**An Introduction to SNMP (Simple Network Management Protocol)**

Published on August 18, 2014

* [Networking](https://www.digitalocean.com/community/tags/networking)
* [Monitoring](https://www.digitalocean.com/community/tags/monitoring)
* [Conceptual](https://www.digitalocean.com/community/tags/conceptual)

By [Justin Ellingwood](https://www.digitalocean.com/community/users/jellingwood)

Developer and author at DigitalOcean.



**Introduction**

A large part of being a system administrator is collecting accurate information about your servers and infrastructure. There are a number of tools and options for gathering and processing this type of information. Many of them are built upon a technology called **SNMP**.

SNMP stands for simple network management protocol. It is a way that servers can share information about their current state, and also a channel through which an administer can modify pre-defined values. While the protocol itself is very simple, the structure of programs that implement SNMP can be very complex.

In this guide, we will introduce you to the basics of the SNMP protocol. We will go over its uses, the way that the protocol is typically used in a network, the differences in its protocol versions, and more.

**Basic Concepts**

SNMP is a protocol that is implemented on the application layer of the networking stack (click here to learn about [networking layers](https://www.digitalocean.com/community/tutorials/an-introduction-to-networking-terminology-interfaces-and-protocols#NetworkLayers)). The protocol was created as a way of gathering information from very different systems in a consistent manner. Although it can be used in connection to a diverse array of systems, the method of querying information and the paths to the relevant information are standardized.

There are multiple versions of the SNMP protocol, and many networked hardware devices implement some form of SNMP access. The most widely used version is SNMPv1, but it is in many ways insecure. Its popularity largely stems from its ubiquity and long time in the wild. Unless you have a strong reason not to, we recommend you use SNMPv3, which provides more advanced security features.

In general, a network being profiled by SNMP will mainly consist of devices containing SNMP **agents**. An agent is a program that can gather information about a piece of hardware, organize it into predefined entries, and respond to queries using the SNMP protocol.

The component of this model that queries agents for information is called an SNMP **manager**. These machines generally have data about all of the SNMP-enabled devices in their network and can issue requests to gather information and set certain properties.

**SNMP Managers**

An SNMP manager is a computer that is configured to poll SNMP agent for information. The management component, when only discussing its core functionality, is actually a lot less complex than the client configuration, because the management component simply requests data.

The manager can be any machine that can send query requests to SNMP agents with the correct credentials. Sometimes, this is implemented as part of a monitoring suite, while other times this is an administrator using some simple utilities to craft a quick request.

Almost all of the commands defined in the SNMP protocol (we will go over these in detail later) are designed to be *sent* by a manager component. These include GetRequest, GetNextRequest, GetBulkRequest, SetRequest, InformRequest, and Response. In addition to these, a manager is also designed to *respond to* Trap, and Response messages.

**SNMP Agents**

SNMP agents do the bulk of the work. They are responsible for gathering information about the local system and storing them in a format that can be queried.updating a database called the “management information base”, or **MIB**.

The MIB is a hierarchical, pre-defined structure that stores information that can be queried or set. This is available to well-formed SNMP requests originating from a host that has authenticated with the correct credentials (an SNMP manager).

The agent computer configures which managers should have access to its information. It can also act as an intermediary to report information on devices it can connect to that are not configured for SNMP traffic. This provides a lot of flexibility in getting your components online and SNMP accessible.

SNMP agents respond to most of the commands defined by the protocol. These include GetRequest, GetNextRequest, GetBulkRequest, SetRequest and InformRequest. In addition, an agent is designed to send Trap messages.

**Understanding the Management Information Base**

The most difficult part of the SNMP system to understand is probably the **MIB**, or management information base. The MIB is a database that follows a standard that the manager and agents adhere to. It is a hierarchical structure that, in many areas, is globally standardized, but also flexible enough to allow vendor-specific additions.

The MIB structure is best understood as a top-down hierarchical tree. Each branch that forks off is labeled with both an identifying number (starting with 1) and an identifying string that are unique for that level of the hierarchy. You can use the strings and numbers interchangeably.

To refer to a specific node of the tree, you must trace the path from the unnamed root of the tree to the node in question. The lineage of its parent IDs (numbers or strings) are strung together, starting with the most general, to form an address. Each junction in the hierarchy is represented by a dot in this notation, so that the address ends up being a series of ID strings or numbers separated by dots. This entire address is known as an object identifier, or **OID**.

Hardware vendors that embed SNMP agents in their devices sometimes implement custom branches with their own fields and data points. However, there are standard MIB branches that are well defined and can be used by any device.

