Id: 0x0f15c607
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.1.0 (SNMPv2-MIB::sysDescr.0)
    Value: STRING: Linux mailworks.guarded.net 2.4.21-4.EL #1 Fri Oct 3 18:13:58 EDT 2003
    i686
    Object identifier 2: 1.3.6.1.2.1.1.4.0 (SNMPv2-MIB::sysContact.0)
    Value: STRING: "kjs@guarded.net"
    Object identifier 3: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
    Value: STRING: box

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 c5 00 52 40 00 00 11 7b d4 7f 00 00 01 7f 00   ...R@...{.......
0020  00 01 00 a1 85 91 00 b1 fe c4 30 81 a6 02 01 01   ..........0.....
0030  04 06 70 75 62 6c 69 63 a2 81 98 02 04 0f 15 c6   ..public........
0040  07 02 01 00 02 01 00 30 81 89 30 57 06 08 2b 06   .......0..0W..+.
0050  01 02 01 01 01 00 04 4b 4c 69 6e 75 78 20 6d 61   .......KLinux ma
0060  69 6c 77 6f 72 6b 73 2e 67 75 61 72 64 65 64 2e   ilworks.guarded.
0070  6e 65 74 20 32 2e 34 2e 32 31 2d 34 2e 45 4c 20   net 2.4.21-4.EL
0080  23 31 20 46 72 69 20 4f 63 74 20 33 20 31 38 3a   #1 Fri Oct 3 18:
0090  31 33 3a 35 38 20 45 44 54 20 32 30 30 33 20 69   13:58 EDT 2003 i
00a0  36 38 36 30 1d 06 08 2b 06 01 02 01 01 04 00 04   6860...+........
00b0  11 22 6b 6a 73 40 67 75 61 72 64 65 64 2e 6e 65   ."kjs@guarded.ne
00c0  74 22 30 0f 06 08 2b 06 01 02 01 01 05 00 04 03   t"0...+.........
00d0  62 6f 78                                          box
```

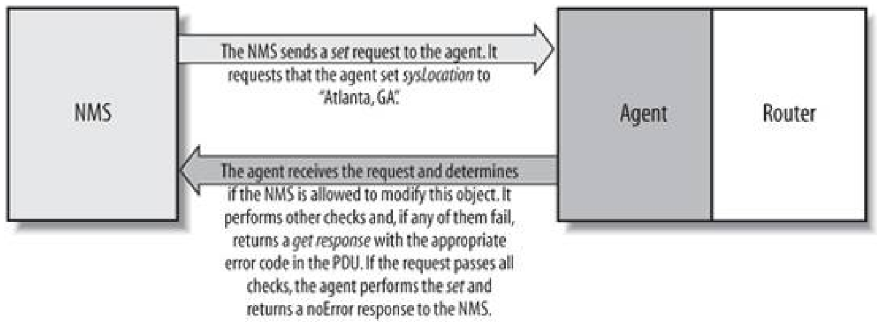
2.6.4. The set Operation

The set command is used to change the value of a managed object or to create a new row in a table. Objects that are defined in the MIB as read-write or read-only can be altered or created using this command. It is possible for an NMS to set more than one object at a time.

Figure 2-8 shows the set request sequence. It's similar to the other commands we've seen so far, but it actually changes something in the device's configuration as opposed to just retrieving a response to a query. Let's look at the set command in action. The following example queries the *sysLocation* variable and sets it to a value:

```
$ snmpget cisco.ora.com public system.sysLocation.0
system.sysLocation.0 = ""
$ snmpset
 cisco.ora.com private system.sysLocation.0 s "Atlanta, GA"
system.sysLocation.0 = "Atlanta, GA"
$ snmpget cisco.ora.com public system.sysLocation.0
system.sysLocation.0 = "Atlanta, GA"
```

Figure 2-8. set request sequence


NMS
The NMS sends a *set* request to the agent. It requests that the agent set *sysLocation* to "Atlanta, GA."
Agent
Router
The agent receives the request and determines if the NMS is allowed to modify this object. It performs other checks and, if any of them fail, returns a *get response* with the appropriate error code in the PDU. If the request passes all checks, the agent performs the *set* and returns a noError response to the NMS.

The first command is the familiar get command, which displays the current value of *sysLocation*. In one of the previous examples, we saw that it was undefined; this is still the case. The second command is snmpset. For this command, we supply the hostname, the read-write community string (*private*), and the variable we want to set (
together with its new value (`s "Atlanta, GA"`). The `s` tells snmpset that we want to set the value of

sysLocation to a string, and `"Atlanta, GA"` is the new value itself. How do we know that requires a string value? The definition of *sysLocation* in RFC 1213 looks like this:

```
sysLocation OBJECT-TYPE
   SYNTAX  DisplayString (SIZE (0..255))
   ACCESS  read-write
   STATUS  mandatory
   DESCRIPTION
      "The physical location of this node (e.g., 'telephone closet,
       3rd floor')."
   ::= { system 6 }
```

The `SYNTAX` for *sysLocation* is `DisplayString (SIZE (0..255))`, which means that it's a string with a maximum length of 255 characters. The snmpset command succeeds and reports the new value of *sysLocation*. But just to confirm, we run a final snmpget, which tells us that the set actually took effect. It is possible to set more than one object at a time, but if any of the sets fail, they all fail (i.e., no values are changed). This behavior is intended.

It's time for more tethereal output. With the following set command:

```
$ snmpset -v 1 -c private 127.0.0.1 sysContact.0s \ snmp@vagrant.org
```

we get the following output:

```
Frame 1 (89 bytes on wire, 89 bytes captured)
   Arrival Time: Sep 20, 2004 14:25:01.895097000
   Time delta from previous packet: 0.000000000 seconds
   Time since reference or first frame: 0.000000000 seconds
   Frame Number: 1
   Packet Length: 89 bytes
   Capture Length: 89 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
   Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
   Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
   Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
```

```
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 75
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7ca0 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: 34102 (34102), Dst Port: snmp (161)
    Source port: 34102 (34102)
    Destination port: snmp (161)
    Length: 55
    Checksum: 0xfe4a (incorrect, should be 0xc029)
Simple Network Management Protocol
    Version: 1 (0)
    Community: private
    PDU type: SET (3)
    Request Id: 0x1df8e7e6
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
    Value: STRING: box

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 4b 00 00 40 00 00 11 7c a0 7f 00 00 01 7f 00   .K..@...|.......
0020  00 01 85 36 00 a1 00 37 fe 4a 30 2d 02 01 00 04   ...6...7.J0-....
0030  07 70 72 69 76 61 74 65 a3 1f 02 04 1d f8 e7 e6   .private........
0040  02 01 00 02 01 00 30 11 30 0f 06 08 2b 06 01 02   ......0.0...+...
0050  01 01 05 00 04 03 62 6f 78                        ......box

Frame 2 (89 bytes on wire, 89 bytes captured)
    Arrival Time: Sep 20, 2004 14:25:01.902787000
    Time delta from previous packet: 0.007690000 seconds
    Time since reference or first frame: 0.007690000 seconds
    Frame Number: 2
    Packet Length: 89 bytes
    Capture Length: 89 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 75
    Identification: 0x0004 (4)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7c9c (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34102 (34102)
    Source port: snmp (161)
    Destination port: 34102 (34102)
    Length: 55
    Checksum: 0xfe4a (incorrect, should be 0xc129)
Simple Network Management Protocol
    Version: 1 (0)
    Community: private
    PDU type: RESPONSE (2)
    Request Id: 0x1df8e7e6
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
    Value: STRING: box

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 4b 00 04 40 00 00 11 7c 9c 7f 00 00 01 7f 00   .K..@...|.......
```

```
0020  00 01 00 a1 85 36 00 37 fe 4a 30 2d 02 01 00 04   .....6.7.J0-....
0030  07 70 72 69 76 61 74 65 a2 1f 02 04 1d f8 e7 e6   .private........
0040  02 01 00 02 01 00 30 11 30 0f 06 08 2b 06 01 02   ......0.0...+...
0050  01 01 05 00 04 03 62 6f 78                        ......box
```

The SNMPv2 set traces are as follows:

```
Frame 1 (89 bytes on wire, 89 bytes captured)
    Arrival Time: Sep 20, 2004 14:25:12.926493000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 89 bytes
    Capture Length: 89 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 75
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7ca0 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: 34102 (34102), Dst Port: snmp (161)
    Source port: 34102 (34102)
    Destination port: snmp (161)
    Length: 55
    Checksum: 0xfe4a (incorrect, should be 0x726b)
Simple Network Management Protocol
    Version: 2C (1)
    Community: private
    PDU type: SET (3)
    Request Id: 0x34df1dbe
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
    Value: STRING: box

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 4b 00 00 40 00 00 11 7c a0 7f 00 00 01 7f 00   .K..@...|.......
0020  00 01 85 36 00 a1 00 37 fe 4a 30 2d 02 01 01 04   ...6...7.J0-....
0030  07 70 72 69 76 61 74 65 a3 1f 02 04 34 df 1d be   .private....4...
0040  02 01 00 02 01 00 30 11 30 0f 06 08 2b 06 01 02   ......0.0...+...
0050  01 01 05 00 04 03 62 6f 78                        ......box

Frame 2 (89 bytes on wire, 89 bytes captured)
    Arrival Time: Sep 20, 2004 14:25:12.989438000
    Time delta from previous packet: 0.062945000 seconds
    Time since reference or first frame: 0.062945000 seconds
    Frame Number: 2
    Packet Length: 89 bytes
    Capture Length: 89 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 75
    Identification: 0x0005 (5)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7c9b (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: snmp (161), Dst Port: 34102 (34102)
    Source port: snmp (161)
    Destination port: 34102 (34102)
    Length: 55
    Checksum: 0xfe4a (incorrect, should be 0x736b)
Simple Network Management Protocol
    Version: 2C (1)
    Community: private
    PDU type: RESPONSE (2)
    Request Id: 0x34df1dbe
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.5.0 (SNMPv2-MIB::sysName.0)
    Value: STRING: box

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 4b 00 05 40 00 00 11 7c 9b 7f 00 00 01 7f 00   .K..@...|.......
0020  00 01 00 a1 85 36 00 37 fe 4a 30 2d 02 01 01 04   .....6.7.J0-....
0030  07 70 72 69 76 61 74 65 a2 1f 02 04 34 df 1d be   .private....4...
0040  02 01 00 02 01 00 30 11 30 0f 06 08 2b 06 01 02   ......0.0...+...
0050  01 01 05 00 04 03 62 6f 78                        ......box
```

2.6.5. get, getnext, getbulk, and set Error Responses

Error responses help you determine whether your get or set request was processed correctly by the agent. The get, getnext, getbulk, and set operations can return the error responses shown in Table 2-6. The error status for each error is shown in parentheses.

Table 2-6. SNMPv1 error messages

SNMPv1 error message	Description
noError(0)	There was no problem performing the request.
tooBig(1)	The response to your request was too big to fit into one response.
noSuchName(2)	An agent was asked to get or set an OID that it can't find; i.e., the OID doesn't exist.
badValue(3)	A read-write or write-only object was set to an inconsistent value.
readOnly(4)	This error is generally not used. The noSuchName this one.
genErr(5)	This is a catchall error. If an error occurs for which none of the previous messages is appropriate, a genErr is issued.

The SNMPv1 error messages are not very robust. In an attempt to fix this problem, SNMPv2 defines additional error responses that are valid for get, set, getnext, and getbulk operations, provided that both the agent and the NMS support SNMPv2. These responses are listed in Table 2-7.

Table 2-7. SNMPv2 error messages

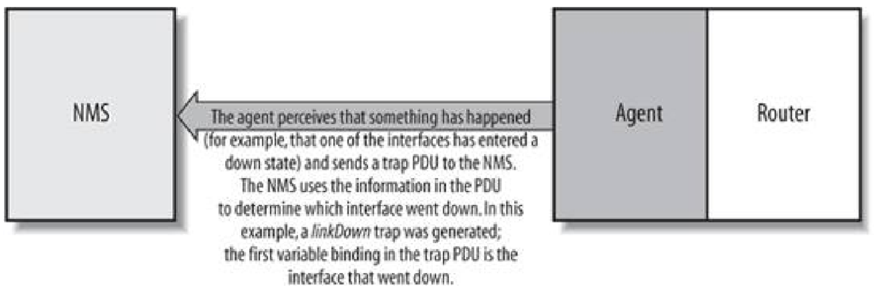
SNMPv2 error message	Description
noAccess(6)	A set to an inaccessible variable was attempted. This typically occurs when the variable has an ACCESS type of
wrongType(7)	An object was set to a type that is different from its definition. This error will occur if you try to set an object that is of type string, for example.
wrongLength(8)	An object's value was set to something other than what it calls for. For instance, a string can be defined to have a maximum character size. This error occurs if you try to set a string object to a value that exceeds its maximum length.
wrongEncoding(9)	A set operation was attempted using the wrong encoding for the object being set.
wrongValue(10)	A variable was set to a value it doesn't understand. This can occur when a read-write is defined as an enumeration, and you try to set it to a value that is not one of the enumerated types.
noCreation(11)	You tried to set a nonexistent variable or create a variable that doesn't exist in the MIB.
inconsistentValue	A MIB variable is in an inconsistent state and is not accepting any set requests.
resourceUnavailable(13)	No system resources are available to perform a set.
commitFailed(14)	This error is a catchall for set failures.

SNMPv2 error message	Description
undoFailed(15)	A set failed and the agent was unable to roll back all the previous sets up until the point of failure.
authorizationError(16)	An SNMP command could not be authenticated; in other words, someone has supplied an incorrect community string.
notWritable(17)	A variable will not accept a set, even though it is supposed to.
inconsistentName(18)	You attempted to set a variable, but that attempt failed because the variable was in some kind of inconsistent state.

2.6.6. SNMP Traps

A trap is a way for an agent to tell the NMS that something bad has happened. In "Managers and Agents" in Chapter 1, we explored the notion of traps at a general level; now we'll look at them in a bit more detail. Figure 2-9 shows the trap-generation sequence

Figure 2-9. Trap-generation sequence


NMS
The agent perceives that something has happened (for example, that one of the interfaces has entered a down state) and sends a trap PDU to the NMS. The NMS uses the information in the PDU to determine which interface went down. In this example, a *linkDown* trap was generated; the first variable binding in the trap PDU is the interface that went down.
Agent
Router

The trap originates from the agent and is sent to the trap destination, as configured within the agent itself. The trap destination is typically the IP address of the NMS. No acknowledgment is sent from the NMS to the agent, so the agent has no way of knowing if the trap makes it to the NMS. Since SNMP uses UDP, and since traps are designed to report problems with your network, traps are especially prone to getting lost and not making it to their destinations. However, the fact that traps can get lost doesn't make them any less useful; in a well-planned environment, they are an integral part of network management. It's better for your equipment to try to tell you that something is wrong, even if the message may never reach you, than simply to give up and let you guess what happened. Here are a few situations that a trap might report:

- A network interface on the device (where the agent is running) has gone down.
- A network interface on the device (where the agent is running) has come back up.
- An incoming call to a modem rack was unable to establish a connection to a modem.
- The fan on a switch or router has failed.

When an NMS receives a trap, it needs to know how to interpret it; that is, it needs to know what the trap means and how to interpret the information it carries. A trap is first identified by its generic trap number. There are seven generic trap numbers (0-6), shown in
Generic trap 6 is a special catchall category for "enterprise-specific " traps, which are traps defined by vendors or users that fall outside of the six generic trap categories. Enterprise-specific traps are further identified by an enterprise ID (i.e., an object ID somewhere in the *enterprises* branch of the MIB tree,*iso.org.dod.internet.private.enterprises* and a specific trap number chosen by the enterprise that defined the trap. Thus, the object ID of an enterprise-specific trap is *enterprise-id.specific-trap-number*. For example, when Cisco defines special traps for its private MIBs, it places them all in its enterprise-specific MIB tree (*iso.org.dod.internet.private.enterprises.cisco*). As we'll see in Chapter 9
define your own enterprise-specific traps ; the only requirement is that you register your own

enterprise number with IANA.

A trap is usually packed with information. As you'd expect, this information is in the form of MIB objects and their values; as mentioned earlier, these object-value pairs are known as variable bindings . For the generic traps 0 through 5, knowledge of what the trap contains is generally built into the NMS software or trap receiver. The variable bindings contained by an enterprise-specific trap are determined by whomever defined the trap. For example, if a modem in a modem rack fails, the rack's agent may send a trap to the NMS informing it of the failure. The trap will most likely be an enterprise-specific trap defined by the rack's manufacturer; the trap's contents are up to the manufacturer, but it will probably contain enough information to let you determine exactly what failed (for example, the position of the modem card in the rack and the channel on the modem card).

Table 2-8. Generic traps

Generic trap name and number	Definition
coldStart (0)	Indicates that the agent has rebooted. All management variables will be reset; specifically, Counters and Gauges will be reset to zero (0). One nice thing about the coldStart trap is that it can be used to determine when new hardware is added to the network. When a device is powered on, it sends this trap to its trap destination. If the trap destination is set correctly (i.e., to the IP address of your NMS), the NMS can receive the trap and determine whether it needs to manage the device.
warmStart (1)	Indicates that the agent has reinitialized itself. None of the management variables will be reset.
linkDown (2)	Sent when an interface on a device goes down. The first variable binding identifies the index in the *interfaces* that went down.
linkUp (3)	Sent when an interface on a device comes back up. The first variable binding identifies which interface came back up.
authenticationFailure (4)	Indicates that someone has tried to query your agent with an incorrect community string; useful in determining if someone is trying to gain unauthorized access to one of your devices.
egpNeighborLoss (5)	Indicates that an EGP neighbor has gone down.
enterpriseSpecific (6)	Indicates that the trap is enterprise-specific. SNMP vendors and users define their own traps under the *private-enterprise* of the SMI *object* tree. To process this trap properly, the NMS has to decode the specific trap number that is part of the SNMP message.

In Chapter 1, we mentioned that RFC 1697 is the RDBMS MIB. One of the traps defined by this MIB is *rdbmsOutOfSpace*:

```
rdbmsOutOfSpace TRAP-TYPE
    ENTERPRISE  rdbmsTraps
    VARIABLES   { rdbmsSrvInfoDiskOutOfSpaces }
    DESCRIPTION
        "An rdbmsOutOfSpace trap signifies that one of the database
         servers managed by this agent has been unable to allocate
         space for one of the databases managed by this agent. Care
         should be taken to avoid flooding the network with these traps."
    ::= 2
```

The enterprise is *rdbmsTraps* and the specific trap number is 2. This trap has one variable binding, *rdbmsSrvInfoDiskOutOfSpaces*. If we look elsewhere in the MIB, we will find that this variable is a scalar object. Its definition is:

```
rdbmsSrvInfoDiskOutOfSpaces OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The total number of times the server has been unable to obtain
         disk space that it wanted, since server startup. This would be
         inspected by an agent on receipt of an rdbmsOutOfSpace trap."
    ::= { rdbmsSrvInfoEntry  9 }
```

The DESCRIPTION for this object indicates why the note about taking care to avoid flooding the network (in the DESCRIPTION text for the trAP-TYPE) is so important. Every time the RDBMS is unable to allocate space for the database, the agent will send a trap. A busy (and full) database could end up sending this trap thousands of times a day.

Some commercial RDBMS vendors, such as Oracle, provide an SNMP agent with their database engines. Agents such as these typically have functionality above and beyond that found in the RDBMS MIB.

Now let's look at an Ethereal trace of an SNMPv1 trap. Given the following command:

[*] Don't worry about the details; they will be explained in Chapter 9

```
$ snmptrap -v 1 -c public .1.3.6.1.4.1.2789.2005 127.0.0.1 6\ 2476317 '' .1.3.6.1.4.
1.2789.2005.1 s "WWW Server Has Been\ Restarted"
```

Ethereal gives us the following trace:

```
Frame 1 (135 bytes on wire, 135 bytes captured)
    Arrival Time: Sep 20, 2004 14:38:40.191174000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 135 bytes
    Capture Length: 135 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 121
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7c72 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: 34108 (34108), Dst Port: snmptrap (162)
    Source port: 34108 (34108)
    Destination port: snmptrap (162)
    Length: 101
    Checksum: 0xfe78 (incorrect, should be 0xf82d)
Simple Network Management Protocol
    Version: 1 (0)
    Community: public
    PDU type: TRAP-V1 (4)
    Enterprise: 1.3.6.1.4.1.2789.2005 (SNMPv2-SMI::enterprises.2789.2005)
    Agent address: 127.0.0.1 (127.0.0.1)
    Trap type: ENTERPRISE SPECIFIC (6)
    Specific trap type: 2476317
    Timestamp: 181730327
    Object identifier 1: 1.3.6.1.4.1.2789.2005.1 (SNMPv2-SMI::enterprises.2789.2005.1)
    Value: STRING: "WWW Server Has Been Restarted"

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 79 00 00 40 00 00 11 7c 72 7f 00 00 01 7f 00   .y..@...|r......
0020  00 01 85 3c 00 a2 00 65 fe 78 30 5b 02 01 00 04   ...<...e.x0[....
0030  06 70 75 62 6c 69 63 a4 4e 06 09 2b 06 01 04 01   .public.N..+....
0040  95 65 8f 55 40 04 7f 00 00 01 02 01 06 02 03 25   .e.U@..........%
0050  c9 1d 43 04 0a d4 fc 17 30 2d 30 2b 06 0a 2b 06   ..C.....0-0+..+.
0060  01 04 01 95 65 8f 55 01 04 1d 57 57 57 20 53 65   ....e.U...WWW Se
0070  72 76 65 72 20 48 61 73 20 42 65 65 6e 20 52 65   rver Has Been Re
0080  73 74 61 72 74 65 64                              started
```

We have only one frame since the agent initiated the trap. An SNMPv1 trap is sent from the agent and is not acknowledged by the receiver in any way, so the agent sends it and forgets about it. This is why we see just the single trace.

2.6.7. SNMP Notification

In an effort to standardize the PDU format of SNMPv1 traps (recall that SNMPv1 traps have a different PDU format from get and set), SNMPv2 defines a NOTIFICATION-TYPE for NOTIFICATION-TYPE is identical to that for get and set. RFC 2863 redefines the notification type like so:

```
linkDown NOTIFICATION-TYPE
```

```
    OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }
    STATUS  current
    DESCRIPTION
        "A linkDown trap signifies that the SNMPv2 entity, acting in an
         agent role, has detected that the ifOperStatus object for one
         of its communication links left the down state and transitioned
         into some other state (but not into the notPresent state). This
         other state is indicated by the included value of ifOperStatus."
    ::= { snmpTraps 3 }
```

The list of bindings is called OBJECTS rather than VARIABLES, but little else has changed. The first object is the specific interface (*ifIndex*) that transitioned from the *linkDown* other condition. The OID for this trap is *1.3.6.1.6.3.1.1.5.3*, or *iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObjects.snmpTraps.linkDown*

Let's look at how to create an SNMP notification:

```
$ snmptrap -v2c -c public 127.0.0.1 '' .1.3.6.1.6.3.1.1.5.3 ifIndex i 2 ifAdminStatus
i 1 ifOperStatus i 1
```

The datagram trace is as follows:

```
Frame 1 (162 bytes on wire, 162 bytes captured)
    Arrival Time: Sep 20, 2004 14:38:53.846768000
    Time delta from previous packet: 0.000000000 seconds
    Time since reference or first frame: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 162 bytes
    Capture Length: 162 bytes
Ethernet II, Src: 00:00:00:00:00:00, Dst: 00:00:00:00:00:00
    Destination: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Source: 00:00:00:00:00:00 (00:00:00_00:00:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 127.0.0.1 (127.0.0.1), Dst Addr: 127.0.0.1 (127.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 148
    Identification: 0x0000 (0)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 0
    Protocol: UDP (0x11)
    Header checksum: 0x7c57 (correct)
    Source: 127.0.0.1 (127.0.0.1)
    Destination: 127.0.0.1 (127.0.0.1)
User Datagram Protocol, Src Port: 34108 (34108), Dst Port: snmptrap (162)
    Source port: 34108 (34108)
    Destination port: snmptrap (162)
    Length: 128
    Checksum: 0xfe93 (incorrect, should be 0x76ba)
Simple Network Management Protocol
    Version: 2C (1)
    Community: public
    PDU type: TRAP-V2 (7)
    Request Id: 0x6737908a
    Error Status: NO ERROR (0)
    Error Index: 0
    Object identifier 1: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
    Value: Timeticks: (181731693) 21 days, 0:48:36.93
    Object identifier 2: 1.3.6.1.6.3.1.1.4.1.0 (SNMPv2-MIB::snmpTrapOID.0)
    Value: OID: IF-MIB::linkDown
    Object identifier 3: 1.3.6.1.2.1.2.2.1.1 (IF-MIB::ifIndex)
    Value: INTEGER: 2
    Object identifier 4: 1.3.6.1.2.1.2.2.1.7 (IF-MIB::ifAdminStatus)
    Value: INTEGER: up(1)
    Object identifier 5: 1.3.6.1.2.1.2.2.1.8 (IF-MIB::ifOperStatus)
    Value: INTEGER: up(1)

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00   ..............E.
0010  00 94 00 00 40 00 00 11 7c 57 7f 00 00 01 7f 00   ....@...|W......
0020  00 01 85 3c 00 a2 00 80 fe 93 30 76 02 01 01 04   ...<......0v....
0030  06 70 75 62 6c 69 63 a7 69 02 04 67 37 90 8a 02   .public.i..g7...
0040  01 00 02 01 00 30 5b 30 10 06 08 2b 06 01 02 01   .....0[0...+....
0050  01 03 00 43 04 0a d5 01 6d 30 17 06 0a 2b 06 01   ...C....m0...+..
```

```
0060  06 03 01 01 04 01 00 06 09 2b 06 01 06 03 01 01   .........+......
0070  05 03 30 0e 06 09 2b 06 01 02 01 02 02 01 01 02   ..0...+.........
0080  01 02 30 0e 06 09 2b 06 01 02 01 02 02 01 07 02   ..0...+.........
0090  01 01 30 0e 06 09 2b 06 01 02 01 02 02 01 08 02   ..0...+.........
00a0  01 01                                             ..
```

2.6.8. SNMP inform

SNMPv2 provides an inform mechanism , which allows for acknowledged sending of traps. This operation can be useful when the need arises for more than one NMS in the network. When an inform is sent, the receiver sends a response to the sender acknowledging receipt of the event. This behavior is similar to that of the get and set requests. Note that an SNMP inform can be used to send SNMPv2 traps to an NMS. If you use an inform for this purpose, the agent will be notified when the NMS receives the trap.

2.6.9. SNMP report

The report operation was defined in the draft version of SNMPv2 but was never implemented. It is now part of the SNMPv3 standard and is intended to allow SNMP engines to communicate with each other (mainly to report problems with processing SNMP messages).

PREV NEXT

2.7. Host Management Revisited

Managing your hosts is an important part of network management. You would think that the Host Resources MIB would be part of every host-based SNMP agent, but this isn't the case. Some SNMP agents implement this MIB, but many don't. A few agents go further and implement proprietary extensions based upon this MIB. This is mainly due to the fact that this MIB was intended to serve as a basic, watered-down framework for host management , designed mainly to foster wide deployment.

The Host Resources MIB defines the following seven groups:

```
host           OBJECT IDENTIFIER ::= { mib-2 25 }

hrSystem        OBJECT IDENTIFIER ::= { host 1 }
hrStorage       OBJECT IDENTIFIER ::= { host 2 }
hrDevice        OBJECT IDENTIFIER ::= { host 3 }
hrSWRun         OBJECT IDENTIFIER ::= { host 4 }
hrSWRunPerf     OBJECT IDENTIFIER ::= { host 5 }
hrSWInstalled   OBJECT IDENTIFIER ::= { host 6 }
```

The *host* OID is *1.3.6.1.2.1.25* (*iso.org.dod.internet.mgmt.mib-2.host*). The remaining six groups define various objects that provide information about the system.

The *hrSystem* (*1.3.6.1.2.1.25.1*) group defines objects that pertain to the system itself. These objects include uptime, system date, system users, and system processes.

The *hrDevice* (*1.3.6.1.2.1.25.3*) and *hrStorage* (*1.3.6.1.2.1.25.2*) groups define objects pertaining to filesystems and system storage, such as total system memory, disk utilization, and CPU nonidle percentage. They are particularly helpful since they can be used to manage the disk partitions on your host. You can even use them to check for errors on a given disk device.

The *hrSWRun* (*1.3.6.1.2.1.25.4*), *hrSWRunPerf* (*1.3.6.1.2.1.25.5*), and *hrSWInstalled* (*1.3.6.1.2.1.25.6*) groups define objects that represent various aspects of software running or installed on the system. From these groups, you can determine what operating system is running on the host, as well as what programs the host is currently running. The *hrSWInstalled* group can be used to track which software packages are installed.

As you can see, the Host Resources MIB provides some necessary system-management objects that can be utilized by almost anyone who needs to manage critical systems.

PREV NEXT

2.8. Remote Monitoring Revisited

A thorough treatment of RMON is beyond the scope of this book, but it's worth discussing the groups that make up RMONv1. RMON probes are typically standalone devices that watch traffic on the network segments to which they are attached. Some vendors implement at least some kind of RMON probe in their routers, hubs, or switches. Chapter 8 provides an example of how to configure RMON on a Cisco router.

The RMON MIB defines the following 10 groups:

```
rmon          OBJECT IDENTIFIER ::= { mib-2 16 }
statistics    OBJECT IDENTIFIER ::= { rmon 1 }
history       OBJECT IDENTIFIER ::= { rmon 2 }
alarm         OBJECT IDENTIFIER ::= { rmon 3 }
hosts         OBJECT IDENTIFIER ::= { rmon 4 }
hostTopN      OBJECT IDENTIFIER ::= { rmon 5 }
matrix        OBJECT IDENTIFIER ::= { rmon 6 }
filter        OBJECT IDENTIFIER ::= { rmon 7 }
capture       OBJECT IDENTIFIER ::= { rmon 8 }
event         OBJECT IDENTIFIER ::= { rmon 9 }
```

RMONv1 provides packet-level statistics about an entire LAN or WAN. The *rmon* OID is *1.3.6.1.2.1.16* (*iso.org.dod.internet.mgmt.mib-2.rmon*). RMONv1 is made up of nine groups:

statistics (*1.3.6.1.2.1.16.1*)

Contains statistics about all the Ethernet interfaces monitored by the probe.

history (*1.3.6.1.2.1.16.2*)

Records periodic statistical samples from the *statistics* group.

alarm (*1.3.6.1.2.1.16.3*)

Allows a user to configure a polling interval and a threshold for any object the RMON probe records.

hosts (1.3.6.1.2.1.16.4)

Records traffic statistics for each host on the network.

hostTopN (*1.3.6.1.2.1.16.5*)

Contains host statistics used to generate reports on hosts that top a list ordered by a parameter in the host table.

matrix (*1.3.6.1.2.1.16.6*)

Stores error and utilization information for sets of two addresses.

filter (*1.3.6.1.2.1.16.7*)

Matches packets based on a filter equation; when a packet matches the filter, it may be captured or an event may be generated.

capture (*1.3.6.1.2.1.16.8*)

Allows packets to be captured if they match a filter in the filter group.

event (*1.3.6.1.2.1.16.9*)

Controls the definition of RMON events.

RMONv2 enhances RMONv1 by providing network- and application-level statistical gathering. Since the only example of RMON in this book uses RMONv1, we will stop here and not go into RMONv2. However, we encourage you to read RFC 2021 to get a feel for what enhancements this version of RMON brings to network monitoring.

PREV NEXT

2.9. Reverse Engineering SNMP

You might be wondering why something like this is even a topic for SNMP. Isn't SNMP a standard, you may ask? Well, it is, but that doesn't prevent vendors from doing things in nonstandard, and downright oblique, ways. In some cases, vendors either do not publish their SNMP MIB, or they use SNMP as a means of updating a network device from a GUI. For example, the Netgear WAG302 access point comes with Windows-based management software. This software uses SNMP to gather information from the WAP. The Netgear device supports several standard SNMP MIBs, but it also has support for two additional private MIBs: Netgear's MIB and that of a third-party provider. Netgear doesn't make its private MIB available. Using Ethereal (yes, it is available for Windows, too), you can capture the traffic as you work with a management application, such as the one that comes with the Netgear device, and see what SNMP requests and responses flow over the network.

As we mentioned already, Ethereal does a nice job of telling you things like the SNMP version, error codes, OIDs, and actual data in the PDU. We even get to see the OIDs and their values. For example, the following is an excerpt from the notification trace:

```
Object identifier 3: 1.3.6.1.2.1.2.2.1.1 (IF-MIB::ifIndex)
Value: INTEGER: 2
Object identifier 4: 1.3.6.1.2.1.2.2.1.7 (IF-MIB::ifAdminStatus)
Value: INTEGER: up(1)
Object identifier 5: 1.3.6.1.2.1.2.2.1.8 (IF-MIB::ifOperStatus)
Value: INTEGER: up(1)
```

We see that *ifIndex* is set to INTEGER 2, *ifAdminStatus* is set to INTEGER 1 (which Ethereal has translated to up for us), and *ifOperStatus* is set to up as well.

We suggest that you add Ethereal to your arsenal of network tools. It can help you greatly, not only in reverse engineering SNMP, but also in terms of learning about datagram structures and the like.

PREV NEXT

Chapter 3. SNMPv3

Security has been the biggest weakness of SNMP since the beginning. Authentication in SNMP versions 1 and 2 amounts to nothing more than a password (community string) sent in clear text between a manager and agent. Any security-conscious network or system administrator knows that clear-text passwords provide no real security at all. It is trivial for someone to intercept the community string, and once he has it, he can use it to retrieve information from devices on your network, modify their configuration, and even shut them down.

The Simple Network Management Protocol Version 3 (SNMPv3) addresses the security problems that have plagued both SNMPv1 and SNMPv2. For all practical purposes, security is the only issue SNMPv3 addresses; there are no other changes to the protocol. There are no new operations; SNMPv3 supports all the operations defined by versions 1 and 2. There are several new textual conventions, but these are really just more precise ways of interpreting the datatypes that were defined in earlier versions.

This chapter provides an introduction to SNMPv3. SNMPv3 agent configurations can be found in Chapter 6. Until recently, SNMPv3 was a draft standard. It is now a full standard. Vendors are notoriously slow to change, but hopefully we will see even more of them begin to support SNMPv3.

3.1. Changes in SNMPv3

Although SNMPv3 makes no changes to the protocol aside from the addition of cryptographic security, its developers have managed to make things look much different by introducing new textual conventions, concepts, and terminology. The changes to the terminology are so radical that it's hard to believe the new terms essentially describe the same software as the old ones, but they do. However, they do differ in how they relate to each other, and they specify much more precisely the pieces that an SNMP implementation needs.

The most important change is that Version 3 abandons the notion of managers and agents. Both managers and agents are now called SNMP entities
entity consists of an SNMP engine and one or more SNMP applications, which are discussed in the following sections. These new concepts are important because they define an architecture rather than simply a set of messages; the architecture helps to separate different pieces of the SNMP system in a way that makes a secure implementation possible. Let's look at what these concepts mean, starting with the RFCs that define them (Table 3-1).

Table 3-1. RFCs for SNMPv3

Number	Name
RFC 3411	Architecture for SNMP Frameworks
RFC 3412	Message Processing and Dispatching
RFC 3413	SNMP Applications
RFC 3414	User-based Security Model (USM)
RFC 3415	View-based Access Control Model (VACM)
RFC 3416	Protocol Operations for SNMPv2
RFC 3417	Transport Mappings for SNMPv2
RFC 3418	MIB for SNMPv2
RFC 2576	Coexistence Between SNMP Versions
RFC 2570	Introduction to SNMPv3
RFC 2786	Diffie-Hellman USM Key Management

Note that USM and VACM are discussed in a little more detail later in this chapter.

3.1.1. The SNMPv3 Engine

The engine is composed of four pieces: the Dispatcher, the Message Processing Subsystem , the Security Subsystem , and the Access Control Subsystem. The Dispatcher's job is to send and receive messages. It tries to determine the version of each received message (i.e., v1, v2, or v3) and, if the version is supported, hands the message off to the Message Processing Subsystem. The Dispatcher also sends SNMP messages to other entities.

The Message Processing Subsystem prepares messages to be sent and extracts data from received messages. A Message Processing Subsystem can contain multiple message processing modules. For example, a subsystem can have modules for processing SNMPv1, SNMPv2, and SNMPv3 requests. It may also contain a module for other processing models that are yet to be defined.

The Security Subsystem provides authentication and privacy services. Authentication uses either community strings (SNMP v1 and v2) or SNMPv3 user-based authentication. User-based authentication uses the MD5 or SHA algorithms to authenticate users without sending a password in the clear. The privacy service uses the DES algorithm to encrypt and decrypt SNMP messages. Currently, DES is the only algorithm used, though others may be added in the future.

The Access Control Subsystem is responsible for controlling access to MIB objects. You can control what objects a user can access as well what operations she is allowed to perform on those objects. For example, you might want to limit a user's read-write access to certain parts of the *mib-2* tree while allowing read-only access to the entire tree.

3.1.2. SNMPv3 Applications

Version 3 divides most of what we have come to think of as SNMP into a number of applications :

Command generator

Generates get, getnext, getbulk, and set requests and processes the responses. This application is implemented by an NMS, so it can issue queries and set requests against entities on routers, switches, Unix hosts, etc.

Command responder

Responds to get, getnext, getbulk, and set requests. This application is implemented by an entity on a Cisco router or Unix host. (For versions 1 and 2, the command responder is implemented by the SNMP agent.)

Notification originator

Generates SNMP traps and notifications. This application is implemented by an entity on a router or Unix host. (For versions 1 and 2, the notification originator is part of an SNMP agent. Freestanding utilities for generating traps are also available.)

Notification receiver

Receives traps and inform messages . This application is implemented by an NMS.

Proxy forwarder

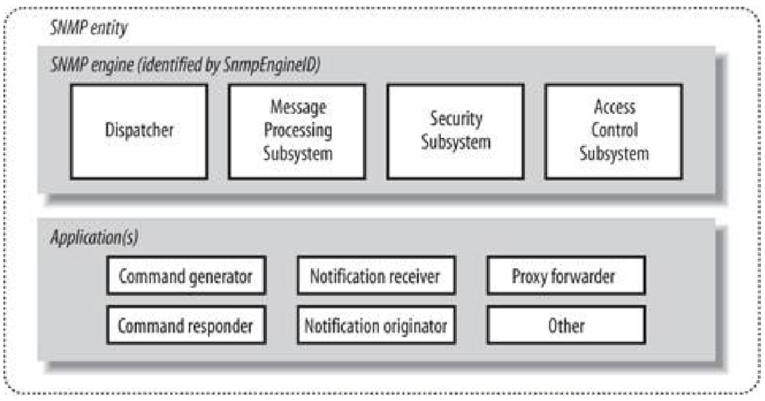
Facilitates message passing between entities.

RFC 3411 allows additional applications to be defined over time. This ability to extend the SNMPv3 framework is a significant advantage over the older SNMP versions.

3.1.3. What Does an Entity Look Like?

Thus far, we've talked about the SNMPv3 entity in terms of abstract definitions. Figure 3-1 (taken from RFC 3411) shows how the components that make up an entity fit together.

Figure 3-1. SNMPv3 entity



3.1.4. SNMPv3 Textual Conventions

SNMPv3 defines a number of additional textual conventions , outlined in 3-2.

Table 3-2. SNMPv3 textual conventions

Textual convention	Description
snmpEngineID	An administratively unique identifier for an SNMP engine. Objects of this type are for identification, not for addressing, even though an address can be used in the generation of a specific value. RFC 3411 provides a detailed discussion of how snmpEngineID created.
snmpSecurityModel	An SNMP securityModel (SNMPv1, SNMPv2, or USM). USM stands for User-based Security Model, which is the security method used in SNMPv3.
snmpMessageProcessingModel	A message processing model used by the Message Processing Subsystem.
snmpSecurityLevel	The level of security at which SNMP messages can be sent, or the level of security at which operations are being processed. Possible values are noAuthNoPriv (without authentication and without privacy), (with authentication but without privacy), and authPriv (with authentication and with privacy). These three values are ordered such that noAuthNoPriv is less than authNoPriv and authNoPriv less than authPriv.
snmpAdminString	An octet string containing administrative information, preferably in human-readable form. The string can be up to 255 bytes long.
snmpTagValue	An octet string containing a tag value. Tag values are preferably in human-readable form. According to RFC 3413, valid example tags include acme, router, and host.
snmpTagList	An octet string containing a list of tag values. Tag values are preferably in human-readable form. According to RFC 3413, valid examples

Textual convention	Description
	of a tag list are the empty string, acme router host managerStation.
KeyChange	An object used to change authentication and privacy keys .

The next two sections will look at the USM and VACM in a little more detail.

PREV NEXT

PREV NEXT

3.2. USM

The User-based Security Model (USM) and the View Access Control Model (VACM) together detail the security enhancements added with SNMPv3 . Let's start with the USM.

3.2.1. The Basics

We need to get some terminology out of the way before we can look at the USM in any detail:

snmpEngineID

This is an unambiguous identifier for an SNMP engine as well as the SNMP entity that corresponds to the engine. The syntax for this identifier is and it cannot be zero length. Most SNMPv3 applications allow for the user to input a value for snmpEngineID. If one is not specified, the value is computed using a combination of enterprise ID and IP or MAC address.

snmpEngineBoots

A count of the number of times an SNMP engine has rebooted.

snmpEngineTime

The number of seconds since the snmpEngineBoots counter was last incremented.

snmpSecurityLevel

There are three security levels. The first is no authentication or privacy (noAuthNoPriv). Note that if this mode is used, a securityName is still required. The second is authentication and no privacy (authNoPriv). The third and final one is authentication and privacy (authPriv). While you can have authentication without privacy, you cannot have privacy without authentication.

Authoritative SNMP engine

A nonauthoritative engine must discover the snmpEngineId of the authoritative engine with which it communicates. The rules for designating the authoritative engine are as follows: if the SNMP message requires a response (get, getnext, getbulk, set, or inform), the receiver of these messages is authoritative. If the message does not require a response (trap or report), the sender of the message is authoritative. Generally, an SNMP agent is authoritative and an NMS is nonauthoritative.

An SNMPv3 message (packet) format has the following fields:

msgVersion

The SNMP version of the message, set to 3.

msgID

The msgID is used between a manager and agent to coordinate request and response messages.

msgMaxSize

The msgMaxSize is the maximum message size supported by a sender of an SNMP message.

msgFlags

msgFlags is an 8-bit value that specifies whether a report PDU is to be generated, whether privacy is used, and whether authentication is used.

msgSecurityModel

Specifies which security model was used by the sender of the message. Current values are 1, 2, and 3 for SNMPv1, SNMPv2c, and SNMPv3, respectively.

msgSecurityParameters

msgSecurityParameters contains security-specific information.

contextEngineID

Uniquely identifies an SNMP entity. An SNMP entity is the combination of an SNMP engine and SNMP applications. This is discussed in the section on VACM.

contextName

contextName identifies a particular context within an SNMP engine.

scopedPDU

A block of data made up of a contextEngineID, contextName, and SNMP PDU.

The msgSecurityParameters in an SNMPv3 message are as follows:

msgAuthoritativeEngineID

The snmpEngineID of the authoritative engine.

msgAuthoritativeEngineBoots

The snmpEngineBoots of the authoritative engine.

msgAuthoritativeEngineTime

The snmpEngineTime of the authoritative engine.

msgUserName

The user who may be authenticating and encrypting the message.

msgAuthenticationParameters

This value is null if no authentication is used. Otherwise, the field contains the computer HMAC message digest for the message. Currently the RFC specifies that MD5 and SHA must be used.

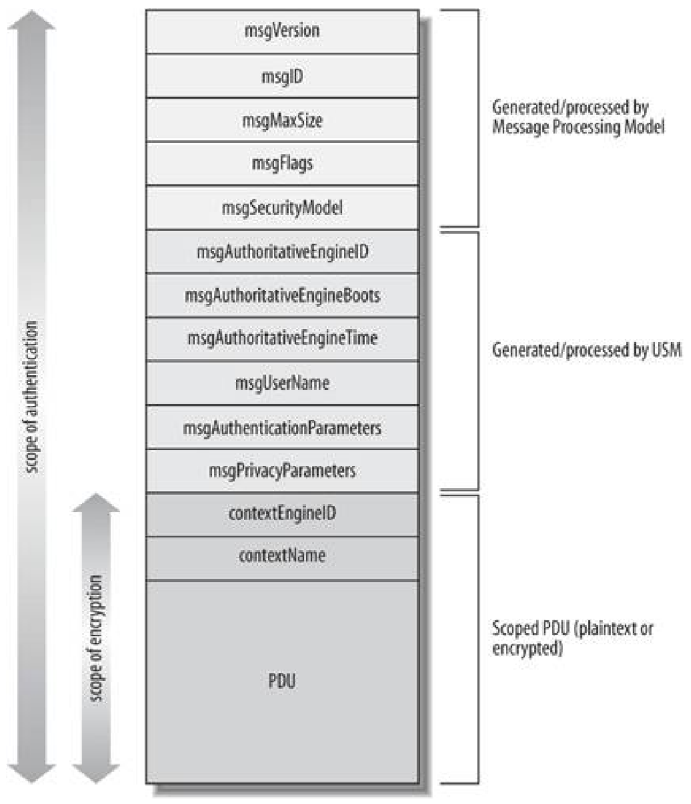
msgPrivacyParameters

This value is null if no encryption is used. Otherwise, this field is used to form the initial value of the Cipher Block Chaining mode of the Data Encryption Standard (CBC-DES) algorithm.

Figure 3-2[*] shows the entire SNMPv3 message.

[*] This image is reprinted from the paper "SNMPv3: A Security Enhancement for SNMP" by William Stallings, which can be found online at http://www.comsoc.org/livepubs/surveys/public/4q98issue/stallings.html

Figure 3-2. SNMPv3 message format


msgVersion
msgID
msgMaxSize
msgFlags
msgSecurityModel
msgAuthoritativeEngineID
msgAuthoritativeEngineBoots
msgAuthoritativeEngineTime
msgUserName
msgAuthenticationParameters
msgPrivacyParameters
contextEngineID
contextName
PDU
scope of authentication
scope of encryption
Generated/processed by Message Processing Model
Generated/processed by USM
Scoped PDU (plaintext or encrypted)

3.2.2. Discovery

The USM requires that the msgSecurityParameters contain the snmpEngineID, snmpEngineBoots snmpEngineTime of the authoritative engine. Before any get, getnext, or set operation can be used, the nonauthoritative engine must obtain these values from the authoritative engine. A discovery process is used to obtain this information.

3.2.3. USM Timeliness

Once a nonauthoritative engine has learned the value of snmpEngineBoots and snmpEngineTime, it must maintain its own local notion of what these values are supposed to be. The nonauthoritative engine increments the learned snmpEngineTime so that it stays up-to-date with the authoritative engine's own notion of snmpEngineTime rolls over, snmpEngineBoots must be incremented. The USM Timeliness Module is intended to help thwart message delay or replay.

3.2.4. Authentication

MD5, or Message Digest 5, and SHA1, or Secure Hash Algorithm 1, are used for authenticating SNMPv3 messages. MD5 creates a digest of 128 bits and SHA1 creates a digest of 160 bits. Both digests are fixed in size and cannot be used solely for authentication . The keyed Hashing for Message Authentication (HMAC) algorithm is used in conjunction with MD5 and SHA1 to compute message digests. An authentication passphrase or secret key is appended to the data before the digests are computed. The secret key must be known by both the sender and the receiver. The RFCs specify that this passphrase must be at least eight characters long.

3.2.5. Privacy

Encryption of SNMP data is accomplished by using the CBC-DES algorithm. As with authentication, a secret key or passphrase must be known by the sender and receiver and used in the encryption process. A USM User Table is used to store the passphrase and other details transmitted with the packet in the msgPrivacyParameters

3.2.6. USM User Table

Every entity maintains a User Table that stores all the users who have access to the system via SNMP. The User Table includes the following elements :

Username

A textual username. Sometimes referred to as a security name.

Authentication protocol

Details what, if any, authentication protocol is to be used. Valid values include usmNoAuthProtocol, usmHMACMD5AuthProtocol, andusmHMACSHAAuthProtocol.

Authentication key

The passphrase used for authentication. Must be at least eight characters long.

Privacy protocol

Details what, if any, privacy protocol is to be used. Valid values include usmNoPrivProtocol and usmDESPrivProtocol.

Privacy key

The passphrase used for privacy. Must be at least eight characters long.

usmUserSpinLock

The usmUserSpinLock is an advisory lock that allows for the coordination of multiple attempts to modify the User Table.

3.2.7. Localized Keys and Changing Keys

A localized key allows for the same passphrase to be used by a single user on many different engines. It keeps an operator from having to remember a different passphrase for each SNMP engine he must interact with. The KeyChange users to change their keys securely.

3.3. VACM

VACM is used to control access to managed objects in a MIB or MIBs. This is where the Access Control Subsystem comes into play.

3.3.1. The Basics

The `msgFlags`, `msgSecurityModel`, and `scopedPDU` fields are used by VACM for message access. Each parameter is used to determine access to managed objects. An error is returned to the sender if access is not allowed for the request type. VACM makes use of four tables for different aspects of access control. We will discuss these tables next.

3.3.2. Context Table

The `vacmContextTable` is a collection of managed objects that have access constraints which are associated with a context name. The `vacmContextTable` stores all available contexts. The table is indexed by a `contextName`, and each row in this table contains:

vacmContextName

A textual name for the context

3.3.3. Security to Group Table

The `vacmSecurityToGroupTable` is used to store group information. A group is made up of zero or more `securityModel` and `securityName` combinations. This combination defines what managed objects can be accessed. The table itself is indexed by a `securityModel` and contains rows made up of the following columns:

vacmSecurityModel

The security model in usee.g., USM.

vacmSecurityName

In the case of the USM, securityName and userName are identical.

vacmGroupName

A textual name for the group to which this table entry belongs.

3.3.4. Access Table

The vacmAccessTable is used to store the access rights defined for groups. This table is indexed by a groupName, contextPrefix, securityModel, and securityLevel. Each row in this table contains:

vacmGroupName

A name of a group with access rights.

vacmAccessContextMatch

A simple form of wildcarding. A value of *exact* dictates that the index exactly match the value in vacmAccessContextPrefix. If set to*prefix*, the index simply match the first few characters of the value in vacmAccessContextPrefix

vacmAccessContextPrefix

An index contextName must match either exactly or partially the value of vacmAccessContextPrefix.

vacmAccessSecurityModel

The securityModel that must be used to gain access.

vacmAccessSecurityLevel

Defines the minimum securityLevel that must be used to gain access.

vacmAccessReadViewName

The authorized MIB viewName used for read access.

vacmAccessWriteViewName

The authorized MIB viewName used for write access.

vacmAccessNotifyViewName

The authorized MIB viewName used for notify access.

3.3.5. View Tree Family Table

The vacmViewTreeFamilyTable is used to store MIB views. A MIB view is defined as a family of view subtrees that pair an OID subtree value with a mask value. The mask indicates which subidentifiers of the associated subtree OID are significant to the MIB view's definition.

All the MIB views are stored in the vacmViewTreeFamilyTable. It is indexed by a
of a MIB subtree. The VACM MIB defines the vacmViewSpinLock advisory lock that is used to allow several SNMP engines to coordinate modifications to this table. Each row in the vacmViewTreeFamilyTable contains:

vacmViewTreeFamilyViewName

A textual name for the MIB view.

vacmViewTreeFamilySubtree

The OID subtree that, when combined with the mask, defines one or more MIB view subtrees.

vacmViewTreeFamilyMask

A bit mask that, in combination with the corresponding OID subtree, defines one or more MIB view subtrees.

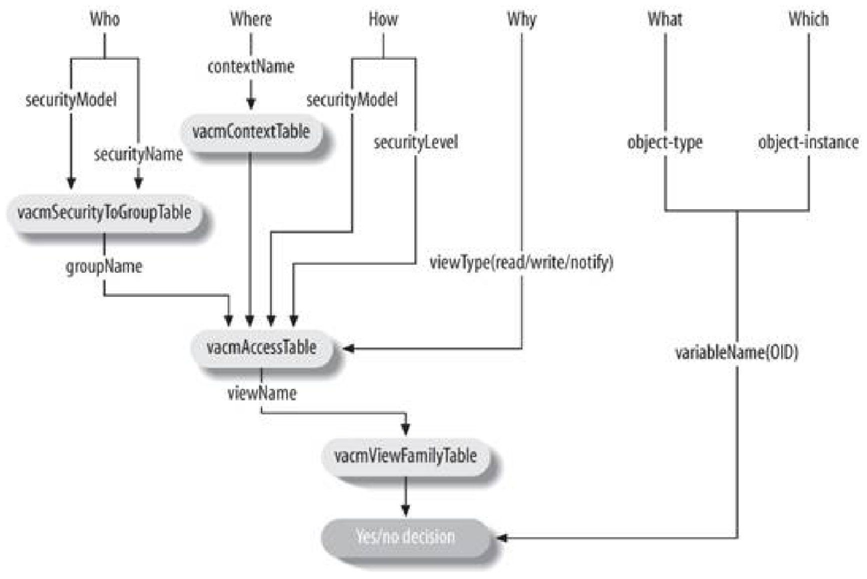
vacmViewTreeFamilyType

Indicates whether the corresponding MIB view subtrees defined by the OID subtree and mask are included or excluded from the MIB view.

Figure 3-3[*] shows the logic flow for VACM.

[*] This image is reprinted from the paper "SNMPv3: A Security Enhancement for SNMP" by William Stallings, which can be found online at http://www.comsoc.org/livepubs/surveys/public/4q98issue/stallings.html

Figure 3-3. VACM logic flow


Who
Where
How
Why
What
Which
securityModel
securityName
contextName
vacmContextTable
securityModel
securityLevel
object-type
object-instance
vacmSecurityToGroupTable
groupName
viewType(read/write/notify)
vacmAccessTable
viewName
variableName(OID)
vacmViewFamilyTable
Yes/no decision

PREV NEXT

3.4. SNMPv3 in the Real World

Let's briefly outline the common configuration options you should expect when you have to configure an SNMPv3 device or network management platform:

Username

This is the textual description of the person responsible for the SNMP entity that is to be managed. Sometimes referred to as security name.

Security level

Some applications require you to explicitly set the security level and others determine it based on the combination of authentication and privacy protocol in use. The specified values are noAuthNoPriv, which is no authentication and no privacy, authNoPriv, which is authentication and no privacy, and authPriv, which is authentication and privacy. Note that you cannot have privacy without authentication, but you can have authentication without privacy.

Authentication protocol

The protocol used for authenticationthat is, to prove that you are who you say you are. Currently, MD5 and SHA1 are specified in the RFCs.

Authentication passphrase

The passphrase used in conjunction with the authentication protocol. It must be at least eight characters long. You may also see it referred to as a password.

Privacy protocol

The protocol used for privacy, that is, to encrypt the data portion of the SNMP packet. Currently, DES is specified in the RFCs.

Privacy passphrase

The passphrase used in conjunction with the privacy protocol. It must be at least eight characters long. You may also see it referred to as a password.

Here are the logical steps you take when using SNMPv3-enabled devices and entities:

1. Create a USM entry on a device with proper USM attributes: username, authentication protocol, etc.

2. Configure the management station (if it supports SNMPv3) with the proper USM attributes for the managed device. Note that the username and passphrases created in step 1 will need to be entered manually in this step.

3. Begin managing the device.

After all the gory technical details, isn't it nice to see that the basics of SNMPv3 really aren't all that scary?

SNMPv3 provides some much-needed security for SNMP. Many vendors already support SNMPv3, but many others, of course, do not. Vendors are often slow to change, mainly because SNMP support is generally an afterthought during the development life cycle of a switch, router, or software system. In fact, SNMP is often a bolt-on feature that isn't heavily tested and is rarely updated. But we in the network management field can only hope that more vendors embrace not only SNMP but also SNMPv3.

PREV NEXT

Chapter 4. NMS Architectures

Now that you understand the basic concepts behind how network management stations (NMSs) and agents communicate, it's time to introduce the concept of a network management architecture . Before rushing out to deploy SNMP management, you owe it to yourself to put some effort into developing a coherent plan. If you simply drop NMS software on a few of your favorite desktop machines, you're likely to end up with something that doesn't work very well. By NMS architecture, we mean a plan that helps you use NMSs effectively to manage your network. A key component of network management is selecting the proper hardware (i.e., an appropriate platform on which to run your NMS) and making sure that your management stations are located in such a way that they can observe the devices on your network effectively.

4.1. Hardware Considerations

Managing a reasonably large network requires an NMS with substantial computing power. In today's complex networked environments, networks can range in size from a few nodes to thousands of nodes. The process of polling and receiving traps from hundreds or thousands of managed entities can be taxing on the best of hardware. Your NMS vendor will be able to help you determine what kind of hardware is appropriate for managing your network. Most vendors have formulas for determining how much RAM you will need to achieve the level of performance you want, given the requirements of your network. It usually boils down to the number of devices you want to poll, the amount of information you will request from each device, and the interval at which you want to poll them. The software you want to run is also a consideration. NMS products such as OpenView are large, heavyweight applications; if you want to run your own scripts with Perl, you can get away with a much smaller management platform.

Is it possible to say something more helpful than "ask your vendor"? Yes. First, although we've become accustomed to thinking of NMS software as requiring a midrange workstation or high-end PC, desktop hardware has advanced so much in the past year or two that running this software is within the range of any modern PC. Specifically, surveying the recommendations of a number of vendors, we have found that they suggest a PC with at least a 2 or 3 GHz CPU, 512 MB to 1 GB of memory, and 1-2 GB of disk space. Requirements for Sun SPARC and HP workstations are similar.

Let's look at each of these requirements:

2 or 3 GHz CPU

> This is well within the range of any modern desktop system, but you probably can't bring your older equipment out of retirement to use as a management station.

512 MB to 1 GB of memory

> You'll probably have to add memory to any off-the-shelf PC; Sun and HP workstations come with more generous memory configurations. Frankly, vendors tend to underestimate memory requirements anyway, so it won't hurt to upgrade to 2 GB. Fortunately, RAM is usually cheap these days, though memory prices fluctuate from day to day.

1-2 GB of disk space

> This recommendation is probably based on the amount of space you'll need to store the software, and not on the space you'll need for logfiles, long-term trend data, etc. But again, disk space is cheap these days, and skimping is counterproductive.

Let's think a bit more about how long-term data collection affects your disk requirements. First, you should recognize that some products have only minimal data-collection facilities, while others exist purely for the purpose of collecting data (for example, MRTG). Whether you can do data collection effectively depends to some extent on the NMS product you've selected. Therefore, before deciding on a software product, you should think about your data-collection requirements. Do you want to do long-term trend analysis? If so, that will affect both the software you choose and the hardware on which you run it.

For a starting point, let's say that you have 1,000 nodes, you want to collect data every minute, and you're collecting 1 KB of data per node. That's 1 MB per minute, 1.4 GB per dayyou'll fill a 40GB disk in about a month. That's bordering on extravagant. But let's look at the assumptions:

- Collecting data every minute is certainly excessive; every 10 minutes should do. Now your 40GB disk will store almost a year's worth of data.
- A network with 1,000 nodes isn't that big. But do you really want to store trend data for all your users' PCs? Much of this book is devoted to showing you how to control the amount of data you collect. Instead of 1,000 nodes, let's first count interfaces. And let's forget about desktop systemswe really care about trend data for our network backbone: key servers, routers, switches, etc. Even on a midsize network, we're probably talking about 100 or 200 interfaces.
- The amount of data you collect per interface depends on many factors, not the least of which is the format of the data. An interface's status may be up or downthat's a single bit. If it's being stored in a binary data structure, it may be represented by a single bit. But if you're using *syslog* to store your log data and writing Perl scripts to do trend analysis, your *syslog* records are going to be 80 bytes or so, even if you are storing only 1 bit of information. Data-storage mechanisms range from *syslog* to fancy database schemesyou obviously need to understand what you're using, and how it will affect your storage requirements. Furthermore, you need to understand how much information you really want to keep per interface. If you want to track only the number of octets going in and out of each interface and you're storing this data efficiently, your 40GB disk could easily last the better part of a century.

Seriously, it's hard to estimate your storage requirements when they vary over two or three orders of magnitude. But the lesson is that no vendor can tell you what your storage requirements will be. A gigabyte should be plenty for log data on a moderately large network, if you're storing data only for a reasonable subset of that network, not polling too often, and not saving too much data. But that's a lot of variables, and you're the only one in control of them. Keep in mind, though, that the more data you collect, the more time and CPU power will be required to grind through all that data and produce meaningful results. It doesn't matter whether you're using expensive trend-analysis software or some homegrown scriptsprocessing lots of data is expensive. At least in terms of long-term data collection, it's probably better to err by keeping too little data around than by keeping too much.

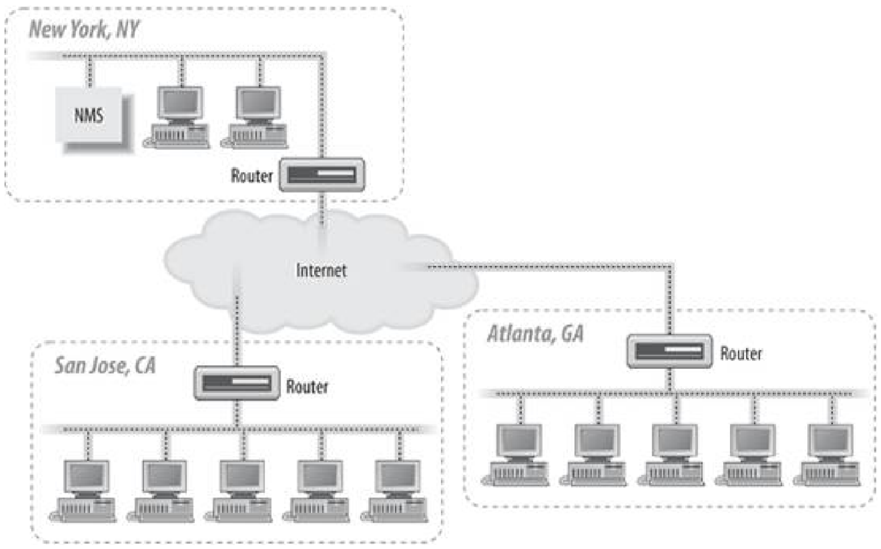
PREV NEXT

4.2. NMS Architectures

Before going out and buying all your equipment, it's worth spending some time coming up with an architecture for your network that will make it more manageable. The simplest architecture has a single management station that is responsible for the entire network, as shown in Figure 4-1.

The network depicted in Figure 4-1 has three sites: New York, Atlanta, and San Jose. The NMS in New York is responsible for managing not only the portion of the network in New York, but also those in Atlanta and San Jose. Traps sent from any device in Atlanta or San Jose must travel over the Internet to get to the NMS in New York. The same thing goes for polling devices in San Jose and Atlanta: the NMS in New York must send its requests over the Internet to reach these remote sites. For small networks, an architecture like this can work well. However, when the network grows to the point that a single NMS can no longer manage everything, this architecture becomes a real problem. The NMS in New York can get behind in its polling of the remote sites, mainly because it has so much to manage. The result is that when problems arise at a remote site, they may not get noticed for some time. In the worst case, they might not get noticed at all.

Figure 4-1. Single NMS architecture

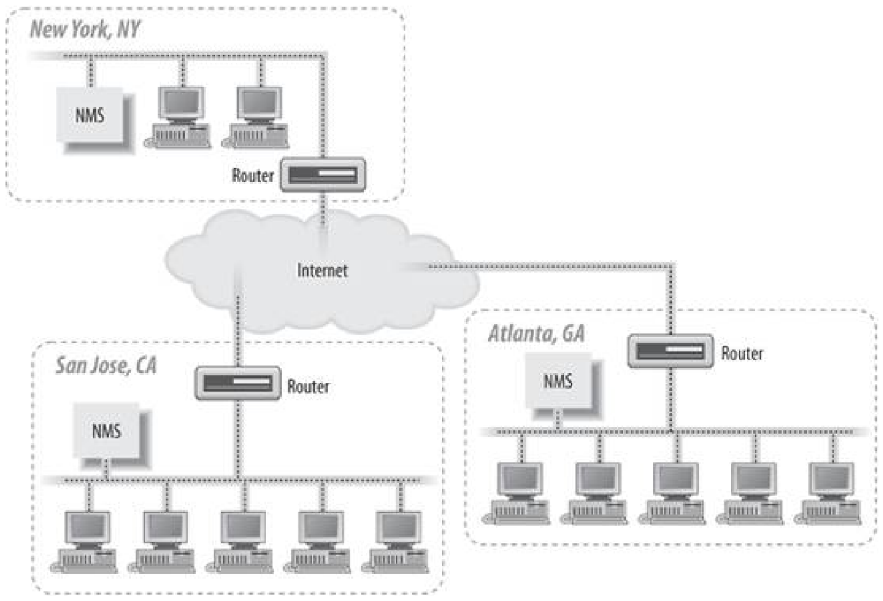


It's also worth thinking about staffing. With a single NMS, your primary operations staff would be in New York, watching the health of the network. But problems frequently require somebody on-site to intervene. This requires someone in Atlanta and San Jose, plus the coordination that entails. You may not need a full-time network administrator, but you will need someone who knows what to do when a router fails.

When your network grows to a point where one NMS can no longer manage everything, it's time to move to a distributed NMS architecture. The idea behind this architecture is simple: use two or more management stations and locate them as close as possible to the nodes they are managing. In the case of our three-site network, we would have an NMS at each site. Figure 4-2 shows the addition of two NMSs to the network.

This architecture has several advantages, not the least of which is flexibility. With the new architecture, the NMSs in Atlanta and San Jose can act as standalone management stations, each with a fully self-sufficient staff, or they can forward events to the NMS in New York. If the remote NMSs forward all events to the NMS in New York, there is no need to put additional operations staff in Atlanta and San Jose. At first glance, this looks like we've returned to the situation of Figure 4-1, but that isn't quite true. Most NMS products provide some kind of client interface for viewing the events currently in the NMS (traps received, responses to polls, etc.). Since the NMS that forwards events to New York has already discovered the problem, we're simply letting the NMS in New York know about it so that it can be dealt with appropri-

Figure 4-2. Distributed NMS architecture



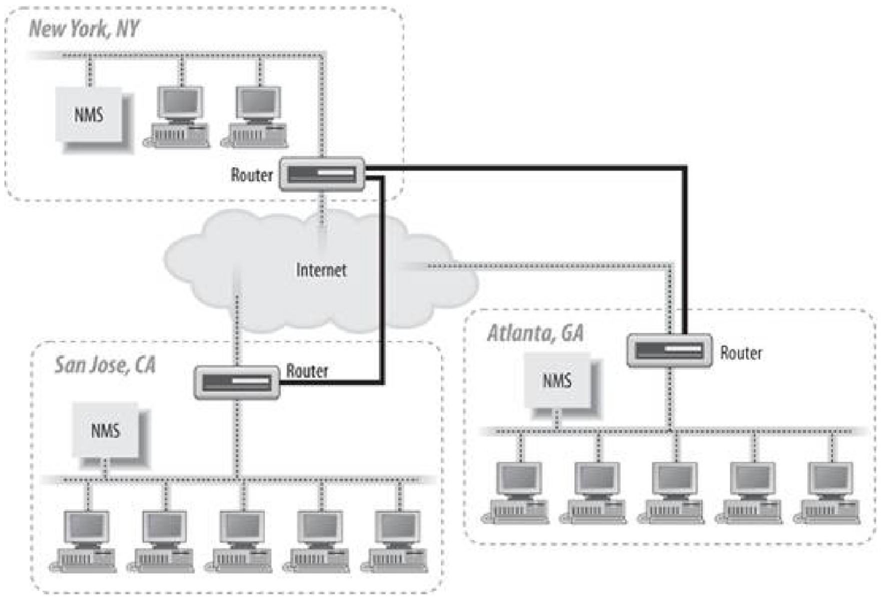
ately. The New York NMS didn't have to use valuable resources to poll the remote network to discover that there was a problem.

The other advantage is that, if the need arises, you can put operations staff in Atlanta and San Jose to manage each of these remote locations. If New York loses connectivity to the Internet, events forwarded from Atlanta or San Jose will not make it to New York. With operations staff in Atlanta and San Jose, and the NMSs at these locations acting in standalone mode, a network outage in New York won't matter. The remote-location staff will continue as if nothing has happened.

Another possibility with this architecture is a hybrid mode: you staff the operations center in New York 24 hours a day, 7 days a week, but you staff Atlanta and San Jose only during business hours. During off-hours, they rely on the NMS and operations staff in New York to notice and handle problems that arise. But during the critical (and busiest) hours of the day, Atlanta and San Jose don't have to burden the New York operators.

Both of the architectures we have discussed use the Internet to send and receive management traffic. This poses several problems, mainly dealing with security and overall reliability. A better solution is to use private links to perform all your network management functions. Figure 4-3 shows how the distributed NMS architecture can be extended to make use of such links.

Figure 4-3. Using private links for network management



Let's say that New York's router is the core router for the network. We establish private (but not necessarily high-speed) links between San Jose and New York, and between New York and Atlanta. This means that San Jose will not only be able to reach New York, but it will also be able to reach Atlanta via New York. Atlanta will use New York to reach San Jose, too. The private links (denoted by thicker router-to-router connections) are primarily devoted to management traffic, though we could put them to other uses. Using private links has the added benefit that our community strings are never sent out over the Internet. The use of private network links for network management works equally well with the single NMS architecture, too. Of course, if your corporate network consists entirely of private links and your Internet connections are devoted to external traffic only, using private links for your management traffic is the proverbial "no-brainer."

One final item worth mentioning is the notion of trap-directed polling . This doesn't really have anything to do with NMS architecture, but it can help to alleviate an NMS's management strain. The idea behind trap-directed polling is simple: the NMS receives a trap and initiates a poll to the device that generated the trap. The goal of this scenario is to determine whether there is indeed a problem with the device while allowing the NMS to ignore (or devote few resources to) the device in normal operation. If an organization relies on this form of management, it should implement it in such a way that non-trap-directed polling is almost done away with. That is, it should avoid polling devices at regular intervals for status information. Instead, the management stations should simply wait to receive a trap before polling a device. This form of management can significantly reduce the resources needed by an NMS to manage a network. However, it has an important disadvantage: traps can get lost in the network and never make it to the NMS. This is a reality of the connectionless nature of UDP and the imperfect nature of networks .

PREV NEXT

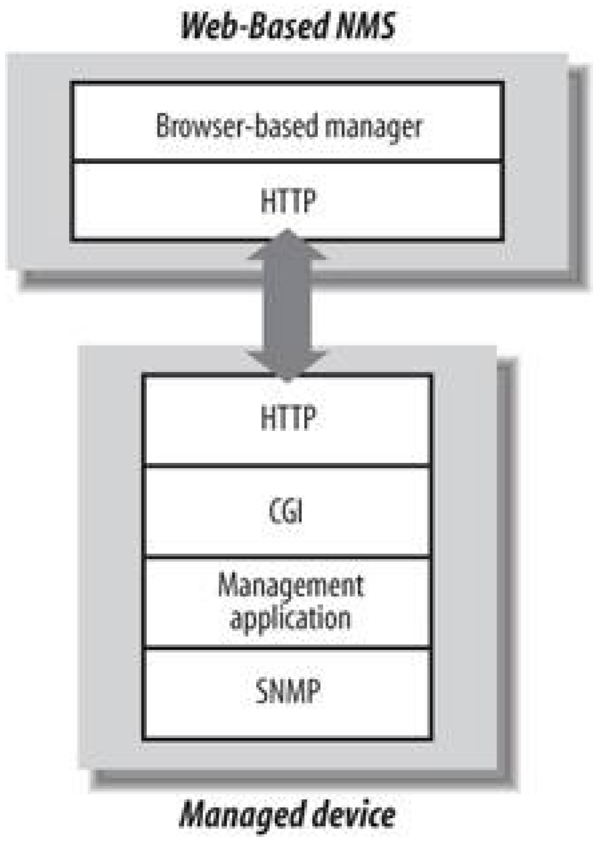
PREV NEXT

4.3. A Look Ahead

Web-based network management entails the use of the HyperText Transfer Protocol (HTTP) and the Common Gateway Interface (CGI) to manage networked entities. It works by embedding a web server in an SNMP-compatible device, along with a CGI engine to convert SNMP-like requests (from a web-based NMS) to actual SNMP operations, and vice versa. Web servers can be embedded into such devices at very low monetary and operating cost.

Figure 4-4 is a simplified diagram of the interaction between a web-based NMS and a managed device. The CGI application bridges the gap between the management application and the SNMP engine. In some cases, the management application can be a collection of Java applets that are downloaded to the web browser and executed on the web-based manager. Current versions of OpenView ship with a web-based GUI. SNMPc also has web-based capabilities. They have a Java client for the network management console and the recently released SNMPc Online, which is a web-based reporting frontend.

Figure 4-4. Web-based network management



Web-based network management could eliminate, or at least reduce, the need for traditional NMS software. NMS software can be expensive to purchase, set up, and maintain. Most of today's major NMS vendors support only a few popular versions of Unix and have only recently begun to support Windows, thus limiting your operating-system choices. With a web-based NMS, however, these two concerns are moot. For the most part, web browsers are free, and Unix, Windows, and Apple platforms all run the popular browsers.

Web-based network management should not be viewed as a panacea, though. It is a good idea, but it will take some time for vendors to embrace this technology and move toward web integration of their existing products. There is also the issue of standardization, or the lack of it. The Web-Based Enterprise Management (WBEM) Initiative addresses this by defining a standard for web-based management. Industry leaders such as Cisco and BMC Software are among the original founders of WBEM. You can learn more about this initiative at the Distributed Management Task Force 's web page, http://www.dmtf.org/standards/wbem.

Another important standard in this area is XML (eXtensible Markup Language) . XML is a markup language used for the interchange of structured data. XML makes use of DTDs (Document Type Definitions) or schemas to specify a document's structure and, in the case of schemas, to validate data. A DTD or schema is similar to an SNMP MIB. XML may be used for network management purposes in the following scenarios:

Using XML in place of standard SNMP over UDP

In environments where UDP traffic isn't permissible, XML can be used as an intermediary application-level protocol. Of course, this requires a mapping layer to translate from XML to SNMP and vice versa.

Converting SNMP MIBs to XML for portability

This has the distinct advantage of allowing languages and systems that support XML parsing to access MIB information. Java is a language that can easily interact with XML.

Using XML for command and control

While this may seem like a perversion of what XML was originally intended for, applications are being written that use XML as an application-level protocol for not only exchanging messages, but also sending control messages.

As new technology comes to the forefront, SNMP researchers, vendors, and users will embrace it whenever it makes sense. This is evidenced by the adoption of SNMPv3 as well as by the use of web technologies for tackling the problems presented by the ever-expanding scope of network management.

PREV NEXT

Chapter 5. Configuring Your NMS

Now that you have picked out some software to use in your environment, it's time to talk about installing and running it. In this chapter, we will look at a few NMS packages in detail. While we list several packages in Appendix F, we will dig into only a few packages here, and we'll use these packages in examples throughout the rest of the book. These examples should allow you to get most other SNMP-based network management packages up and running with very little effort.

PREV NEXT

5.1. HP's OpenView Network Node Manager

Network Node Manager (NNM) is a licensed software product. The package includes a feature called Instant-On that allows you to use the product for a limited time (60 days) while you are waiting for your real license to arrive. During this period, you are restricted to a 250-managed-node license, but the product's capabilities aren't limited in any other way. When you install the product, the Instant-On license is enabled by default.

Check out the OpenView scripts located in OpenView's (normally */opt/OV/bin*). One particularly important group of scripts sets environment variables that allow you to traverse OpenView's directory structure much more easily. These scripts are named *ov.envvars.csh*, *ov.envvars.sh*, etc. (that is,*ov.envvars* the name of the shell you're using). When you run the appropriate script for your shell, it defines environment variables such as $OV_BIN, $OV_MAN, and $OV_TMP, which point to the OpenView *bin*, *man*, and *tmp* directories, respectively. Thus, you can easily go to the directory containing OpenView's manual pages with the command *cd $OV_MAN*. These environment variables are used throughout this book and in all of OpenView's documentation.

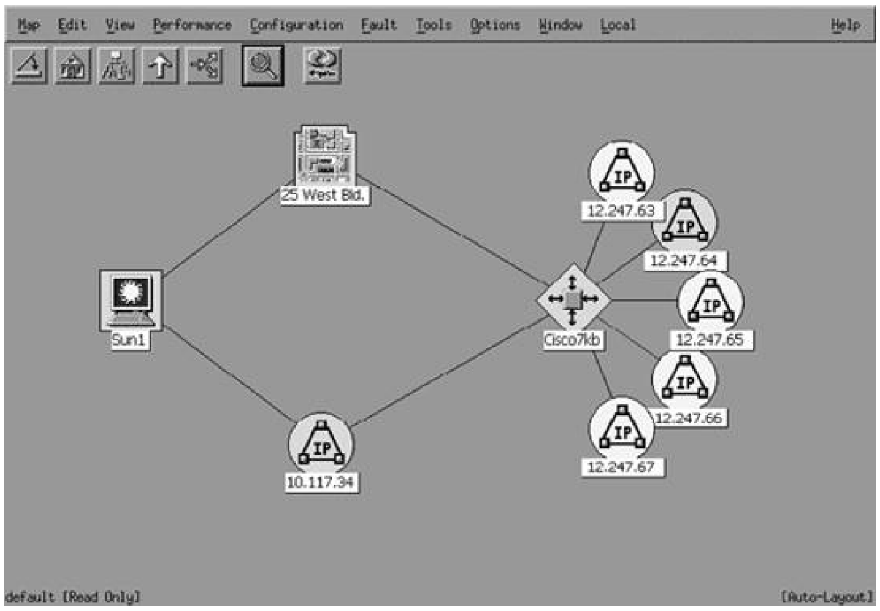
5.1.1. Running NNM

To start the OpenView GUI on a Unix machine, define your DISPLAY environment variable and run the command $OV_BIN/ovw. This starts OpenView's NNM . If your NNM has performed any discovery, the nodes it has found should appear under your Internet (top-level) icon. If you have problems starting NNM, run the command $OV_BIN/ovstatus -c and then $OV_BIN/ovstart or $OV_BIN/ovstop, respectively, to start or stop it. By default, NNM installs the necessary scripts to start its daemons when the machine boots. OpenView will perform all of its functions in the background, even when you aren't running

This means that you do not have to keep a copy of NNM running on your console at all times and you don't have to start it explicitly when your machine reboots.

When the GUI starts, it presents you with a clickable high-level map. This map, called the Root map , provides a top-level view of your network. The map gives you the ability to see your network without having to see every detail at once. If you want more information about any item in the display, whether it's a subnet or an individual node, click on it. You can drill down to see any level of detail you wantfor example, you can look at an interface card on a particular node. The more detail you want, the more you click. Figure 5-1 NNM map.

Figure 5-1. A typical NNM map

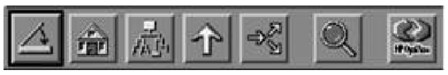


The menu bar (see Figure 5-2) allows you to traverse the map with a bit more ease. You have options such as closing NNM (the leftmost button, which resembles a closing door), going straight to the Home map (the second button from the left, which is, not surprisingly, a house),[*] the Root map (the third-left, a hierarchical diagram), the parent or previous map (the fourth-left button, an up arrow), or the quick navigator (the fifth-left button, a right arrow with two diverging arrows).[†] There is also a magnifying glass button that lets you pan through the map or zoom in on a portion of it.

[*] You can set any map as your Home map. When you've found the map you'd like to use, go to Map ⟶ Submap ⟶ Set This Submap as Home.

[†] This is a special map in which you can place objects that you need to watch frequently. It allows you to access them quickly, without having to find them by searching through the network map.

Figure 5-2. OpenView NNM menu bar



Before you get sick of looking at your newly discovered network, keep in mind that you can add some quick and easy customizations that will transform your hodgepodge of names, numbers, and icons into a coordinated picture of your network.

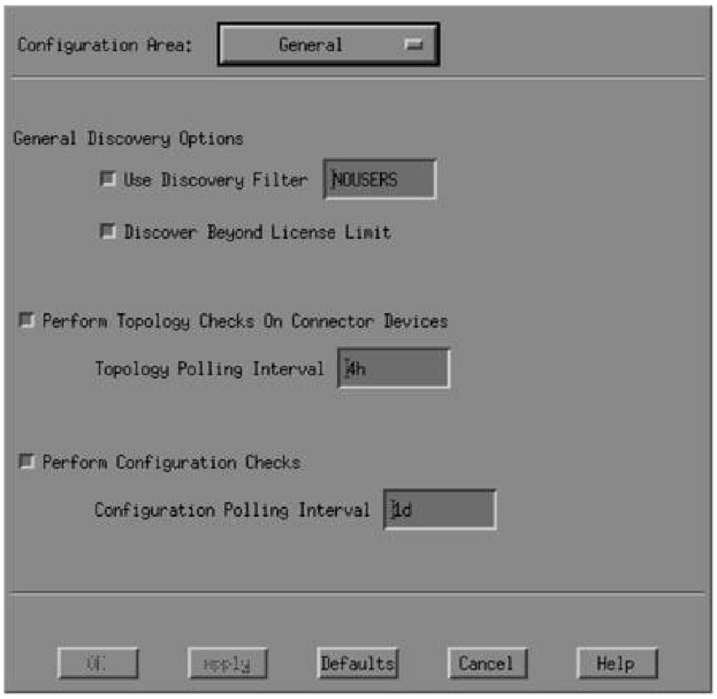
5.1.2. The netmon Process

NNM's daemon process (*netmon*) starts automatically when the system boots and is responsible for discovering nodes on your network, in addition to a few other tasks. In NNM's menu, go to Options ⟶ Network Polling Configurations: IP. A window should appear that looks similar to Figure 5-3.

Figure 5-3 shows the General area of the configuration wizard. The other areas are IP Discovery , Status Polling, and Secondary Failures. The General area allows us to specify a filter (in this example, NOUSERS) that controls the discovery process we might not want to see every device on the network. We discuss the creation of filters in "Using OpenView Filters later in this chapter. We elected to discover beyond the license limit, which means that NNM will discover more objects on our network than our license allows us to manage. "Excess" objects (objects past the license's limit) are placed in an unmanaged state so that you can see them on your maps but can't control them through NNM. This option is useful when your license limits you to a specific number of managed nodes.

The IP Discovery area (Figure 5-4) lets us enable or disable the discovery of IP nodes. Using the "auto adjust" discovery feature allows NNM to figure out how often to probe the network for new devices. The more new devices it finds, the more often it

Figure 5-3. OpenView's General network polling configuration options


Configuration Area: General

General Discovery Options

Use Discovery Filter NOUSERS

Discover Beyond License Limit

Perform Topology Checks On Connector Devices

Topology Polling Interval 4h

Perform Configuration Checks

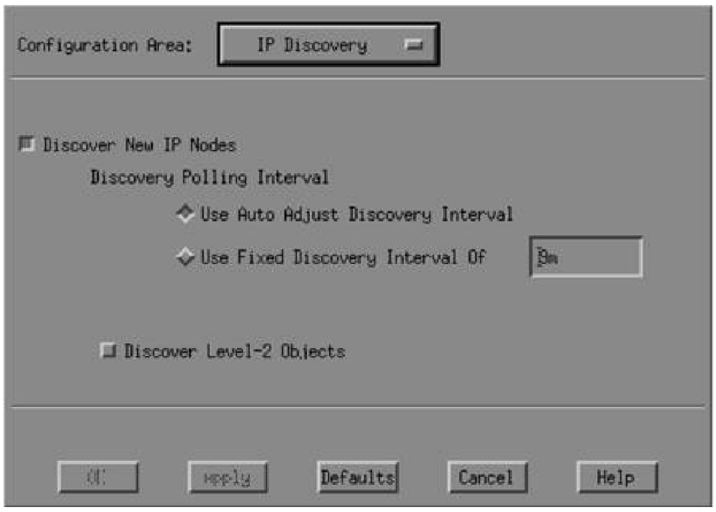
Configuration Polling Interval 1d

polls; if it doesn't find any new devices, it slows down, eventually waiting one day (1d) before checking for any new devices. If you don't like the idea that the discovery interval varies (or perhaps more realistically, if you think that probing the network to find new devices will consume more resources than you like, either on your network management station or on the network itself), you can specify a fixed discovery interval. Finally, the Discover Level-2 Objects button tells NNM to discover and report devices that are at the second layer of the OSI network model. This category includes things such as unmanaged hubs and switches, many AppleTalk devices, and so on.

Figure 5-5 shows the Status Polling configuration area. Here you can turn status polling on or off and delete nodes that have been down or unreachable for a specified length of time. The example in Figure 5-5 is configured to delete nodes after they've been down for one week (1w).

The DHCP polling options are, obviously, especially useful in environments that use DHCP. They allow you to establish a relationship between polling behavior and IP addresses. You can specify a filter that selects addresses that are assigned by DHCP. Then you can specify a time after which *netmon* will delete nonresponding DHCP addresses from its map of your network. If a device is down for the given amount of time, disassociates the node and IP address. The rationale for this behavior is

Figure 5-4. OpenView's IP Discovery network polling configuration options


Configuration Area: IP Discovery

Discover New IP Nodes

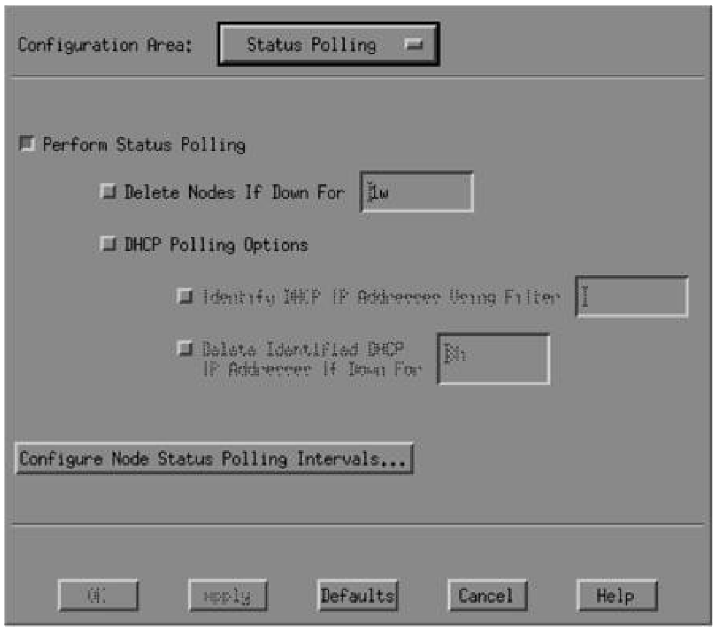
Discovery Polling Interval

Use Auto Adjust Discovery Interval

Use Fixed Discovery Interval Of 8m

Discover Level-2 Objects

Figure 5-5. OpenView's Status Polling network polling configuration options



Configuration Area: Status Polling

Perform Status Polling

Delete Nodes If Down For 1w

DHCP Polling Options

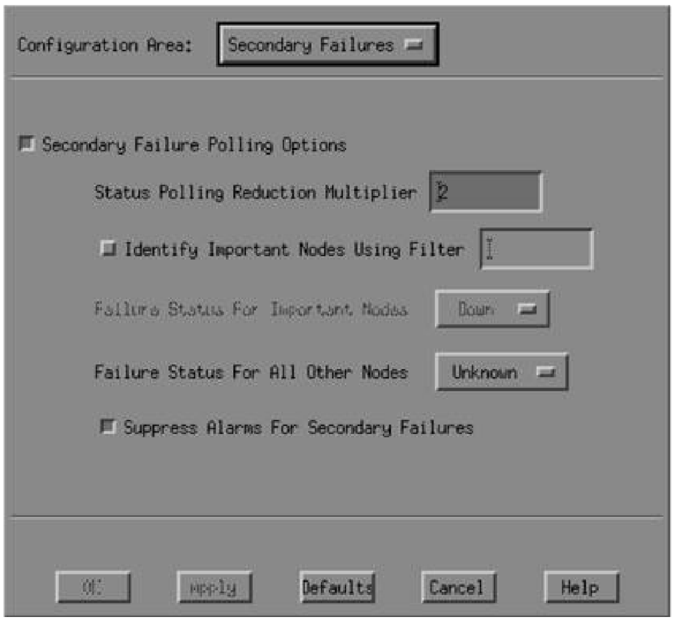
Identify DHCP IP Addresses Using Filter

Delete Identified DHCP IP Addresses If Down For 8h

simple: in a DHCP environment, the disappearance of an IP address often means that the node has received a new IP address from a DHCP server. In that case, continuing to poll the old address is a waste of effort and is possibly even misleading, since the address may be reassigned to a different host.

Finally, the Secondary Failures configuration area shown in Figure 5-6 poller how to react when it sees a secondary failure. This occurs when a node beyond a failed device is unreachablefor example, when a router goes down, making the file server that is connected via one of the router's interfaces unreachable. In this configuration area, you can state whether to show alarms for secondary failures or suppress them. If you choose to suppress them, you can set up a filter that identifies important nodes in your network that should not be suppressed even if they are deemed secondary failures.

Figure 5-6. OpenView's Secondary Failures network polling configuration options



Configuration Area: Secondary Failures

Secondary Failure Polling Options

Status Polling Reduction Multiplier: 2

Identify Important Nodes Using Filter

Failure Status For Important Nodes: Down

Failure Status For All Other Nodes: Unknown

Suppress Alarms For Secondary Failures

Once your map is up, you may notice that nothing is getting discovered. Initially, won't discover anything beyond the network segment to which your NMS is attached. If your NMS has an IP address of 24.92.32.12, you will not discover your devices on 123.67.34.0. NNM finds adjacent routers and their segments, as long as they are SNMP compatible, and places them in an unmanaged (tan colored) state on the map.[*] This means that anything in and under that icon will not be polled or discovered. Selecting the icon and going to Edit Manage Objects tells NNM to begin managing this network and allows discovering nodes within it. You can quit managing nodes at any time by clicking on UnManage instead of Manage.

[*] In NNM, go to Help ⟶ Display Legend for a list of icons and their colors.

If your routers do not show any adjacent networks, you should try testing ⟶ Test IP/TCP/SNMP. Add the name of your router, click Restart, and see what kind of results you get back. If you get a message that says "OK except for SNMP," read as well as the next section in this chapter, which discusses setting up the default community names within OpenView .

netmon also allows you to specify a seed file that helps it to discover objects faster. The seed file contains individual IP addresses , IP address ranges, or domain names that narrow the scope of hosts that are discovered. You can create the seed file with any text editorjust put one address or hostname on each line. Placing the addresses of your gateways in the seed file sometimes makes the most sense since gateways maintain ARP tables for your network. *netmon* will subsequently discover all the other nodes on your network, thus freeing you from having to add all your hosts to the seed file. For more useful information, see the documentation for the -s switch to *netmon* and the Local Registration Files (LRFs).

NNM has another utility, called *loadhosts* , that lets you add nodes to the map one at a time. Here is an example of how you can add hosts, in a sort of freeform mode, to the OpenView map. Note the use of the -m option, which sets the subnet to 255.255.255.0:

```
$ loadhosts -m 255.255.255.0
10.1.1.12 gwrouter1
```

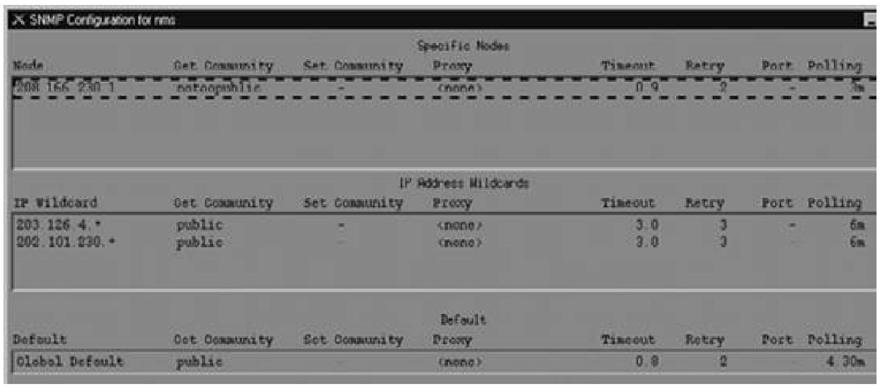
Once you have finished adding as many nodes as you'd like, press Ctrl-D to exit the command.

5.1.3. Configuring Polling Intervals

The SNMP Configuration page is located off the main screen under Options Configuration. A window similar to the one in Figure 5-7 should appear. This window has four sections: Specific Nodes, IP Address Wildcards, Default, and the entry area (cropped in this example). Each section contains the same general areas: Node or IP Address, Get Community, Set Community, Proxy (if any), Timeout, Retry, Port, and Polling. The Default area, which is unfortunately at the bottom of the screen, sets up the default behavior for SNMP on your networkthat is, the behavior (community strings, etc.) for all hosts that aren't listed as "specific nodes" or that match one of the wildcards. The Specific Nodes section allows you to specify exceptions on a per-node basis. IP Wildcard allows you to configure properties for a range of addresses. This is especially useful if you have networks that have different get and set community names.[*] All areas allow you to specify a Timeout in seconds and a Retry value. The Port field gives you the option of inserting a different port number (the default port is 161). Polling is the frequency at which you would like to poll your nodes.

[*] These community names are used in different places throughout NNM. For example, when polling an object with *xnmbrowser*, you won't need to enter (or remember) the community string if it (or its network) is defined in the SNMP configurations.

Figure 5-7. OpenView's SNMP Configuration page



SNMP Configuration for nms

Specific Nodes

Node	Get Community	Set Community	Proxy	Timeout	Retry	Port	Polling
208.166.230.1	[illegible]	-	<none>	0.9	2	-	5m

IP Address Wildcards

IP Wildcard	Get Community	Set Community	Proxy	Timeout	Retry	Port	Polling
203.126.4.*	public	-	<none>	3.0	3	-	6m
202.101.230.*	public	-	<none>	3.0	3	-	6m

Default

Default	Get Community	Set Community	Proxy	Timeout	Retry	Port	Polling
Global Default	public	-	<none>	0.8	2	-	4.30m

It's important to understand how timeouts and retries work. If we look at Specific Nodes, we see a Timeout of .9 seconds and a Retry of 2 for 208.166.230.1. If OpenView doesn't get a response within .9 seconds, it tries again (the first retry) and waits 1.8 seconds. If it still doesn't get anything back, it doubles the timeout period again to 3.6 seconds (the second retry); if it still doesn't get anything back, it declares the node unreachable and paints it red on NNM's map. With these Timeout and Retry values, it takes about 6 seconds to identify an unreachable node.

Imagine what would happen if we had a Timeout of 4 seconds and a Retry of 5. By the fifth try, we would be waiting 128 seconds, and the total process would take 252 seconds. That's more than four minutes! For a mission-critical device, four minutes can be a long time for a failure to go unnoticed.

This example shows that you must be very careful about your Timeout and Retry settingsparticularly in the Default area, because these settings apply to most of your network. Setting your Timeout and Retry too high and your Polling periods too low will make *netmon* fall behind; it will be time to start over before the poller has worked through all your devices.[*] This is a frequent problem when you have many nodes, slow networks, small polling times, and high numbers for Timeout and Retry.[†] Once a system falls behind, it will take a long time to discover problems with the devices it is currently monitoring, as well as to discover new devices. In some cases, NNM may not discover problems with downed devices at all! If your Timeout and Retry values are set inappropriately, you won't be able to find problems and you will be unable to respond to outages.

[*] Keep in mind that most of NNM's map is polled using regula*ping*
SNMP.

[†] Check the manpage for*netmon* for the -a switch, especially around -a12. You can try to execute *netmon* with an -a \ ?, which lists all the valid -a options. If you see any negative numbers in *netmon.trace* after running netmon -a12, your system is running behind.

Falling behind can be very frustrating. We recommend starting your Polling period very high and working your way down until you feel comfortable. Ten to twenty minutes is a good starting point for the Polling period. During your initial testing phase, you can always set a wildcard range for your test servers, etc.

5.1.4. A Few Words About NNM Map Colors

By now, discovery should be taking place, and you should be starting to see some new objects appear on your map. You should see a correlation between the colors
objects and the colors in NNM's Event Categories (see Chapter 9 for more about Event Categories). If a device is reachable via ping, its color will be green. If the device cannot be reached, it will turn red. If something "underneath" the device fails, the device will become off-green, indicating that the device itself is OK, but something underneath it has a nonnormal status. For example, a router may be working, but a web server on the LAN behind it may have failed. The status source for an object like this is Compound or Propagated. (The other types of status source are Symbol and Object.) The Compound status source is a great way to see if there is a problem at a lower level while still keeping an eye on the big picture. It alerts you to the problem and allows you to start drilling down until you reach the object that is under duress.

It's always fun to shut off or unplug a machine and watch its icon turn red on the map. This can be a great way to demonstrate the value of the new management system to your boss. You can also learn how to cheat and make OpenView miss a device, even though it was unplugged. With a relatively long polling interval, it's easy to unplug a device and plug it back in before OpenView has a chance to notice that the device isn't there. By the time OpenView gets around to it, the node is back up and looks fine. Long polling intervals make it easy to miss such temporary failures. Lower polling intervals make it less likely that OpenView will miss something, but more likely that *netmon* will fall behind, and in turn miss other failures. Take small steps so as not to crash or overload *netmon* or your network.

5.1.5. Using OpenView Filters

Your map may include some devices you don't need, want, or care about. For example, you may not want to poll or manage users' PCs, particularly if you have many users and a limited license. It may be worthwhile for you to ignore these user devices to open more slots for managing servers, routers, switches, and other more important devices.
filtering mechanism that allows you to control precisely which devices you manage. It lets you filter out unwanted devices, cleans up your maps , and can reduce the amount of management traffic on your network.

In this book, we warn you repeatedly that polling your network the wrong way can generate huge amounts of management traffic. This happens when people or programs use default polling intervals that are too fast for the network or the devices on the network to handle. For example, a management system might poll every node in your 10.1.0.0 networkconceivably thousands of themevery two minutes. The poll may consist of SNMP get or set requests, simple pings, or both. OpenView's NNM uses a combination of these to determine if a node is up and running. Filtering saves you (and your management) the trouble of having to pick through a lot of useless nodes and reduces the load on your network. Using a filter allows you to keep the critical nodes on your network in view. It allows you to poll the devices you care about and ignore the devices you don't care about. The last thing you want is to receive notification each time a user turns off his PC when he leaves for the night.

Filters also streamline network management by letting you exclude DHCP users from network discovery and polling. DHCP and BOOTP are used in many environments to manage large IP address pools. While these protocols are useful, they can make network management a nightmare, since it's often hard to figure out what's going on when addresses are being assigned, deallocated, and recycled.

In our environment, we use DHCP only for our users. All servers and printers have hardcoded IP addresses. With our setup, we can specify all the DHCP clients and then state that we want everything *but* these clients in our discovery, maps, etc. The following example should get most users up and running with some pretty good filtering. Take some time to review OpenView's "A Guide to Scalability and Distribution for HP OpenView Network Node Manager" manual for more in-depth information on filtering.[*]

> [*] This manual is available at http://ovweb.external.hp.com/ovnsmdps/pdf/j1240-90001.pdf.

The default filter file, which is located in *$OV_CONF/C*, is broken up into three sections:

- Sets
- Filters
- FilterExpressions

In addition, lines that begin with // are comments. // comments can appear anywhere; some of the other statements have their own comment fields built in.

Sets allow you to place individual nodes into a group. This can be useful if you want to separate users based on their geographic locations, for example. You can then use these groups or any combination of IP addresses to specify your Filters, which are also grouped by name. You then can take all of these groupings and combine them into FilterExpressions. If this seems a bit confusing, it is! Filters can be very confusing, especially when you add complex syntax and not-so-logical logic (&&, ||, etc.). The basic syntax for defining Sets, Filters, and FilterExpressions looks like this:

```
name "comments or description" { contents }
```

Every definition contains a name, followed by comments that appear in double quotes and then the command surrounded by brackets. Our default filter,[*] named *$OV_CONF/C* and looks like this:

> [*] Your filter, if right out of the box, will look much different. The one shown here is trimmed to ease the pains of writing a filter.

```
// lines that begin with // are considered COMMENTS and are ignored!
// Beginning of MyCompanyName Filters
```

```
Sets {

   dialupusers "DialUp Users" { "dialup100", " dialup101", \
             " dialup102" }
}

Filters {

   ALLIPRouters "All IP Routers" { isRouter }

   SinatraUsers "All Users in the Sinatra Plant" { \
      ("IP Address" ~ 199.127.4.50-254) || \
      ("IP Address" ~ 199.127.5.50-254) || \
      ("IP Address" ~ 199.127.6.50-254) }

   MarkelUsers "All Users in the Markel Plant" { \
      ("IP Address" ~ 172.247.63.17-42) }

   DialAccess "All DialAccess Users" { "IP Hostname" in dialupusers }
}

FilterExpressions
{
   ALLUSERS "All Users" { SinatraUsers || MarkelUsers || DialAccess }

   NOUSERS "No Users " { !ALLUSERS }
}
```

Now let's break down this file into pieces to see what it does.

5.1.5.1. Sets

First, we defined a Set[†] called `dialupusers` containing the hostnames (from DNS) that our dial-up users will receive when they dial into our facility. These are perfect examples of things we don't want to manage or monitor in our OpenView environment.

[†] These Sets have nothing to do with the*snmpset* operation with which we have become familiar.

5.1.5.2. Filters

The Filters section is the only required section. We defined four filters: MarkelUsers, and DialAccess. The first filter says to discover nodes that have the field value OpenView can set the object attribute for a managed device to values such as isNode, etc.[*] These attributes can be used in Filter expressions to make it easier to filter on groups of managed objects, as opposed to IP address ranges, for example.

[*] Check out the*$OV_FIELDS* folder for a list of fields.

The next two filters specify IP address ranges. The SinatraUsers filter is the more complex of the two. In it, we specify three IP address ranges, each separated by logical OR symbols (The first range (("IP Address" ~ 199.127.6.50-254) says that if the IP address is in the range 199.127.6.50-199.127.6.254, filter it and ignore it. If it's not in this range, the filter looks at the next range to see if it's in that one. If it's not, the filter looks at the final IP range. If the IP address isn't in any of the three ranges, the filter allows it to be discovered and subsequently managed by NNM. Other logical operators should be familiar to most programmers: && represents a logical AND, and ! represents a logical NOT.

The final filter, DialAccess, allows us to exclude all systems that have a hostname listed in the dialupusers Set, which was defined at the beginning of the file.

5.1.5.3. FilterExpressions

The next section, FilterExpressions, allows us to combine the filters we have previously defined with additional logic. You can use a FilterExpression anywhere you would use a Filter. Think of it like this: you create complex expressions using Filters, which in turn can use Sets in the contents parts of their expressions. You can then use FilterExpressions to create simpler yet more robust expressions. In our case, we take all the filters from above and place them into a FilterExpression called ALLUSERS. Since we want our NNM map to contain nonuser devices, we then define a group called NOUSERS and tell it to ignore all user-type devices with the command !ALLUSERS. As you can see, FilterExpressions can also aid in making things more readable. When you have finished setting up your filter file, use the $OV_BIN/ovfiltercheck program to check your new filters' syntax. If there are any problems, it will let you know so that you can fix them.

Now that we have our filters defined, we can apply them by using the ovtopofix command.

If you want to remove nodes from your map, use $OV_BIN/ovtopofix -f FILTER_NAME. Let's say that someone created a new DHCP scope without telling you and suddenly all the new users are now on the map. You can edit the *filters* file, create a new group with the IP address range of the new DHCP scope, add it to the ALLUSERS FilterExpression, and run ovfiltercheck. If there are no errors, run $OV_BIN/ovtopofix -f NOUSERS to update the map on the fly. Then stop and restart *netmon*otherwise, it will keep discovering these unwanted nodes using the old filter. We run ovtopofix every month or so to take out some random nodes.

5.1.6. Loading MIBs into OpenView

Before you continue exploring OpenView's NNM, take time to load some vendor-specific MIBs.[*] This will help you later on when you start interacting (polling, graphing, etc.) more with SNMP-compatible devices. Go to Options → Load/Unload MIBs: SNMP. This presents you with a window in which you can add vendor-specific MIBs to your database. Alternatively, you can run the command $OV_BIN/xnmloadmib and bypass having to go through NNM directly.

[*] Some platforms and environments refer to loading a MIB as *compiling*

That's the end of our brief tour of OpenView configuration . It's impossible to provide a complete introduction to configuring OpenView in this chapter, so we tried to provide a survey of the most important aspects of getting it running. There can be no substitute for the documentation and manual pages that come with the product itself.

PREV NEXT

PREV NEXT

5.2. Castle Rock's SNMPc Enterprise Edition

We'll end the chapter with a brief discussion of Castle Rock's SNMPc, Version 7.0, which runs on Microsoft Windows.[†] SNMPc is a simpler product than OpenView in many respects. However, even though it's simpler, it's far from featureless. It's also cheaper than OpenView, which makes it ideal for shops that don't have a lot of money to spend on an NMS platform but need the support and backing that a commercial product provides.

[†] SNMPc runs on Windows Server 2003, Windows 2000, Windows XP, and Windows NT. The WorkGroup edition and console also run on Windows ME and Windows 98.

Installation of SNMPc is straightforward. The installer asks for the license number and a discovery seed device. The seed device is similar to a seed file for OpenView's
the case of SNMPc, we recommend giving it the IP address (or hostname) of your gateway since this device can be used to discover other segments of your network. The installer gives you the option (checkbox) of turning on or off the discovery during the initial start. Bigger networks might opt to deactivate the initial discovery to prevent a flood of requests (and maybe auth failures).

5.2.1. SNMPc's Map

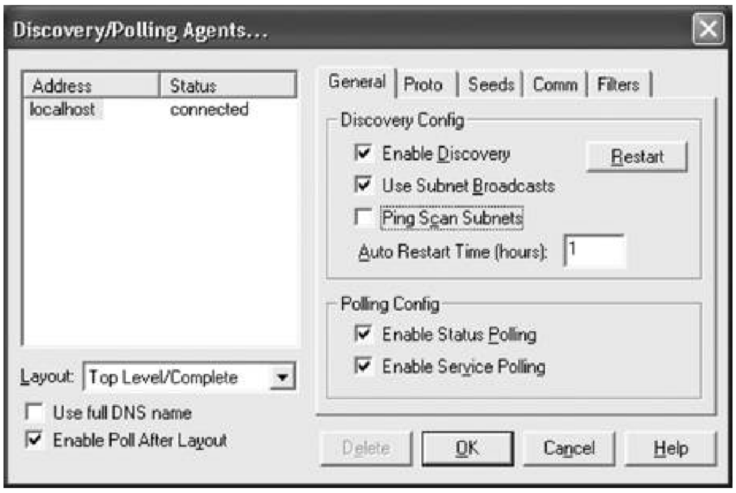
Once SNMPc is up and running, you will see any devices it has discovered in the Root map view. Figure 5-8 shows the main button bar. The far-right button (the house) gets you to the highest level on the map. The zooming tools allow you to pan in and out of the map, increasing or decreasing the amount of detail it shows. You can also reach the Root submap by selecting View ⟶ Map View ⟶ Root Submap.

Figure 5-8. SNMPc main button bar

5.2.2. Discovery and Filters

Once you are done playing around with your maps, it's time to start tuning your polling parameters. Go to Config ⟶ Discovery/Polling. This should bring up a menu that looks like Figure 5-9. Looking at the menu tabs, it's easy to tell that you will be able to configure your Seeds, Communities, Filters, and TCP service polling (Proto) here. SNMPc filters equivalent to OpenView filters, but they are much simpler.

Figure 5-9. SNMPc Discovery/Polling Agents menu



Discovery/Polling Agents...

Address	Status
localhost	connected

Layout: Top Level/Complete

- [ ] Use full DNS name
- [x] Enable Poll After Layout

General | Proto | Seeds | Comm | Filters

Discovery Config
- [x] Enable Discovery
- [x] Use Subnet Broadcasts
- [ ] Ping Scan Subnets
- Auto Restart Time (hours): 1

Polling Config
- [x] Enable Status Polling
- [x] Enable Service Polling

The General tab lets you control SNMPc's polling and discovery behavior. The checkbox for enabling and disabling discovery is self-explanatory. The Enable Status Polling checkbox determines whether SNMPc will ping the nodes on your network periodically to determine whether they are responding. By default, all nodes are polled every 10 to 30 seconds. To change these default values, you can either edit the properties of each device (one by one), select and highlight multiple devices (using your Ctrl key), or use the object selection tool. You can bring up this tool by going to View → Selection Tool. The Discover Ping Nodes checkbox (under the Proto tab) lets you specify if you want to discover devices that have an IP or IPX entity but do not have an SNMP agent. SNMPc will also check whether a device supports various protocols such as SMTP, HTTP, etc. This feature allows you to set up custom menu items based on what services the device is running. The Find TCP Ports section of the Proto tab lets you specify the protocols for which SNMPc will test.

The Seeds tab allows you to specify SNMP devices that will help the discovery process along. This tab allows you to specify more than one seed IP address. (Remember that you're asked for a seed address device when you install the product.)

The Comm tab lets you specify the community strings for your network. You can specify multiple community names; SNMPc will try the different community names when discovering your nodes. Once SNMPc figures out which community is correct for a given device, it inserts the community string in the Get Community attribute for that particular device. This simply means the newly discovered device will be saved with its community string.

The final tab, Filters, allows you to exclude certain IP addresses from being discovered. You can specify individual addresses or use an asterisk (*) as a wildcard to specify entire networks.

5.2.3. Discovery Run-Through

Let's have SNMPc discover a small home network. First, I configure the discovery engine to find non-SNMP devices.

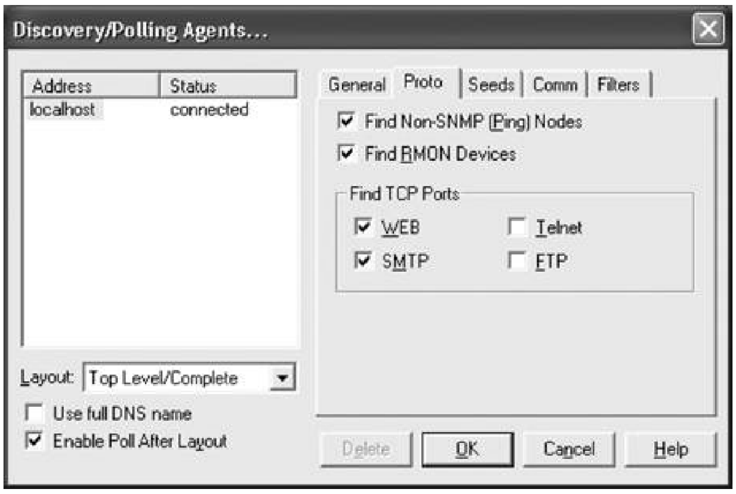
Figure 5-10 shows that the Find Non-SNMP (Ping) Nodes checkbox is selected. This setting will find devices on your network that are up but are not running an SNMP agent and can be helpful in getting a general map of your entire network. When you click OK, SNMPc does its thing. After a few moments, SNMPc creates a map that looks like Figure 5-11

In this example, SNMPc has found my small home network, 192.168.1. If I double-click on the icon on the map, I see the actual devices on my home network, as shown in

I have two machines: a Linux server and a laptop. Figure 5-13 shows a closer view of these discovered devices.

Notice that the LinuxServer icon has the word *snmp* as part of the icon. This means it responded to SNMP polls. The loanera22p icon, however, is not running an SNMP agent, yet the discovery engine found it and placed the word *icmp* to denote that the machine is up and running but is not responding to SNMP polls.

Figure 5-10. Finding non-SNMP devices



Discovery/Polling Agents...

Address	Status
localhost	connected

Layout: Top Level/Complete

- [ ] Use full DNS name
- [✓] Enable Poll After Layout

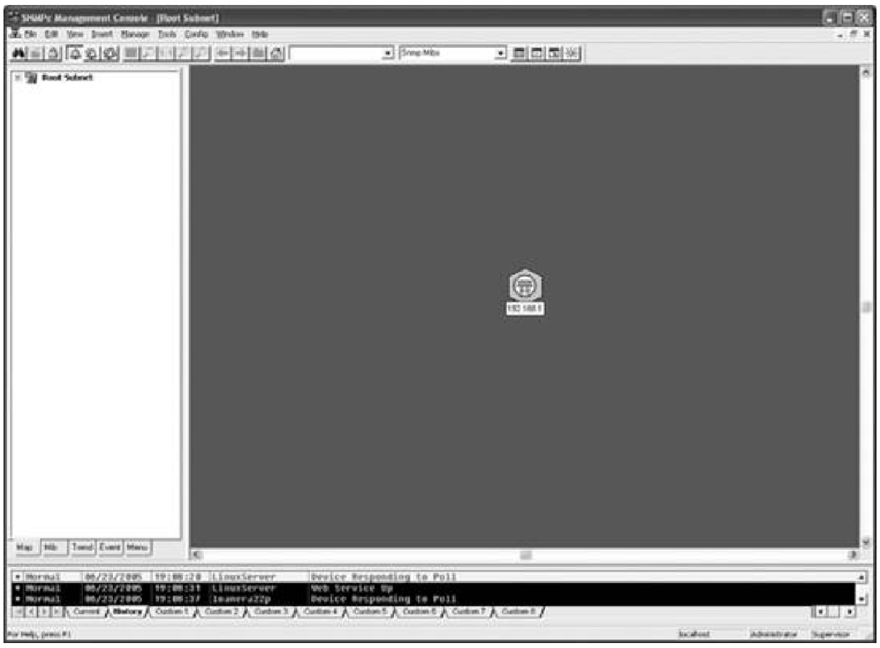
Proto tab:

- [✓] Find Non-SNMP (Ping) Nodes
- [✓] Find RMON Devices

Find TCP Ports

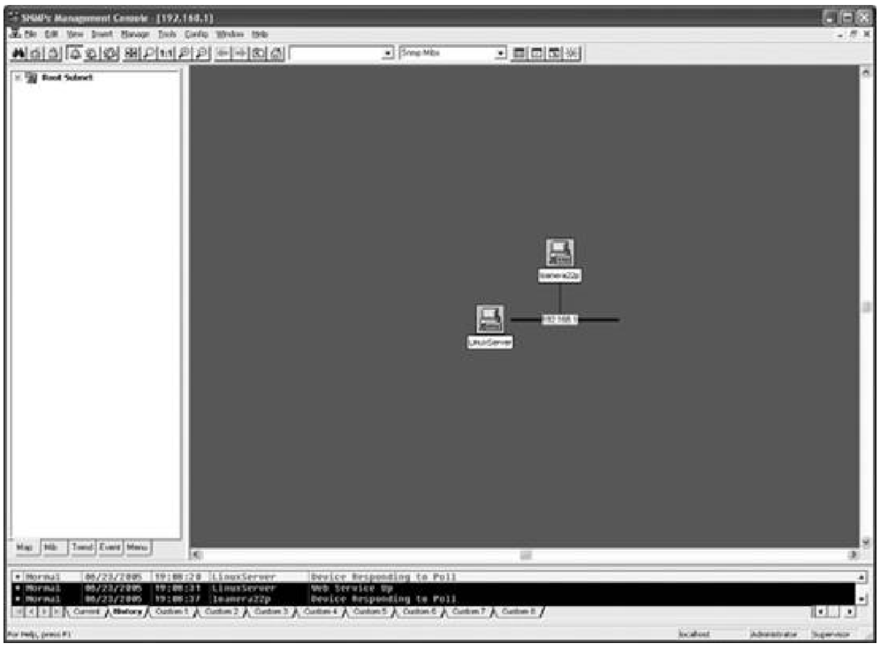
- [✓] WEB
- [ ] Telnet
- [✓] SMTP
- [ ] FTP

Figure 5-11. Root network discovered


SNMPc Management Console - [Root Subnet]
File Edit View Insert Manage Tools Config Window Help
Snmp Mibs
Root Subnet
192.168.1
Map Mib Trend Event Menu
Current History Custom 1 Custom 2 Custom 3 Custom 4 Custom 5 Custom 6 Custom 7 Custom 8
For Help, press F1
localhost Administrator Supervisor

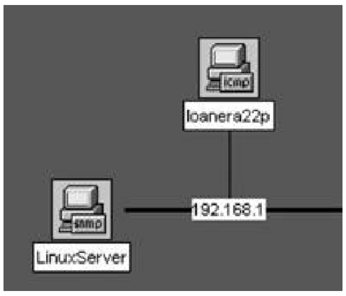
```
• Normal   08/23/2005  19:08:28  LinuxServer   Device Responding to Poll
• Normal   08/23/2005  19:08:31  LinuxServer   Web Service Up
• Normal   08/23/2005  19:08:37  leanerz22p    Device Responding to Poll
```

Figure 5-12. The devices on the 192.168.1 network


SNMPc Management Console - [192.168.1]
File Edit View Insert Manage Tools Config Window Help
Snmp Mibs
Root Subnet
loanera22p
192.168.1
LinuxServer
Map Mib Trend Event Menu
Current History Custom 1 Custom 2 Custom 3 Custom 4 Custom 5 Custom 6 Custom 7 Custom 8
For Help, press F1
localhost Administrator Supervisor