The standard branches we will be discussing will all be under the same parent branch structure. This branch defines information that adheres to the MIB-2 specification, which is a revised standard for compliant devices.

The base path to this branch is:

1.3.6.1.2.1

This can also be represented in strings like:

iso.org.dod.internet.mgmt.mib-2

The section 1.3.6.1 or iso.org.dod.internet is the OID that defines internet resources. The 2 or mgmtthat follows in our base path is for a management subcategory. The 1 or mib-2 under that defines the MIB-2 specification.

This is a [great resource for familiarizing yourself with the MIB tree](http://www.alvestrand.no/objectid/1.3.6.1.2.1.html). This particular page represents the connecting nodes at the junction we have been talking about. You can check what is further up and down the tree by checking out the “superior” and “subsidiary” references respectively.

Another similar tool is a [SNMP Object Navigator](http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en&substep=2&translate=Translate&tree=NO) provided by Cisco. This can be used to drill down into the hierarchy to find information you need. A [similar tree](https://support.ipmonitor.com/mibs_byoidtree.aspx) is provided by SolarWinds.

Basically, if we want to query our devices for information, most of the paths will begin with 1.3.6.1.2.1. You can browse the tree interfaces to learn what kind of information is available to query and set.

**SNMP Protocol Commands**

One of the reasons that SNMP has seen such heavy adoption is the simplicity of the commands available. There are very few operations to implement or remember, but they are flexible enough to address the utility requirements of the protocol.

The following PDUs, or protocol data units, describe the exact messaging types that are allowed by the protocol:

* **Get**: A Get message is sent by a manager to an agent to request the value of a specific OID. This request is answered with a Response message that is sent back to the manager with the data.
* **GetNext**: A GetNext message allows a manager to request the next sequential object in the MIB. This is a way that you can traverse the structure of the MIB without worrying about what OIDs to query.
* **Set**: A Set message is sent by a manager to an agent in order to change the value held by a variable on the agent. This can be used to control configuration information or otherwise modify the state of remote hosts. This is the only write operation defined by the protocol.
* **GetBulk**: This manager to agent request functions as if multiple GetNext requests were made. The reply back to the manager will contain as much data as possible (within the constraints set by the request) as the packet allows.
* **Response**: This message, sent by an agent, is used to send any requested information back to the manager. It serves as both a transport for the data requested, as well as an acknowledgement of receipt of the request. If the requested data cannot be returned, the response contains error fields that can be set with further information. A response message must be returned for any of the above requests, as well as Inform messages.
* **Trap**: A trap message is generally sent by an agent to a manager. Traps are asynchronous notifications in that they are unsolicited by the manager receiving them. They are mainly used by agents to inform managers of events that are happening on their managed devices.
* **Inform**: To confirm the receipt of a trap, a manager sends an Inform message back to the agent. If the agent does not receive this message, it may continue to resend the trap message.

With these seven data unit types, SNMP is capable of querying for and sending information about your networked devices.

**Protocol Versions**

The SNMP protocol has gone through many changes since it was first introduced. The initial spec was formulated with RFC 1065, 1066, and 1067 in 1988. By the simple fact that it has been around so long, this version is still widely supported. However, there are many security issues with the protocol, including authenticating in plain text, so its use is highly discouraged, especially when used on unprotected networks.

Work on version 2 of the protocol was initiated in 1993 and offers some substantial improvements on the earlier standard. Included in this version was a new “party-based” security model meant to address the security issues inherent with the prior revision. However, the new model was not very popular because it was difficult to understand and implement.

Because of this, a few “spin-offs” of version 2 were created, each of which kept the bulk of the version 2 improvements, but swapped out the security model. In SNMPv2c, community-based authentication, the same model used in v1, was reintroduced. This was the most popular version of the v2 protocol. Another implementation, called SNMPv2u, uses user-based security, although this was never very popular. This allowed for per-user authentication settings.

In 1998, the third (and current) version of the SNMP protocol entered as a spec proposal. From a user’s perspective, the most relevant change was the adoption of a user-based security system. It allows you to set a user’s authentication requirements as one of these models:

* **NoAuthNoPriv**: Users connecting with this level have no authentication in place and no privacy of the messages they send and receive.
* **AuthNoPriv**: Connections using this model must authenticate, but messages are sent without any encryption.
* **AuthPriv**: Authentication is required and messages are encrypted.

In addition to authentication, an access control mechanism was implemented to provide granular control over which branches a user can access. Version 3 also has the ability to leverage the security provided by the transport protocols, such as SSH or TLS.

**Conclusion**

Now that you have a good idea about how the protocol operates, you have the foundation needed to implement SNMP in your own infrastructure.

In the [next guide](https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-an-snmp-daemon-and-client-on-ubuntu-14-04), we’ll discuss how to install and configure the components necessary to leverage SNMP on your systems.

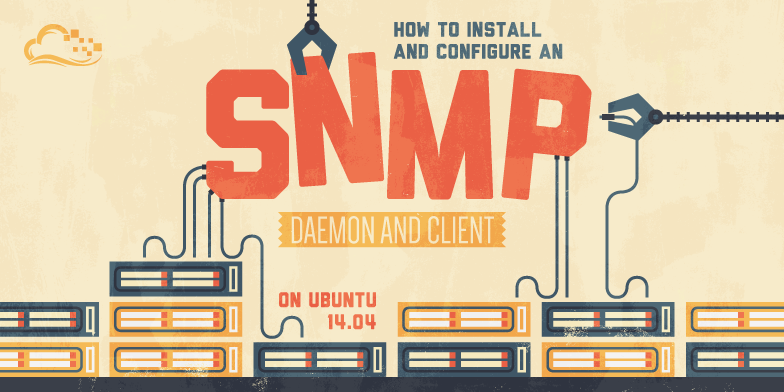
**How To Install and Configure an SNMP Daemon and Client on Ubuntu 14.04**

Published on August 18, 2014

* [Ubuntu](https://www.digitalocean.com/community/tags/ubuntu)
* [Networking](https://www.digitalocean.com/community/tags/networking)
* [System Tools](https://www.digitalocean.com/community/tags/system-tools)

By [Justin Ellingwood](https://www.digitalocean.com/community/users/jellingwood)

Developer and author at DigitalOcean.



**Introduction**

A large part of being a system administrator is collecting accurate information about your servers and infrastructure. There are a number of tools and options for gathering and processing this type of information. Many of them are built upon a technology called **SNMP**.

SNMP stands for simple network management protocol. It is a way that servers can share information about their current state, and also a channel through which an administer can modify pre-defined values. While the protocol itself is very simple, the structure of programs that implement SNMP can be very complex.

In our last guide, we learned about [the basics of the SNMP protocol](https://www.digitalocean.com/community/tutorials/an-introduction-to-snmp-simple-network-management-protocol). In this guide, we will begin to demonstrate how to setup the tools to communicate using SNMP. We will be using two Ubuntu 14.04 servers to demonstrate, but most other systems should be able to follow along with a few modifications.

**Install the SNMP Daemon and Utilities**

We can begin to explore how SNMP can be implemented on a system by installing the daemon and tools on some Ubuntu systems.

We will use two servers, one will contain the manager portion, while the other server will have the agent. We could choose to install the agent on the manager machine as well, but keeping them separate makes it easier to demonstrate what functionality is provided by each component.

On the first server, update the apt database and install the manager component. Along with this, we will also download another package called snmp-mibs-downloader which contains some proprietary information about standard MIBs that allow us to access most of the MIB tree by name:

sudo apt-get update

sudo apt-get install snmp snmp-mibs-downloader

On our second server, the one that we will be interacting with that will run the daemon, we can install the necessary components by typing:

sudo apt-get update

sudo apt-get install snmpd

Now that you have installed these components, we need to configure our setup.

**Configuring the SNMP Manager**

As we mentioned above, most of the bulk of the work happens in the agent component, so our configuration is actually pretty easy on this machine. We just need to modify one file to make sure that our client can use the extra MIB data we installed.

Open the /etc/snmp/snmp.conf file in your text editor with sudo privileges:

sudo nano /etc/snmp/snmp.conf

In this file, there are a few comments and a single un-commented line. To allow the manager to import the MIB files, we simply need to comment out the mibs : line:

#mibs :

Save and close the file when you are finished.

We are now finished configuring the manager portion, but we will still need to use this server to help us configure our agent computer.

**Configuring the SNMP Agent Machine**

As a true client-server system, the agent computer does not have any of the external tools needed to configure its own SNMP setup. We can modify some configuration files to make some changes, but most of the changes we need to make will be done by connecting to our agent server from our management server.

To get started, on our agent computer, we need to open the daemon’s configuration file with sudo privileges:

sudo nano /etc/snmp/snmpd.conf

Inside, we will have to make a few changes. These will mainly be used to bootstrap our configuration so that we can manage it from our other server.