```
Normal   08/23/2005   19:08:28   LinuxServer   Device Responding to Poll
Normal   08/23/2005   19:08:31   LinuxServer   Web Service Up
Normal   08/23/2005   19:08:37   loanera22p    Device Responding to Poll
```

Figure 5-13. Discovered devices

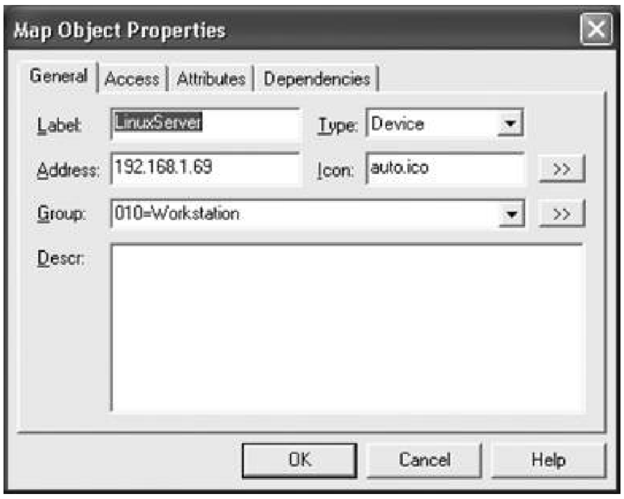


5.2.4. Configuring SNMPv3

Not only can SNMPc poll SNMPv1 and SNMPv2 devices, but it can also poll SNMPv3 devices. This is a very nice feature and further shows that Castle Rock is committed to producing a full-featured network management platform. Let's briefly look at how you would configure SNMPv3-specific parameters.

Right-click on a device and select Properties to display a window like that shown in 5-14.

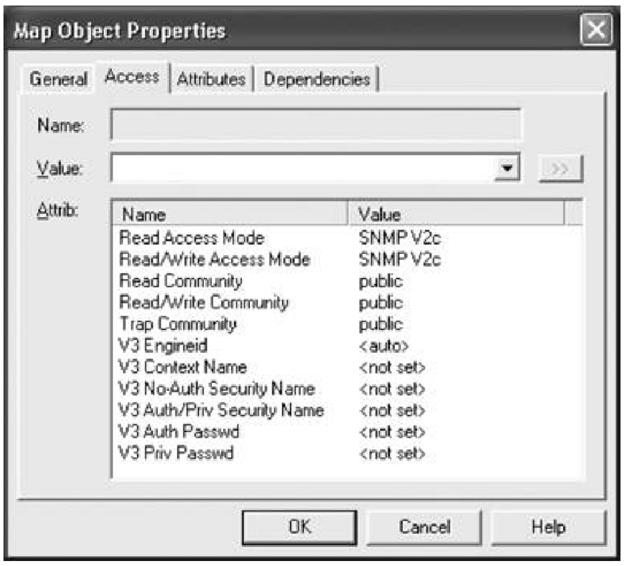
Figure 5-14. Map Object Properties



Label: LinuxServer
Type: Device
Address: 192.168.1.69
Icon: auto.ico
Group: 010=Workstation
Descr:

Select the Access tab to see the SNMP configuration parameters, in Figure 5-15

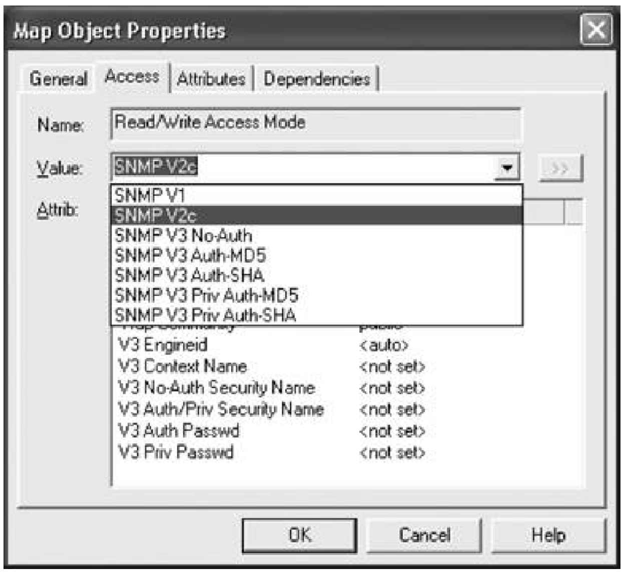
Figure 5-15. SNMP parameters



Name	Value
Read Access Mode	SNMP V2c
Read/Write Access Mode	SNMP V2c
Read Community	public
Read/Write Community	public
Trap Community	public
V3 Engineid	<auto>
V3 Context Name	<not set>
V3 No-Auth Security Name	<not set>
V3 Auth/Priv Security Name	<not set>
V3 Auth Passwd	<not set>
V3 Priv Passwd	<not set>

As you can see, there are several attributes to choose from. If you select either Read/Access Mode or Read/Write Access Mode and expand the Value drop-down box, you will see the different modes available to you, as in Figure 5-16.

Figure 5-16. Access modes



Name: Read/Write Access Mode

Value: SNMP V2c

Attrib:

SNMP V1
SNMP V2c
SNMP V3 No-Auth
SNMP V3 Auth-MD5
SNMP V3 Auth-SHA
SNMP V3 Priv Auth-MD5
SNMP V3 Priv Auth-SHA

V3 Engineid	<auto>
V3 Context Name	<not set>
V3 No-Auth Security Name	<not set>
V3 Auth/Priv Security Name	<not set>
V3 Auth Passwd	<not set>
V3 Priv Passwd	<not set>

Figure 5-16 shows the various SNMPv1, SNMPv2, and SNMPv3 access modes available to you. SNMPc supports only DES for privacy. If my SNMP agent used MD5 for authentication and DES for privacy, I would configure the map object as shown in Figure 5-17

Note that Read Access Mode and Read/Write Access Mode are both SNMP V3 Priv Auth-MD5. This means SNMPc will use MD5 for authentication. Since SNMPc supports only DES for privacy, it is implied that it will use this protocol for encrypting the data. My SNMPv3 username is kjs, the authentication password ismypassword, and the privacy password is myotherpassword.[*]

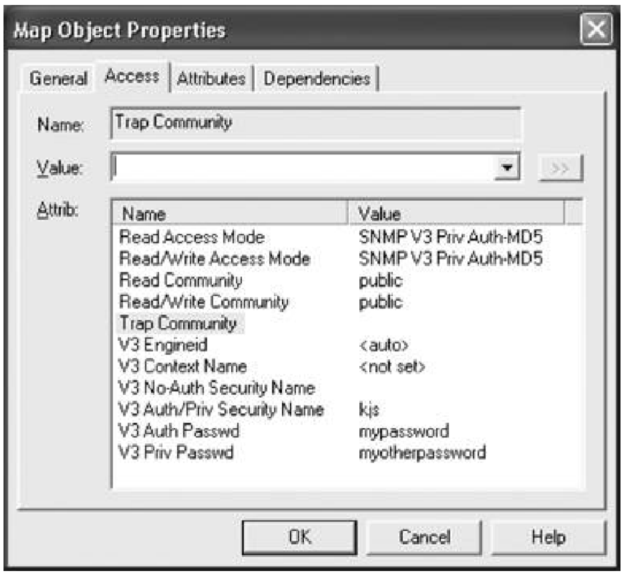
[*] Of course, these are just passwords for this example. In real life, one would never use such lame passwords.

5.2.5. Loading MIBs into SNMPc

Like any reasonably comprehensive network management product, SNMPc can load new MIBs. According to the SNMPc documentation, you place new MIB source files in the

\snmpcnt\mibfiles directory on the server computer. Note that on my system, the full path for the MIB files is *C:\Program Files\SNMPc Network Manager\mibfiles*, so check both locations.

Figure 5-17. SNMPv3 configuration



Name: Trap Community

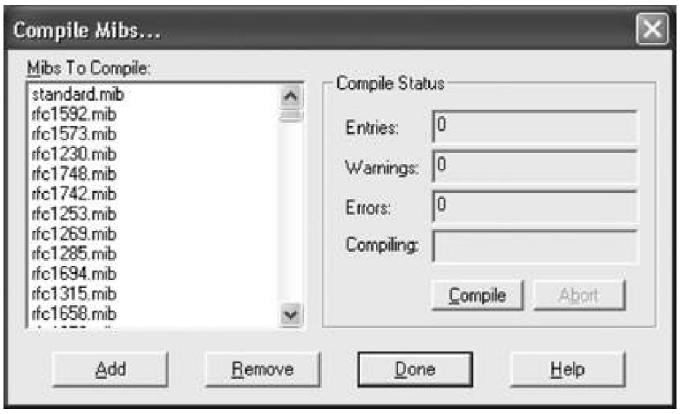
Value:

Attrib:

Name	Value
Read Access Mode	SNMP V3 Priv Auth-MD5
Read/Write Access Mode	SNMP V3 Priv Auth-MD5
Read Community	public
Read/Write Community	public
Trap Community	
V3 Engineid	<auto>
V3 Context Name	<not set>
V3 No-Auth Security Name	
V3 Auth/Priv Security Name	kjs
V3 Auth Passwd	mypassword
V3 Priv Passwd	myotherpassword

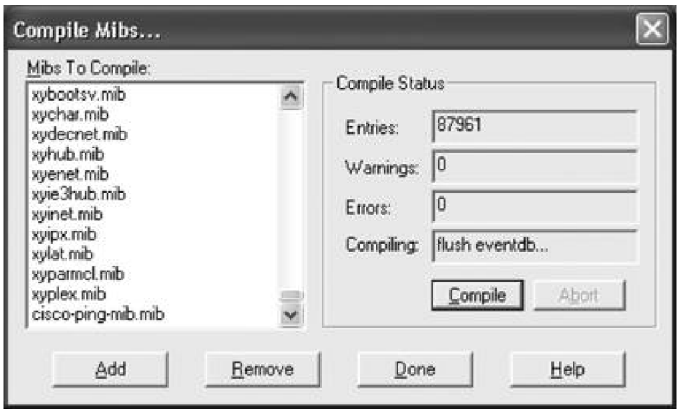
Once you have copied your MIB file to this directory (make sure it has a select Config ⟶ Mib Database from the menu bar to display the Compile Mibs window shown in Figure 5-18.

Figure 5-18. Compile Mibs window



Click Add to find the MIB file you want to add. This brings up another window where you can find the MIB you want to add in the list and click OK. You will now be back at the Compile Mibs window. Click Compile to compile all the MIBs, including the one you just added. This may take a little bit of time. Once it's done and there are no errors, you should see a window similar to Figure 5-19.

Figure 5-19. Completion of MIB compile



If you want to learn more about adding MIBs, click on the Help button.

SNMPc is a compact NMS that provides some added features, such as trend reporting. A thorough treatment of its installation is beyond the scope of this book. The online help system that comes with SNMPc is very good, and we recommend that you take full advantage of it.

PREV NEXT

PREV NEXT

Chapter 6. Configuring SNMP Agents

By this time, you should understand what an SNMP agent is: it's nothing more than software that lives on the device you want to monitor. It responds to requests from the NMS and generates traps. This chapter discusses how to configure agents. It starts by defining some standard configuration parameters that are common to all SNMP agents, and then goes into some advanced parameters you might run into when configuring your equipment. The bulk of this chapter walks through the configuration for a number of common devices, paying attention to security issues.

PREV NEXT

6.1. Parameter Settings

All SNMP devices share the following common configurable parameters:

- *sysLocation*
- *sysContact*
- *sysName*
- Read-write and read-only access community strings (and frequently, a trap community string)
- Trap destination

sysLocation is the physical location for the device being monitored. Its definition in RFC 1213 is:

```
sysLocation OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "The physical location of this node (e.g., 'telephone closet,
         3rd floor')."
    ::= { system 6 }
```

As you can see, its SYNTAX is DisplayString, which means it can be an ASCII string of characters; its size is declared to be, at most, 255 characters. This particular object is useful for determining where a device is located. This kind of practical information is essential in a large network, particularly if it's spread over a wide area. If you have a misbehaving switch, it's very convenient to be able to look up the switch's physical location. Unfortunately, *sysLocation* frequently isn't set when the device is installed and even more often isn't changed when the device is moved. Unreliable information is worse than no information, so use some discipline and keep your devices up-to-date.

RFC 1213's definition of *sysContact* is similar to that of *sysLocation*:

```
sysContact OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write

    STATUS  mandatory
    DESCRIPTION
        "The textual identification of the contact person for this managed
         node, together with information on how to contact this person."
    ::= { system 4 }
```

sysContact is a DisplayString. It's fairly obvious what it's used for: it identifies the primary contact for the device in question. It is important to set this object with an appropriate value, as it can help your operations staff determine who needs to be contacted in the event of some catastrophic failure. You can also use it to make sure you're notified, if you're responsible for a given device, when someone needs to take your device down for maintenance or repairs. As with *sysLocation*, make sure to keep this information up-to-date as your staff changes. It's not uncommon to find devices for which the *sysContact* is someone who left the company several years ago.

*sysName*s hould be set to the fully qualified domain name (FQDN) for the managed device. In other words, it's the hostname associated with the managed device's IP address. The RFC 1213 definition follows:

```
sysName OBJECT-TYPE
   SYNTAX  DisplayString (SIZE (0..255))
   ACCESS  read-write
   STATUS  mandatory
   DESCRIPTION
      "An administratively-assigned name for this managed node. By
       convention, this is the node's fully-qualified domain name."
   ::= { system 5 }
```

The read-only and read-write parameters are the community strings for read-only and read-write access. Notice that *sysLocation*, *sysContact*, and *sysName* all have ACCESS values of read-write. With the appropriate read-write community string, anyone can change the definition of these objects and many more objects of significantly greater importance. Ultimately, it's not a huge problem if somebody maliciously makes your router lie about its locationyou probably already know that it isn't located in Antarctica. But someone who can do this can also fiddle with your routing tables and do other kinds of much more serious damage. Someone who has only the read-only community string can certainly find out more information about your network than you would like to reveal to an outsider. Setting the community strings is extremely important to maintaining a secure environment. Most devices are shipped with default community strings that are well known. Don't assume that you can put off setting your community strings until later.

The trap destination parameters specify the addresses to which traps are sent. There's nothing really magical heresince traps are asynchronous notifications generated by your devices, the agent needs to know who should receive notification. Many devices support authentication-failure traps, which are generated if someone attempts to access them using incorrect community strings. This feature is extremely useful, as it allows you to detect attempts to break into your devices. Many devices also support the ability to include a community string with traps; you can configure the NMS to respond only to traps that contain the proper community string.

Many devices have additional twists on the access and trap parameters. For example, Cisco devices allow you to create different community strings for different parts of the MIByou can use this to allow people to set some variables, but not others. Many vendors allow you to place restrictions on the hosts that are allowed to make SNMP requests. That is, the device will respond only to requests from certain IP addresses, regardless of the community string.

The range of configuration options you're likely to run into is limited only by the imagination of the vendors, so it's obviously impossible for us to describe everything you might encounter. "Agent Configuration Walkthroughs," later in this chapter, will give you an idea of how some agents implement the standard configuration parameters, and a little insight into what other features might be available.

PREV NEXT

6.2. Security Concerns

Chapter 2 discussed the security issues with SNMPv1 and SNMPv2. The biggest problem, of course, is that the read-only and read-write community strings are sent as clear-text strings; the agent or the NMS performs no encryption. Therefore, the community strings are available to anyone with access to a packet sniffer. That certainly means almost anyone on your network with a PC and the ability to download widely available software. Does that make you uncomfortable? It should.

Obviously, you need to take the same precautions with the community strings that you would with your superuser or administrator passwords. Choose community strings that are hard to guess. Mixed-case alphanumeric strings are good choices for community strings; don't use dictionary words. Although someone with the read-only community string can't do as much damage as someone with the read-write string, you might as well take the same precautions for both. Don't forget to change your community stringsmost devices ship with preconfigured community strings that are extremely easy to guess.

That doesn't solve the problems with packet sniffers . When you're configuring an agent, it's a good idea to limit the devices that can make SNMP requests (assuming that your agent allows you to make this restriction). That way, even if someone gets the community strings, he'll have to spoof the IP address of one of your management stations to do any damage.

Of course, many people know how to spoof IP addresses these days, and it's not a really good idea to assume that you can trust your employees. A better solution to the problem is to prevent the SNMP packets from being visible on your external network connections and parts of your network where you don't want them to appear. This requires configuring your routers and firewalls with access lists that block SNMP packets from the outside world (which may include parts of your own network). If you don't trust the users of your network, you may want to set up a separate administrative network to be used for SNMP queries and other management operations. This is expensive and inflexibleit's hard to imagine extending such a network beyond your core routers and serversbut it may be what your situation requires.

If you want to use SNMP to monitor your network from home, be extremely careful. You do not want your community strings traveling over the public Internet in an unencrypted form. If you plan to use SNMP tools directly from home, make sure to install VPN software, or some form of tunneling, to keep your SNMP traffic private. A better approach to home monitoring is to use a web interface; by using SSL, you can prevent others from seeing your usage graphs. (No network management products that we're aware of support SSL out of the box, but they do allow you to integrate with external servers, such as Apache, which do support SSL.)

SNMPv3 fixes most of the security problems; in particular, it makes sure that the community strings are always encrypted.

PREV NEXT

6.3. Agent Configuration Walkthroughs

In the following sections, we will walk through the configurations of some typical SNMP agents. We have chosen devices that are found on almost every modern network (PCs, Unix servers, routers, UPSs, etc.). The point of this discussion isn't to show you how your particular agent is configuredthat would not be practical, given the hundreds of devices and vendors out there. Our intent is to give you a feel for what the common options are, and what steps you'll typically go through to configure an agent.

6.3.1. Windows Agents (Net-SNMP)

Microsoft makes an SNMP agent for its operating systems. Unfortunately, it's not exactly the most feature-rich agent in the world. In this section, we'll install the Net-SNMP agent on Windows. The nice thing about this agent is that it's free. For those of you who need the support of a commercial product, take a look at Concord's SNMP agent; it's covered later in this chapter.

The Net-SNMP agent is available from http://net-snmp.sourceforge.net
for Windows in this section, and the nice thing about this is that both environments share the same configuration settings. Review the section about the Unix agent later in this chapter for details on advanced configurations.

We should point out some differences between the Windows version and the Unix version of the agent. If we wait until the end of this section, they might not catch your attention as easily:

- The version that you download from the Internet is a precompiled binary. It doesn't have support for SNMPv3 built in. You can build this support in, but you will need the Microsoft Visual C++ compiler. Download the source distribution from the Net-SNMP URL listed earlier, unpack it, and read the section in the *README.win32* file called "Microsoft Visual C++ - Workspace - Building" if you want to compile a version with SNMPv3 support.

- The configuration file takes all the same options as the Unix version of the agent. The one difference is that on Windows, paths use forward slashes. For example, a normal Windows path looks like this: *C:\somepath\where\data\lives*. In the Net-SNMP configuration files for Windows, this same path would be *C:/somepath/where/data/lives*. Just be aware of this difference.

Go to the Net-SNMP site and download the prebuilt binary for Windows. Once it's downloaded, double-click the icon to display a setup screen as in Figure 6-1.

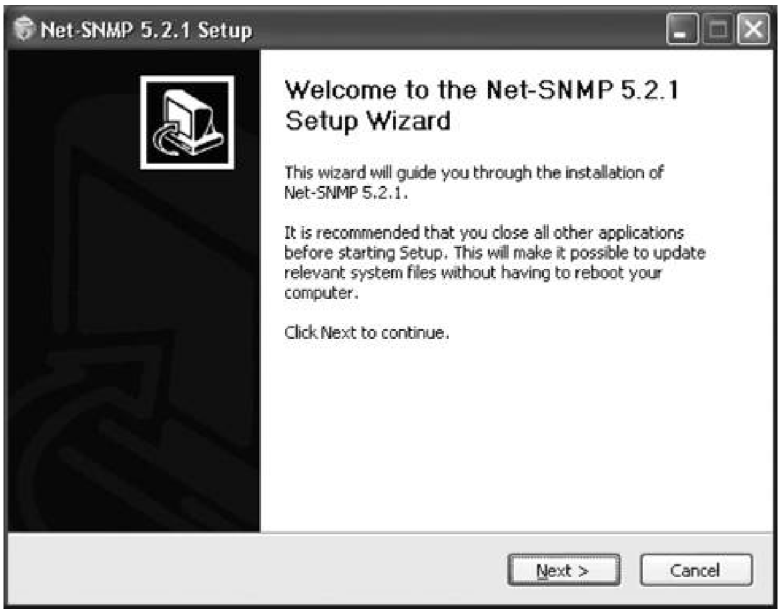
Click Next, which brings up a window like the one in Figure 6-2.

Select "I accept the terms in the License Agreement" (after you have actually read it) and click Next to display Figure 6-3.

The Base Components option will install command-line tools and other things. This is actually a good idea, because then you can use the same Net-SNMP command-line tools we use throughout the book; the options are the same. The next two options install the SNMP agent and a trap receiver. The fourth option installs the Net-SNMP Perl modules. If you are interested in using Perl, you will need to install a Perl interpreter. You can get a very good Windows interpreter from http://www.activestate.com

Figure 6-4 appears when you click Next. You are selecting the location where you want to install the package. The default, *C:\usr*, is a bit odd for Windows, but we suggest leaving it as is mainly because all the Unix examples in this book use that base directory.

Figure 6-1. Net-SNMP initial install screen



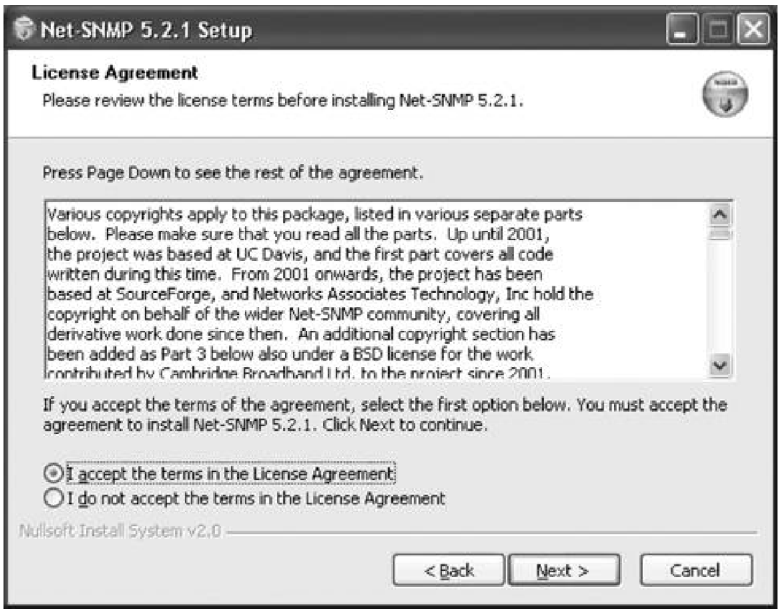
Welcome to the Net-SNMP 5.2.1 Setup Wizard

This wizard will guide you through the installation of Net-SNMP 5.2.1.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 6-2. Net-SNMP license agreement



License Agreement

Please review the license terms before installing Net-SNMP 5.2.1.

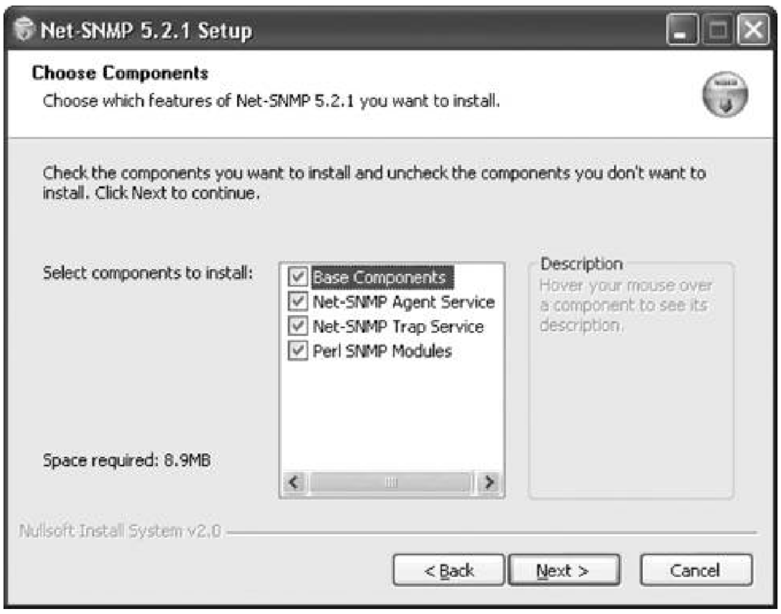
Press Page Down to see the rest of the agreement.

Various copyrights apply to this package, listed in various separate parts below. Please make sure that you read all the parts. Up until 2001, the project was based at UC Davis, and the first part covers all code written during this time. From 2001 onwards, the project has been based at SourceForge, and Networks Associates Technology, Inc hold the copyright on behalf of the wider Net-SNMP community, covering all derivative work done since then. An additional copyright section has been added as Part 3 below also under a BSD license for the work contributed by Cambridge Broadband Ltd. to the project since 2001.

If you accept the terms of the agreement, select the first option below. You must accept the terms of the agreement to install Net-SNMP 5.2.1. Click Next to continue.

[✓ I accept the terms in the License Agreement]
I do not accept the terms in the License Agreement

Figure 6-3. Net-SNMP component selection



Net-SNMP 5.2.1 Setup

Choose Components

Choose which features of Net-SNMP 5.2.1 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Base Components
- [✓] Net-SNMP Agent Service
- [✓] Net-SNMP Trap Service
- [✓] Perl SNMP Modules

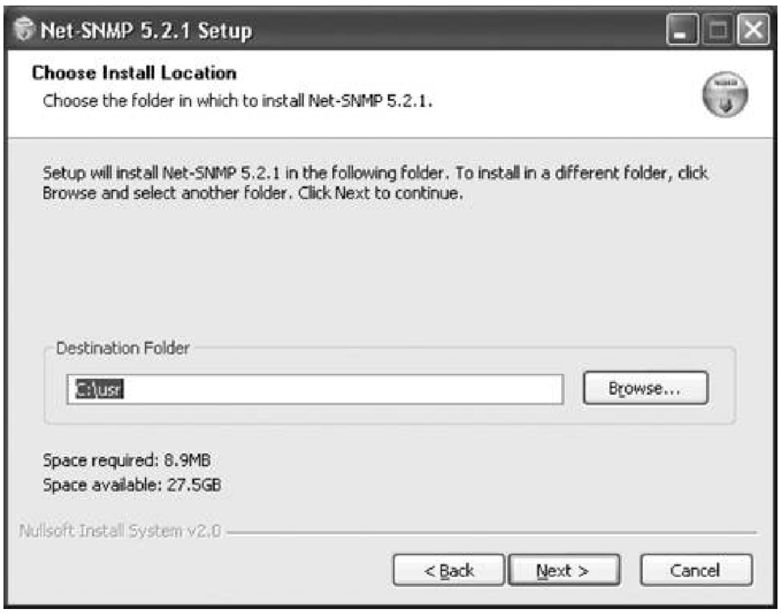
Description

Hover your mouse over a component to see its description.

Space required: 8.9MB

Nullsoft Install System v2.0

Figure 6-4. Net-SNMP install location


Net-SNMP 5.2.1 Setup

Choose Install Location

Choose the folder in which to install Net-SNMP 5.2.1.

Setup will install Net-SNMP 5.2.1 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\usr

Space required: 8.9MB
Space available: 27.5GB

Once the package is installed, you need to register the agent as a service so that it will start upon reboot. This also gives you greater control over the agent. Here is the command sequence, assuming you installed the package in *C:\usr*.

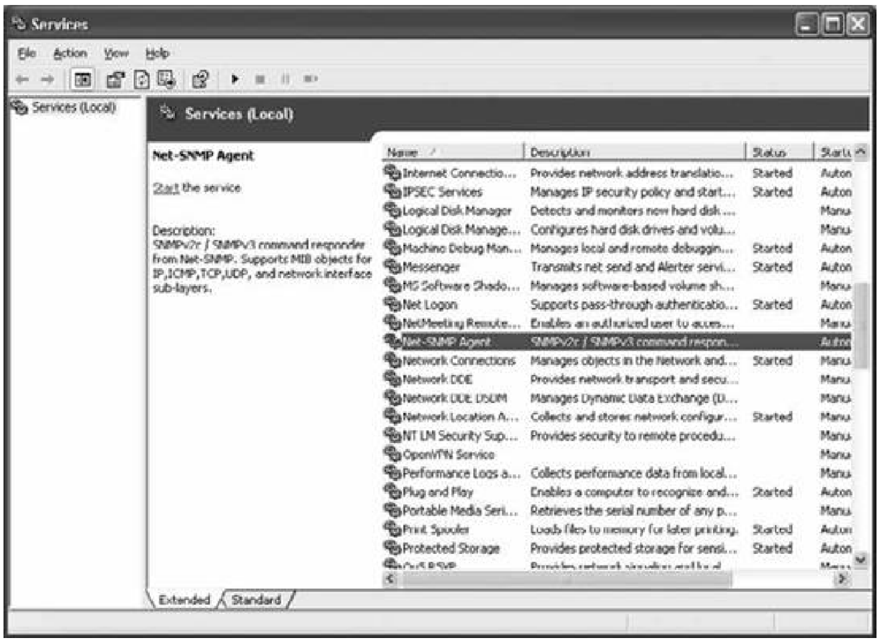
```
C:\Documents and Settings\kschmidt\Desktop>cd c:\usr