First, we need to change the agentAddress directive. Currently, it is set to only allow connections originating from the local computer. We need to comment out the current line, and uncomment the line underneath, which allows all connections (we will be locking this down soon):

# Listen for connections from the local system only

#agentAddress udp:127.0.0.1:161

# Listen for connections on all interfaces (both IPv4 \*and\* IPv6)

agentAddress udp:161,udp6:[::1]:161

Next, we will need to temporarily insert a createUser line. These directives are not normally kept in this file, but we will be removing it again in a moment, so it doesn’t matter too much.

The user we are creating will be called bootstrap and will be used as a template in which to create our first “real” user. The SNMP packages do this through a process of cloning the user’s properties.

When defining a new user, you must specify the authentication type (MD5 or SHA) as well as supply a passphrase that **must** be at least 8 characters. If you plan on using encryption for the transfer, like we are, you also must specify the privacy protocol (DES or AES) and optionally a privacy protocol passphrase. If no privacy protocol passphrase is supplied, the authentication passphrase will be used for the privacy protocol as well.

Our user creation line will look like this:

createUser bootstrap MD5 temp\_password DES

Now that we have a new user specified, we need to set up the level of access that this user will have. We will set this up for our bootstrap user, and also for the new user we will be creating, called demo. We will allow them read and write access by using the rwuser directive (the alternative is rouser for read-only access).

We will enforce the use of encryption by specifying priv after our user. If we wanted to restrict the user to a specific part of the MIB, we could specify the highest-level OID that the user should have access to at the end of the line.

For our purposes, both of our lines will be fairly simple:

rwuser bootstrap priv

rwuser demo priv

When you are finished making these changes, save and close the file.

To implement these changes, restart the snmpd service:

sudo service snmpd restart

Now, from the machine that you installed the management software on, we can connect to our agent server to create our regular user.

We will do this using the snmpusm tool, which is used for user management. You will need to know the IP address of your agent server for this to function correctly.

Before we begin, we will talk a bit about the general structure of sending an SNMP command.

**The General Structure of SNMP Commands**

When using the suite of tools included in the snmp package (the net-snmp software suite), you will notice a few patterns in the way you must call the commands.

The first thing you must do is authenticate with the SNMP daemon that you wish to communicate with. This usually involves supplying quite a few pieces of information. The common ones are below:

* **-v VERSION**: This flag is used to specify the version of the SNMP protocol that you would like to use. We will be using v3 in this guide.
* **-c COMMUNITY**: This flag is used if you are using SNMP v1 or v2-style community strings for authentication. Since we are using v3-style user-based authentication, we will not be needing this.
* **-u USER-NAME**: This parameter is used to specify the username that you wish to authenticate as. To read or modify anything using SNMP, you must authenticate with a known username.
* **-l LEVEL**: This is used to specify the security level that you are connecting with. The possible values are noAuthNoPriv for no authentication and no encryption, authNoPriv for authentication but no encryption, and authPriv for authentication and encryption. The username that you are using must be configured to operate at the security level you specify, or else the authentication will not succeed.
* **-a PROTOCOL**: This parameter is used to specify the *authentication* protocol that is used. The possible values are MD5 or SHA. This must match the information that was specified when the user was created.
* **-x PROTOCOL**: This parameter is used to specify the *encryption* protocol that is used. The possible values are DES or AES. This must match the information that was specified when the user was created. This is necessary whenever the user’s privilege specification has priv after it, making encryption mandatory.
* **-A PASSPHRASE**: This is used to give the authentication passphrase that was specified when the user was created.
* **-X PASSPHRASE**: This is the encryption passphrase that was specified when the user was created. If none was specified but an encryption algorithm was given, the authentication passphrase will be used. This is required when the -x parameter is given or whenever a user’s privilege specification has a priv after it, requiring encryption.

Using this information, we can begin to construct our commands. Given how we set up our bootstrap user, the commands we will be using with that account will look like this:

snmp\_command -u bootstrap -l authPriv -a MD5 -x DES -A temp\_password -X temp\_password remote\_host snmp\_sub\_command\_or\_options

For instance, from your management server, you can test to make sure your bootstrap account is available by typing:

snmpget -u bootstrap -l authPriv -a MD5 -x DES -A temp\_password -X temp\_password remote\_host 1.3.6.1.2.1.1.1.0

SNMPv2-MIB::sysDescr.0 = STRING: Linux target 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86\_64

The 1.3.6.1.2.1.1.1.0 string is the OID that is responsible for displaying system information. It will basically return the output of uname -a on the remote system.

Now that we have verified that we can correctly authenticate to the server running the SNMP daemon, we can continue on to create our regular user account.