C:\usr>registeragent.bat
Registering snmpd as a service using the following additional options:
.
 -Lf "C:/usr/log/snmpd.log"
.
.
For information on running snmpd.exe and snmptrapd.exe as a Windows
service, see 'How to Register the Net-SNMP Agent and Trap Daemon as
Windows services' in README.win32.
.
Press any key to continue . . .

C:\usr>
```

Once this is done, you need to start the agent. Go to the Control Panel and bring up the Administrative Tools. Once this loads, double-click the Services icon. Scroll through the list of services until you find the Net-SNMP service. It should not be running, as shown in the status column in

Figure 6-5. Windows Services



Net-SNMP Agent

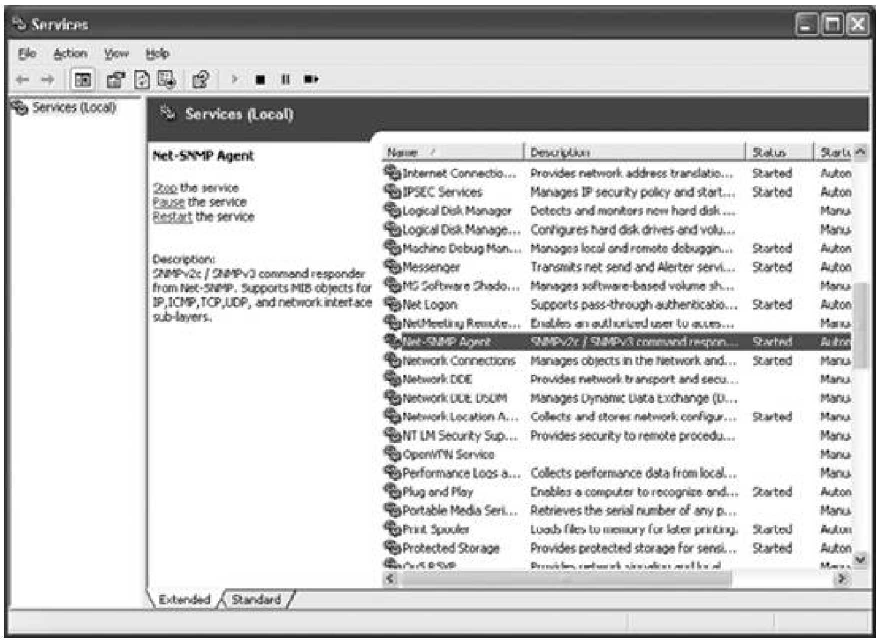
Start the service

Description:
SNMPv2c / SNMPv3 command responder from Net-SNMP. Supports MIB objects for IP,ICMP,TCP,UDP, and network interface sub-layers.

Name	Description	Status	Startu
Internet Connectio...	Provides network address translatio...	Started	Auton
IPSEC Services	Manages IP security policy and start...	Started	Auton
Logical Disk Manager	Detects and monitors new hard disk ...		Manu
Logical Disk Manage...	Configures hard disk drives and volu...		Manu
Machine Debug Man...	Manages local and remote debuggin...	Started	Auton
Messenger	Transmits net send and Alerter servi...	Started	Auton
MS Software Shado...	Manages software-based volume sh...		Manu
Net Logon	Supports pass-through authenticatio...	Started	Auton
NetMeeting Remote...	Enables an authorized user to acces...		Manu
Net-SNMP Agent	SNMPv2c / SNMPv3 command respon...		Autom
Network Connections	Manages objects in the Network and...	Started	Manu
Network DDE	Provides network transport and secu...		Manu
Network DDE DSDM	Manages Dynamic Data Exchange (D...		Manu
Network Location A...	Collects and stores network configur...	Started	Manu
NT LM Security Sup...	Provides security to remote procedu...		Manu
OpenVPN Service			Manu
Performance Logs a...	Collects performance data from local...		Manu
Plug and Play	Enables a computer to recognize and...	Started	Auton
Portable Media Seri...	Retrieves the serial number of any p...		Manu
Print Spooler	Loads files to memory for later printing.	Started	Auton
Protected Storage	Provides protected storage for sensi...	Started	Auton

Click on "Start the service" and it should start running, as shown in Figure 6-6

Figure 6-6. The running Net-SNMP agent



Net-SNMP Agent

Stop the service
Pause the service
Restart the service

Description:
SNMPv2c / SNMPv3 command responder from Net-SNMP. Supports MIB objects for IP,ICMP,TCP,UDP, and network interface sub-layers.

Name	Description	Status	Startu
Internet Connectio...	Provides network address translatio...	Started	Auton
IPSEC Services	Manages IP security policy and start...	Started	Auton
Logical Disk Manager	Detects and monitors new hard disk		Manu
Logical Disk Manage...	Configures hard disk drives and volu...		Manu
Machine Debug Man...	Manages local and remote debuggin...	Started	Auton
Messenger	Transmits net send and Alerter servi...	Started	Auton
MS Software Shado...	Manages software-based volume sh...		Manu
Net Logon	Supports pass-through authenticatio...	Started	Auton
NetMeeting Remote...	Enables an authorized user to acces...		Manu
Net-SNMP Agent	SNMPv2c / SNMPv3 command respon...	Started	Auton
Network Connections	Manages objects in the Network and...	Started	Manu
Network DDE	Provides network transport and secu...		Manu
Network DDE DSDM	Manages Dynamic Data Exchange (D...		Manu
Network Location A...	Collects and stores network configur...	Started	Manu
NT LM Security Sup...	Provides security to remote procedu...		Manu
OpenVPN Service			Manu
Performance Logs a...	Collects performance data from local...		Manu
Plug and Play	Enables a computer to recognize and...	Started	Auton
Portable Media Seri...	Retrieves the serial number of any p...		Manu
Print Spooler	Loads files to memory for later printing.	Started	Auton
Protected Storage	Provides protected storage for sensi...	Started	Auton
QoS RSVP	Provides network signaling and local		Manu

You can now stop, pause, and restart the agent. Start it and then try to access the agent. You can use the SNMP tools installed on the machine to do this, using the following commands:

```
C:\usr>snmpwalk -v2c -c public localhost
Timeout: No Response from localhost

C:\usr>
```

We get a timeout because we haven't configured the agent to respond to any SNMP queries. To do this, use your favorite editor to create a file called *c:\usr\etc\snmp\snmpd.conf*

```
rwcommunity public
```

This sets up a read-write community string of public. Please note that this example is just for demonstration purposes. You should choose a more robust community string. For the changes to take effect, you need to go to the Services panel and click Restart.

When you make changes to Net-SNMP files, to have them take effect you must go to the Services panel and restart the agent service.

Once you have done that, try the same command again, and the agent responds to our SNMP query:

```
C:\usr>snmpwalk -v2c -c public localhost
SNMPv2-MIB::sysDescr.0 = STRING: Windows loanera22p 5.1.2600 Service Pack 1 XP P
rofessional x86 Family 6 Model 8 Stepping 10
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.13
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (4540) 0:00:45.40
SNMPv2-MIB::sysContact.0 = STRING: USER
SNMPv2-MIB::sysName.0 = STRING: loanera22p
SNMPv2-MIB::sysLocation.0 = STRING: unknown
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (5) 0:00:00.05
SNMPv2-MIB::sysORID.1 = OID: IF-MIB::ifMIB
SNMPv2-MIB::sysORID.2 = OID: TCP-MIB::tcpMIB
SNMPv2-MIB::sysORID.3 = OID: IP-MIB::ip
SNMPv2-MIB::sysORID.4 = OID: UDP-MIB::udpMIB
SNMPv2-MIB::sysORID.5 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.6 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
SNMPv2-MIB::sysORID.7 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.8 = OID: SNMP-MPD-MIB::snmpMPDCompliance
SNMPv2-MIB::sysORID.9 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
```

Now, if you don't want to run the agent as a service, you can run it from the command line. The command is simply *snmpd*:

```
C:\usr>snmpd
No log handling enabled - turning on stderr logging
NET-SNMP version 5.2.1
```

The Net-SNMP folks have done a really nice job of putting the Windows distribution together. It may seem like quite a few steps are involved, but believe me you would not even be done installing the Microsoft agent at this point. If you want to know more about configuring the agent, read the section about the Unix version later in this chapter. The *snmpconf* Perl script that comes with the Unix version also comes with the Windows version, so you can use it to create configuration files.

6.3.2. HP OpenView Agent for HP-UX and Solaris

One text configuration file controls the parameters for the OpenView agent; the file is typically named */etc/SnmpAgent.d/snmpd.conf*, or */etc/snmpd.conf* on older systems. You don't need to edit this file for the agent to function normally. If you do edit it, you must stop and restart the master agent by executing the *SnmpMaster* script, first with a stop and then a start :

```
$ /sbin/init.d/SnmpMaster stop
$ /sbin/init.d/SnmpMaster start
```

6.3.2.1. Simple configuration

The following configuration file configures the agent to respond to get requests using the community name *public* and set requests using the community name *private*. There are no restrictions on which MIBs can be queried, or which hosts can make the queries. This configuration has no security, since the community strings are set to commonly used defaults and are widely known. The OpenView authentication-failure traps by default, so you don't have to enable these traps in the configuration file.

```
get-community-name:    public
set-community-name:    private
trap-dest:             127.0.0.1
contact:               B.Gates
location:              12 Pyramid - Egypt
```

The simplest configuration is to edit the file and place more reasonable community names in the first two lines. We can't say it too much: community names are essentially passwords. Use the same rules for picking community names that you would for choosing the root password. You should always set the destination trap host (`TRap-dest`) to the IP address of the host that will receive the trap.

The next example configures several different community names:

```
get-community-name:    public
```

```
get-community-name:    media
set-community-name:    hushed
set-community-name:    veryprivate
set-community-name:    shhhh
```

We have created two get (read-only) communities and three set (read-write) communities. These communities can be used as you see fit. (In real life, we would have chosen more obscure names.) For example, you might give your operations group in New York *public* community access and your operations group in Atlanta *media* community access. The remaining set communities can further be subdivided among various administrators and other staff who need read-write access.

6.3.2.2. Advanced configuration

Setting up multiple community strings doesn't sound very useful, and by itself, it isn't. But you can take the concept a step further and create different communities, each consisting of a few particular hosts and able to access only some of the objects SNMP manages. The next example allows the host 10.123.56.25 to issue gets using the community name *comname* and sets using the community name
10.123.46.101 can issue gets using only the community name *comname*
the IP: directive; you must use IP addresses.

```
get-community-name     comname IP: 10.123.56.25 10.123.46.101
set-community-name     private IP: 10.123.56.25
```

You can also configure the agent to restrict access to MIB subtrees based on IP addresses. The next example allows any host to get any object under *iso.org.dod.internet.mgmt.mib-2*
interfaces subtree. The minus sign (-) in front of*interfaces* instructs the agent to disallow access to this subtree.

```
get-community-name     public VIEW: mib-2 -interfaces
```

The final example sets up multiple community names for both sets and gets. An administrator who is located at host 10.123.46.25 and knows the *admin* community string has read access to the entire MIB tree; with the *adminset* community string, he has write access to the entire tree. Someone with the

community string can sit anywhere and access everything in *mib-2* except for the
must be sitting at his desk (10.123.56.101) to issue sets and is not allowed to set anything in the
subtree.

```
get-community-name     operator VIEW: mib-2 -interfaces
get-community-name     admin    IP: 10.123.56.25
set-community-name     operset  IP: 10.123.46.101 VIEW: -mib-2
set-community-name     adminset IP: 10.123.56.25
```

6.3.3. Net-SNMP for Unix

Net-SNMP is an open source agent that is freely available from http://net-snmp.sourceforge.net
focus on Net-SNMP Version 5.2.1, which is the most recent as of this publication. Once you have downloaded and unpacked the distribution, cd into the directory in which you unpacked Net-SNMP and read the *README* and *INSTALL* files. These files provide general information on installing the agent and don't require much explanation here.

Net-SNMP uses a *configure* script to make sure your environment has some key utilities and libraries installed, so it can be compiled successfully. Many configuration options are settable when you run this script. To see a list of them, run the following command:

```
net-snmp-5.2.1/> ./configure --help
```

One common option is - -prefix=PATH. This specifies an alternate installation
Net-SNMP will install in */usr/local/bin*, */usr/local/man*, etc.

We'll be running *configure* without any options, which means our Net-SNMP build will have default values assigned for various options. For example, the agent binary will be placed in
following command to begin the configuration process:

```
net-snmp-5.2.1/> ./configure
```

Note that by default, Net-SNMP does not compile in the Host Resources MIB. You will want to add this to the configure line if you want to access things like CPU statistics, memory and disk information, etc. Run configure like so:

```
net-snmp-5.2.1/> ./configure -with-mib-modules=host
```

You will see various messages about what features *configure* is looking for and whether they're found.

After running for a while, *configure* will ask for some basic SNMP information:

```
    ************** Configuration Section **************

    You are about to be prompted by a series of questions. Answer
them carefully, as they determine how the snmp agent and related
applications are to function.

    After the configure script finishes, you can browse the newly
created config.h file for further - less important - parameters to
modify. Be careful if you re-run configure though since config.h will
be over written.

-Press return to continue-
```

When you press Return, you'll be prompted for the particular version of SNMP you wish to use:

```
disabling above prompt for future runs...  yes
checking System Contact Information...

*** Default SNMP Version:

        Starting with Net-SNMP 5.0, you can choose the default version of
the SNMP protocol to use when no version is given explicitly on the
command line, or via an 'snmp.conf' file.  In the past this was set to
SNMPv1, but you can use this to switch to SNMPv3 if desired.  SNMPv3
will provide a more secure management environment (and thus you're
encouraged to switch to SNMPv3), but may break existing scripts that
rely on the old behaviour.  (Though such scripts will probably need to
be changed to use the '-c' community flag anyway, as the SNMPv1
command line usage has changed as well.).
   At this prompt you can select "1", "2" (for SNMPv2c), or "3" as
the default version for the command tools (snmpget, ...) to use.  This
can always be overridden at runtime using the -v flag to the tools, or
```

```
by using the "defVersion" token in your snmp.conf file.
   Providing the --with-default-snmp-version="x" parameter to ./configure
will avoid this prompt.

Default version of SNMP to use (3): 3
setting Default version of SNMP to use to...  3
```

We've decided to select version 3 as our default. Since it's now a full IETF standard, there is no reason to not use it. As noted in the text before the prompt, you can always override the version for the command-line tools with the -v switch.

Next we are asked to configure the system contact:

```
checking System Contact Information...


*** System Contact Information:

        Describes who should be contacted about the host the agent is
running on.  This information is available in the MIB-II tree.  This
can also be over-ridden using the "syscontact" syntax in the agent's
configuration files.
  Providing the --with-sys-contact="contact" parameter to ./configure
will avoid this prompt.

System Contact Information (kjs@): snmpadmin@oreilly.com
setting System Contact Information to...  snmpadmin@ora.com
```

We've decided to set our contact information to something useful, but we could have left it blank. The next item you're asked to configure is system location. We've chosen an informative value, but again we could have left it blank:

```
checking System Location...


*** System Location:

        Describes the location of the system.  This information is
```

```
available in the MIB-II tree.  this can also be over-ridden using the
"syslocation" syntax in the agent's configuration files.
  Providing the --with-sys-location="location" parameter to ./configure
will avoid this prompt.

System Location (Unknown): **FTP Server #1, O'Reilly Data Center**
setting System Location to...  FTP Server #1, O'Reilly Data Center
```

The final options you need to configure are the locations of *snmpd*'s logfile and its persistent storage. In both cases, you simply press Enter to accept the default locations.

```
checking Location to write logfile...


*** Logfile location:

        Enter the default location for the snmpd agent to dump
information & errors to.  If not defined (enter the keyword "none"
at the prompt below) the agent will use stdout and stderr instead.
(Note: This value can be over-ridden using command line options.)
  Providing the --with-logfile="path" parameter to ./configure
will avoid this prompt.

Location to write logfile (/var/log/snmpd.log): **<Enter>**
setting Location to write logfile to...  /var/log/snmpd.log
checking Location to write persistent information...


*** snmpd persistent storage location:

        Enter a directory for the SNMP library to store persistent
data in the form of a configuration file.  This default location is
different than the old default location (which was for ucd-snmp).  If
you stay with the new path, I'll ask you in a second if you wish to
copy your files over to the new location (once only).  If you pick
some other path than the default, you'll have to copy them yourself.
There is nothing wrong with picking the old path (/var/ucd-snmp) if
you'd rather.
  Providing the --with-persistent-directory="path" parameter to
./configure will avoid this prompt.
```

```
Location to write persistent information (/var/net-snmp): <Enter>
setting Location to write persistent information to...  /var/net-snmp
configure: creating ./config.status
config.status: creating Makefile
config.status: creating snmplib/Makefile
config.status: creating apps/Makefile
config.status: creating apps/snmpnetstat/Makefile
config.status: creating agent/Makefile
config.status: creating agent/helpers/Makefile
config.status: creating agent/mibgroup/Makefile
config.status: creating local/Makefile
config.status: creating testing/Makefile
config.status: creating man/Makefile
config.status: creating mibs/Makefile
config.status: creating net-snmp-config
config.status: creating include/net-snmp/net-snmp-config.h
config.status: executing default commands

---------------------------------------------------------
            Net-SNMP configuration summary:
---------------------------------------------------------

  SNMP Versions Supported:    1 2c 3
  Net-SNMP Version:           5.2.1
  Building for:               linux
  Network transport support:  Callback Unix TCP UDP
  SNMPv3 Security Modules:    usm
  Agent MIB code:             mibII ucd_snmp snmpv3mibs notification target agent_mibs agentx utilities host
  SNMP Perl modules:          disabled
  Embedded perl support:      disabled
  Authentication support:     MD5 SHA1
  Encryption support:         DES AES

---------------------------------------------------------
```

You can now compile your new package with the make command. The compilation process displays many messages, most of which you can ignore. In short, if it completes, you've succeeded and can proceed to installation. If not, you will see errors and should investigate what went wrong. Install your new package with the command make install. By default, this command installs various executables in other important information in */usr/local/share/snmp*.

At this point, you can configure the agent further by using one of two approaches:

- Run the program */usr/local/bin/snmpconf*, which asks you a lot of questions and creates a configuration file. The configuration script is surprisingly confusing, though, so it's hard to recommend this approach.
- Craft a configuration by hand. If you're not interested in SNMPv3, this is fairly easy.

6.3.3.1. Running the configuration script

The configuration script is rather long and complex. Here are a few pointers:

- It starts by asking whether you want to create *snmp.conf* or *snmpd.conf* select *snmpd.conf*. *snmp.conf* sets up some defaults for command-line tools such as Strictly speaking, creating *snmp.conf* isn't necessary.
- Most of the configurable options have to do with SNMPv3.
- When you're finished configuring, the script leaves the configuration file in your current directory. You can place the files in *~/.snmp*, if they're for your own use, or in this configuration to be used by everyone on the system.

6.3.3.2. Creating a configuration by hand

If you don't want to use SNMPv3, creating your own configuration file is easy. Here's a very simple configuration file:

```
syslocation    "O'Reilly Data Center"
syscontact     snmpadmin@oreilly.com
rwcommunity    private
rocommunity    public
authtrapenable 1
```

```
trapcommunity  trapsRus
trapsink       nmshost.oreilly.com
trap2sink      nmshost.oreilly.com
```

The configuration items should be familiar: we're setting up the system location; the system contact; the read-write, read-only, and trap community strings; and the destination to which traps should be sent. We're also enabling authentication traps. Note that we configured destinations for both SNMP Version 1 and Version 2 traps. The trap destination lines (`trapsink` and `trap2sink`) can also have a trap community string, if the NMS at the given host requires a different community name.

The `rwcommunity` and `rocommunity` lines allow us to be a bit more sophisticated than the example indicates. We're allowed to specify the network or subnet to which the community strings apply, and an object ID that restricts queries to MIB objects that are underneath that OID. For example, if you want to restrict read-write access to management stations on the subnetwork 10.0.15.0/24, you could use the line:

```
rwcommunity    private 10.0.15.0
```

If you take this route, you should certainly look at the *EXAMPLE.conf* file in the directory in which you built Net-SNMP . You can modify this file and install it in the appropriate location (either */usr/local/share/snmp/snmpd.conf*), or you can take ideas from it and use them in your own configuration. It includes some particularly clever tricks that we'll discuss in Chapter 10
configuration we're discussing here).

Finally, let's look at configuring Net-SNMP to use SNMPv3 . We'll also discuss a few utility commands that can help make managing the various security options much easier. Keep in mind that because of its security features, SNMPv3 is user based. Even though you may not want to use authentication or privacy you may still need to provide a username, even if it is blank.

> To use SHA and DES encryption, you will need to install the OpenSSL libraries on the machine where you built (or plan to build) Net-SNMP. Go to http://www.openssl.org to get these.

To create a user named *kschmidt* who has read-write access to the *system*
to your *snmpd.conf* file:

```
rwuser  kschmidt auth system
```

To create a user with read-only access, use the command `rouser` instead of
secure authentication, but not privacy: the SNMP packets themselves aren't encrypted. The other possibilities are `noauth` (no authentication and no privacy) and`priv` (authentication and privacy). Now add the following line to */usr/local/share/snmp/snmpd.conf*:

```
createUser kschmidt MD5 mysecretpass
```

This creates an MD5 password for the user *kschmidt*. The password assigned to
create a user with a DES passphrase in addition to an MD5 password, add the following line to */usr/local/share/snmp/snmpd.conf:*

```
createUser kschmidt MD5 mysecretpass DES mypassphrase
```

If you omit `mypassphrase`, Net-SNMP sets the DES passphrase to be the same as the MD5 password. The RFCs for SNMPv3 recommend that passwords and passphrases be at least eight characters long; Net-SNMP enforces this recommendation and won't accept shorter passwords.

When the agent is started, it reads the configuration file and computes secret keys for the users added.

Now we can perform an snmpwalk using Version 3 authentication. The following command specifies Version 3, with the username `kschmidt`, requesting authentication without privacy using the MD5 algorithm The password is `mysecretpass`:

```
$ snmpwalk -v 3 -u kschmidt -l authNoPriv -a MD5 -A mysecretpass \
server.ora.com
system.sysDescr.0 = Linux server 2.2.14-VA.2.1 #1 Mon Jul 31 21:58:22 PDT 2000 i686
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
system.sysUpTime.0 = Timeticks: (1360) 0:00:13.60
system.sysContact.0 = "Ora Network Admin"
system.sysName.0 = server
system.sysLocation.0 = "Atlanta, Ga"
system.sysServices.0 = 0
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
system.sysORTable.sysOREntry.sysORID.1 = OID: ifMIB
...
UDP-MIB::udpOutDatagrams.0 = No more variables left in this MIB View (It is past the end of the MIB tree)
```

Note that we see only objects from the *system* subtree, even though the command tries to walk the entire tree. This limitation occurs because we have given *kschmidt* access only to the
tries to query a subtree he is not allowed to access, he gets the following result:

```
$ snmpwalk -v 3 -u kschmidt -l authNoPriv -a MD5 -A mysecretpass \
server.ora.com interfaces
IF-MIB::interfaces = No Such Object available on this agent at this OID
```

If you want privacy in addition to authentication, use a command like this:

```
$ snmpwalk -v 3 -u kschmidt -l authPriv -a MD5 -A mysecretpass -x DES -X \
mypassphrase server.ora.com
```

6.3.3.3. Using snmpusm to manage users

The Net-SNMP utility *snmpusm* is used to maintain SNMPv3 users. This utility can be very useful when it comes to managing and creating users on the fly.

> Note that to use this command, your SNMPv3 user must have write access to the `usmUserTable` in the agent. With user*kschmidt*, we restricted his access to the subtree by adding this line to the *snmpd.conf* file: `rwuser kschmidt auth system`
> remedied by simply removing the system attribute: `rwuser kschmidt auth`

The following command creates the user *kjs* by cloning the *kschmidt* user:

```
$ snmpusm -v 3 -u kschmidt -l authNoPriv -a MD5 -A mysecretpass localhost create \ kjs kschmidt
```

Since *kjs* was cloned from*kschmidt*, the two users now have the same authorization, password, and passphrase. It's obviously essential to change *kjs*'s password. To do so, use snmpusm with the -Ca option. Similarly, to change the privacy passphrase, use -Cx. The following two commands change the password and passphrase for the new user *kjs*:

```
$ snmpusm -v 3 -l authNoPriv -u kschmidt -a MD5 -A mysecretpass localhost -Ca
passwd mysecretpass mynewpass kjs
$ snmpusm -v 3 -l authPriv -u kschmidt -a MD5 -A mysecretpass localhost -Cx
passwd mypassphrase mynewphrase kjs
```

There are many things to note about this seemingly simple operation:

- You must know both the password and passphrase for *kschmidt* passphrase for *kjs*. Presumably this is the case since you are the admin who is allowed to write to the `usmUserTable`.
- According to the documentation, Net-SNMP allows you to clone on to the same user only once. Attempts to reclone a previously cloned user appear to succeed but are silently ignored. The SNMPv3 USM specification (RFC 3414) mandates this particular behavior.
- *snmpusm* can only clone users; it can't create them from scratch. Therefore, you must create the initial user by hand, using the process described earlier. (This isn't quite true. user, but once you've done so you have to assign it a password by changing its previous password. So, you're in a Catch-22: the new user doesn't have a password, but you can't change his password. The only way to do this is by cloning the last user you created, and changing the password as we described here.)

For the user to be written to the persistent *snmpd.conf* file, you must either stop and restart the agent or send a HUP signal to the *snmpd* process. This forces the agent to write the current state of the user table to disk, so the agent can reread it upon startup. Note that using kill -9 does not produce the desired result.

The snmpusm command exists primarily to allow end users to manage their own passwords and passphrases. As the administrator, you may want to change your users' passwords and passphrases periodically. This is possible only if you keep a master list of users and their passwords and passphrases.

If the engine ID changes, you will have to regenerate all the usernames, passwords, and passphrases. (Remember that the engine ID depends in part on the host's IP address and therefore changes if you have to change the address.) To do this, stop the agent and edit the */var/net-snmp/snmpd.conf* the persistent `usmUser` enTRies and add new `createUser` commands (as described previously) for your users. A `usmUser` enTRy looks something like this:

```
usmUser 1 3 0x80001f8880389a8f7c2f5ba342 0x6b6a7300 0x6b6a7300 NULL .1.3.6.1.6.3.10.1.1.2
0x0ecdaf0c88993c416bd8f0a555a11a3a .1.3.6.1.6.3.10.1.2.2 0xf64fee1207d9a959e53c47398e05e872 ""
```

6.3.4. Concord SystemEDGE Agent for Unix and Windows

Concord SystemEDGE is a commercial product that can be used as a subagent to the standard Windows agent. On Unix systems, this agent can be used either as a standalone agent or side by side with an existing agent. It runs on Linux, Solaris, and other operating systems. The CD on which the product is shipped includes agents for all the platforms SystemEDGE supports. Whenever possible, SystemEDGE uses the platform's native package manager to make installation easier. Each architecture-dependent version of the agent comes with an easy-to-follow *README* file for installation. See discussion of this agent's capabilities.

6.3.4.1. Simple configuration

The SystemEDGE configuration file is located in */etc/sysedge.cf*. Use your editor of choice to make changes to this file. You must stop and restart SystemEDGE for your changes to take effect. The configuration file format is the same for all the versions of SystemEDGE

For a typical SNMP configuration, *sysedge.cf* looks like this:

```
community public read-only
community veryprivate read-write 127.0.0.1 10.123.56.25
community traps 127.0.0.1
```

Comment lines begin with a `#` character. The first parameter sets the read-only community to read-write community is defined to be `veryprivate`. The two IP addresses following the read-write community string are an access list that tells the agent to allow set operations from only. Always use an access list if possible; without this security feature, any host can execute set operations. Note that there is a space between the two addresses, not a Tab character. The third option tells the agent where to send traps; in this case, to *localhost* (`127.0.0.1`).

The agent sends authentication-failure traps by default, and we strongly recommend using them. If you don't want authentication-failure traps, include the following line in your configuration file:

```
no_authen_traps
```

6.3.4.2. Advanced configuration

SystemEDGE provides some powerful self-monitoring capabilities . These extensions (found only in Concord's Empire private enterprise MIB) are similar to the RMON MIB, which is discussed in
Empire's extensions can reduce network load by allowing the agent, rather than an NMS, to perform monitoring (polling) of important system objects. For example, the agent can be instructed to make sure the free space available in the root filesystem stays above some predefined threshold. When this threshold is crossed, the agent sends a trap to the NMS so that the condition can be dealt with appropriately.

The following line shows how you can monitor and restart sendmail if it dies:

```
watch process procAlive 'sendmail' 1 0x100 60 'Watch Sendmail' '/etc/init.d/sendmail start'
```

This monitor sends a trap to the NMS, defined earlier as `community traps 127.0.0.1`
dies. The agent then executes */etc/init.d/sendmail start* to restart the process. The general form of this command is:

```
watch process procAlive 'procname' index flags interv 'description' 'action'
```

The *procname* parameter is a regular expression that SystemEDGE uses to select the processes that it is monitoring; in this case, we're watching processes with the name *sendmail*
process-monitoring table must have a unique *index*; in this example, we used the value 1. We could have picked any integer, as long as that integer was not already in use in the table. The
hexadecimal[*] flag that changes the behavior of the monitor. We specified a flag of
monitor that the process it's watching spawns child processes; this flag ensures that SystemEDGE will take action only when the parent sendmail process dies, not when any of the children die. The use of process-monitor flags is beyond the scope of this chapter; see the manual that comes with SystemEDGE for more information. The *interv* parameter specifies how often (in seconds) the agent checks the process's status. We have set the interval to 60 seconds. The *description* parameter contains information about the process being monitored; it can be up to 128 characters in length. It is a good idea to use a description that indicates what is being monitored, since the agent stores this value in the monitor table for retrieval by an NMS and includes it in the variable bindings when a trap is sent. The final parameter is the action the monitor will take when the process dies; we chose to restart the daemon.

[*] Generally speaking, there are several ways to represent hexadecimal numbers. SystemEDGE uses the notion of a number prefixed with *0x*, which should be familiar to C and Perl programmers.

SystemEDGE can be extended by using plug-ins . These plug-ins manage and monitor applications such as Apache (web server), Exchange (Microsoft mail), and Oracle (database), to name a few. A "top processes" plug-in named *topprocs* comes with every distribution. The following statement tells SystemEDGE to load this plug-in for 64-bit Solaris (this statement is similar for other Unix platforms, and for Windows):

```
sysedge_plugin /opt/EMPsysedge/plugins/topprocs/topprocs-sol64bit.so
```

The folks at Concord have taken great care to add useful comments to the
are often all you need to configure the agent.

6.3.5. Cisco Devices

Cisco Systems produces a wide range of routers, switches, and other networking equipment. The configuration process is virtually the same on all Cisco devices , because they share the IOS operating system.[†] There are some minor differences in the parameters that can be configured on every device; these generally have to do with the capabilities of the device, rather than the SNMP implementation.

[†] There are some exceptions to this rule, such as the PIX firewalls
usually mean that the product is made by a company that Cisco acquired.

To configure the SNMP parameters, you must be in *enable* mode. You can use the following commands to see what traps are available:

```
router> enable
Password: mypassword
router# config terminal
router(config)# snmp-server enable traps ?
  bgp         Enable BGP state change traps
  envmon      Enable SNMP environmental monitor traps
  frame-relay  Enable SNMP frame-relay traps
```

```
 isdn        Enable SNMP isdn traps
 <cr>
```

The question mark tells the router to respond with the possible completions for the command you're typing. You can use this feature throughout the entire command-line interface. If the part of the command you have already typed has a syntax error, the router will give you the "Unrecognized command" message when you type the question mark. `<cr>` tells you that you can exit without configuring the command (`enable traps` in this case) by pressing Return.

6.3.5.1. Simple configuration

Here's a simple configuration that lets you start using the SNMP agent:

```
router(config)# snmp-server community private RW
router(config)# snmp-server community public RO
router(config)# snmp-server trap-authentication
router(config)# snmp-server location Delta Building - 1st Floor
router(config)# snmp-server contact J Jones
router(config)# snmp-server host 10.123.135.25 public
```

Most of these commands set parameters with which you should be familiar by now. We define two communities, `public` and `private`, with read-only (RO) and read-write (RW) permissions, respectively. `trap-authentication` turns on authentication-failure traps. The command `snmp-server host 10.123.135.25 public` destination to which traps should be sent. The IP address is set to the address of our NMS. The community string `public` will be included in the traps.

6.3.5.2. Advanced configuration

The following configuration item tells the device what interface it should use when sending out SNMP traps:

```
router(config)# snmp-server trap-source VLAN1
```

Configuring the trap source is useful because routers, by definition, have multiple interfaces. This command allows you to send all your traps out through a particular interface.

There may be times when you want to send only certain traps to your NMS. The next item sends only environmental monitor traps to the specified host, 172.16.52.25 (the *envmon*
Cisco devices):

```
router(config)# snmp-server host 172.16.52.25 public envmon
```

One of the most frightening SNMP sets is the Cisco shutdown, which lets you shut down the router from the NMS. The good news is that you have to include a switch in the configuration before the router will respond to shutdown commands. Issuing the following command disables shutdowns:

```
router(config)# no snmp-server system-shutdown
```

To receive traps about authentication failures (something trying to poll your device with the wrong community name), add the following line:

```
router(config)# snmp-server trap-authentication
```

The final advanced configuration parameter is an access list. The first line sets up access list 15. It states that the IP address 10.123.56.25 is permitted to access the agent. The second line says that anyone that passes access list 15 (i.e., a host with IP address 10.123.56.25) and gives the community name *notsopublic* has read-only (RO) access to the agent. Access lists are a very powerful tool for controlling access to your network. They're beyond the scope of this book, but if you're not familiar with them, you should be.

```
router(config)# access-list 15 permit 10.123.56.25
router(config)# snmp-server community notsopublic RO 15
```

6.3.5.3. Configuring SNMPv3

The first task in configuring SNMPv3 is to define a view. To simplify things, we'll create a view that allows access to the entire *internet* subtree:

```
router(config)# snmp-server view readview internet included
```

This command creates a view called *readview*. If you want to limit the view to the
replace internet with system. The included keyword states that the specified tree should be included in the view; use excluded if you want to exclude a certain subtree.

Next, create a group that uses the new view. The following command creates a group called
means that SNMPv3 should be used. The auth keyword specifies that the entity should authenticate packets without encrypting them; read readview says that the view named *readview*
members of the *readonly* group access the router.

```
router(config)# snmp-server group readonly v3 auth read readview
```

Now let's create a user. The following command creates a user called
readonly group. auth md5 specifies that the router should use MD5 to authenticate the user (the other possibility is sha). The final item on the command line is the user's password or passphrase, which cannot exceed 64 characters.

```
router(config)# snmp-server user kschmidt readonly v3 auth md5 mysecretpass
```

This configuration uses encryption only to prevent passwords from being transferred in the clear. The SNMP packets themselves, which may contain information that you don't want available to the public, are sent without encryption and can therefore be read by anyone who has a packet sniffer and access to your network. If you want to go a step further and encrypt the packets themselves, use a command like this:

```
router(config)# snmp-server user kschmidt readonly v3 auth md5 mysecretpass \
priv des56 passphrase
```

The additional keywords on this command specify privacy (i.e., encryption for all SNMP packets), use of DES 56-bit encryption, and a passphrase to use when encrypting packets.

The encrypted passwords and passphrases depend on the engine ID, so if the engine ID changes, you'll need to delete any users you have defined (with the familiar IOS no command), and re-create them (with snmp-server user commands). Why would the engine ID change? It's possible to set the engine ID on the IOS command line. You should never need to set the engine ID explicitly, but if you do, you'll have to delete and re-create your users.

This has been the briefest of introductions to configuring SNMPv3 on a Cisco router. For more details, see Cisco's documentation, which is available at http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t3/snmp3.htm

That's it! You now have a working SNMP configuration for your Cisco router.

6.3.6. APC Symetra

APC's uninterruptible power supplies (UPSs) are typical of a large class of products that aren't usually considered network devices, but that have incorporated a network interface for the purpose of management.

To configure an APC UPS, you can use its management port (a familiar serial port to which you can connect a console terminal) or, assuming that you've performed basic network configuration, telnet to the UPS's IP address. SNMP configuration is the same regardless of the method you use. Either way, you get a Text User Interface (TUI) that presents you with rather old-fashioned menusyou type your menu selection (usually a number) followed by Enter to navigate through the menus.

We'll assume that you've already performed basic network configuration, such as assigning an IP address for the UPS. To configure SNMP, go to the Network menu and select 5 to go into the SNMP submenu. You should get a menu like this:

```
------- SNMP -------------------------------------------------------------

    1- Access Control 1
    2- Access Control 2
    3- Access Control 3
    4- Access Control 4
    5- Trap Receiver 1
    6- Trap Receiver 2
    7- Trap Receiver 3
    8- Trap Receiver 4
    9- System
    10- Summary
```

```
     ?- Help
<ENTER> Redisplay Menu
  <ESC> Return To Previous Menu

>
```

You need to configure three distinct sections: Access Control, Trap Receiver, and System. To see a summary of the current SNMP settings, use the Summary submenu.

This particular device allows us to specify four IP addresses for access control and four IP addresses to receive traps. The access control items allow you to configure the IP addresses of your management stationsthis is similar to the access lists we've seen in other devices, and is obviously basic to security. The UPS will reply only to queries from the IP addresses you have listed. Configuration is a bit awkwardyou need to go to a separate menu to configure each IP address. Here's what you'll see when configuring the Access Control 1 submenu:

```
------- Access Control 1 ------------------------------------------------

        Access Control Summary
        #  Community  Access      NMS IP
        -----------------------------------------------------------------
        1  public     Read        10.123.56.25
        2  private    Write       10.123.56.25
        3  public2    Disabled    0.0.0.0
        4  private2   Disabled    0.0.0.0

     1- Community      : public
     2- Access Type    : Read
     3- NMS IP Address : 10.123.56.25
     4- Accept Changes :

     ?- Help
<ENTER> Redisplay Menu
  <ESC> Return To Previous Menu

>
```

The first part of the menu summarizes the state of access control. On this menu, we can change only the first item on the list. The special address 0.0.0.0 is a wildcardit means that the UPS will respond to queries from any IP address. Although addresses 3 and 4 are set to 0.0.0.0, these addresses are currently disabled, and that's how we want to keep them. We want the UPS to respond only to the management stations we explicitly list.

On this menu, we've configured items 1 (the community string), 2 (the access type), and 3 (the IP address). We've set the community string to `public` (not a choice you'd want in a real configuration), the access type to `Read` (allowing various SNMP get operations, but no set operations), and the NMS IP address to `10.123.56.25`. The net effect is that the UPS's SNMP agent will accept get requests from IP address 10.123.56.25 with the community name *public*. When you are satisfied with the configuration, enter a 4 to accept your changes.

To configure the second access control item, press Esc to return to the previous menu; then select 2. As you can see, we allow 10.123.56.25 to perform set operations. We don't have any other management stations, so we've left items 3 and 4 disabled.

Once the Access Control section is complete, you can start configuring traps. The Trap Receivers section is simply a list of NMSs that receive traps. As with Access Control, four trap receivers can be configured. To get to the first trap receiver, return to the SNMP menu and select menu 5. A typical trap receiver setup looks like this:

```
------- Trap Receiver 1 ------------------------------------------------

    Trap Receiver Summary
    #  Community  Generation  Authentication  Receiver NMS IP
    ----------------------------------------------------------------------
    1  public     Enabled     Enabled         10.123.56.25
    2  public     Enabled     Enabled         0.0.0.0
    3  public     Enabled     Enabled         0.0.0.0
    4  public     Enabled     Enabled         0.0.0.0

  1- Trap Community Name : public
  2- Trap Generation     : Enabled
  3- Authentication Traps: Enabled
  4- Receiver NMS IP     : 10.123.56.25
  5- Accept Changes      :

  ?- Help
<ENTER> Redisplay Menu
 <ESC> Return To Previous Menu
```

```
>
```

Once again, the first part of the menu is a summary of the trap receiver configuration. We've already set the first trap receiver to the address of our NMS, enabled trap generation, and enabled the generation of authentication trapsas always, a good idea. The traps we generate will include the community string Note that trap receivers 2, 3, and 4 are set to 0.0.0.0. On this menu, 0.0.0.0 is not a wildcard; it's just an invalid address that means you haven't yet configured the trap receiver's IP address. It's basically the same as leaving the entry disabled.

The final configuration items that should be set are on the System submenu, found under the SNMP main menu:

```
------- System ----------------------------------------------------------

     1- sysName        : ups1.ora.com
     2- sysContact     : Douglas Mauro
     3- sysLocation    : Apache Hilo Deck
     4- Accept Changes :

     ?- Help
<ENTER> Redisplay Menu
  <ESC> Return To Previous Menu

>
```

After you have finished configuring all your SNMP parameters, use the Summary submenu for a quick look at what you have done. A typical setup will look something like this:

```
-------------------------------------------------------------------------
        SNMP Configuration Summary

        sysName          : ups1.ora.com
        sysLocation      : Apache Hilo Deck
        sysContact       : Douglas Mauro

        Access Control Summary
        #  Community  Access      NMS IP
        -----------------------------------------------------------------
```

```
1  public     Read        10.123.56.25
2  private    Write       10.123.56.25
3  public2    Disabled    0.0.0.0
4  private2   Disabled    0.0.0.0

Trap Receiver Summary
#  Community  Generation  Authentication  Receiver NMS IP
--------------------------------------------------------
1  public     Enabled     Enabled         10.123.56.25
2  public     Enabled     Enabled         0.0.0.0
3  public     Enabled     Enabled         0.0.0.0
4  public     Enabled     Enabled         0.0.0.0

Press <ENTER> to continue...
```

Upon completion and verification, use the Esc key to take you all the way out to the Logout menu.

PREV NEXT