**Set Up the Regular User Account**

Although we have specified the privileges for the demo user account in our snmpd.conf file, we haven’t actually created this user yet. We are going to use the bootstrap user as a template for our new user.

On the management server, we can create the user from the template using the snmpusm tool and the following general syntax:

snmpusm authentication\_info remote\_host create new\_user existing\_user

So, using what we know about the authentication flags we need to pass, and leveraging the user account we already have (bootstrap), we can make a user that fits the user privileges we have already defined (demo).

The command will look like this:

snmpusm -u bootstrap -l authPriv -a MD5 -x DES -A temp\_password -X temp\_password remote\_host create demo bootstrap

You should receive the following message:

User successfully created.

We now have a fully functioning user called demo on our remote server. However, it is still using the same authentication information as the bootstrap account. We should change the password to something else. This time, we will use the demo account to authenticate. Remember, passwords **must** be at least 8 characters long:

snmpusm -u demo -l authPriv -a MD5 -x DES -A temp\_password -X temp\_password remote\_host passwd temp\_password my\_new\_password

You should receive the following message back:

SNMPv3 Key(s) successfully changed.

We can test our new credentials and password by asking our remote server how long the SNMP service has been running. We will use the snmpget command to get a single value from the other machine.

This time, we will take advantage of the extra MIB definitions we downloaded. We can use these to ask for the value by name instead of the OID numeric ID.

snmpget -u demo -l authPriv -a MD5 -x DES -A my\_new\_password -X my\_new\_password remote\_host sysUpTime.0

You should get back a value that represents the last time that the remote SNMP daemon was restarted:

DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (383018) 1:03:50.18

**Creating a Client Configuration File**

You have probably noticed by this point that the authentication details for all of your SNMP commands will be fairly static with each request. Rather than typing these in each time, we can create a client-side configuration file that will contain the credentials we are connecting with.

The client configuration file can be placed in two different locations depending on how wide-spread you wish to share it.

If you want to share your login credentials with any valid user on your management machine, you can place your configuration details into the global snmp.conf file. You would need to open that file with sudo privileges:

sudo nano /etc/snmp/snmp.conf

If, however, you want to define the authentication credentials for your user alone, you can create a hidden .snmp directory within your user’s home directory, and create the file there:

mkdir ~/.snmp

cd ~/.snmp

nano snmp.conf

Regardless of your decision on where to place your configuration, the contents will be the same.

The commands that we are using to authenticate are in the table below. In the right-hand column, you can see the directive names that should be used to set those configuration details within the snmp.conf file:

| **Command Flag** | **Description** | **Translated snmp.conf directive** |
| --- | --- | --- |
| -u USERNAME | The SNMPv3 username to authenticate as. | defSecurityName USERNAME |
| -l authPriv | The security level to authenticate with. | defSecurityLevel authPriv |
| -a MD5 | The authentication protocol to use. | defAuthType MD5 |
| -x DES | The privacy (encryption) protocol to use. | defPrivType DES |
| -A PASSPHRASE | The authentication passphrase for the supplied username. | defAuthPassphrase PASSPHRASE |
| -X PASSPHRASE | The privacy passphrase fro the supplied username. | defPrivPassphrase PASSPHRASE |

Using this information, you can construct an appropriate snmp.conf file. For our guide, this will look like this:

defSecurityName demo

defSecurityLevel authPriv

defAuthType MD5

defPrivType DES

defAuthPassphrase my\_new\_password

defPrivPassphrase my\_new\_password

When you are finished, save and close the file.

Now, you can issue commands without supplying the authentication details. You will only need the SNMP command, the host, and the command arguments.

Instead of typing:

snmpget -u demo -l authPriv -a MD5 -x DES -A my\_new\_password -X my\_new\_password remote\_host sysUpTime.0

We can simply type:

snmpget remote\_host sysUpTime.0

As you can see, this significantly reduces the amount of information we need to supply in each request.

**Removing the Bootstrap Account**

Now that your regular account is configured correctly, we can remove the bootstrap account, since it is fairly insecure.

On your agent server, open the /etc/snmp/snmpd.conf file again with sudo privileges.

Find and comment out (or remove) both of the lines that we previously added that reference the bootstrap user:

#createUser bootstrap MD5 temp\_password DES

#rwuser bootstrap priv

Save and close the file.

Now, restart the SNMP daemon:

sudo service snmpd restart

This will fulfill the recommendation of not having createUser directives in the normal snmpd.conf file. It will also remove privileges from that temporary user.

If you want to completely remove the bootstrap user from the usmUserTable, you can do so by issuing this command from the management server:

snmpusm remote\_host delete bootstrap

You will receive the following response:

User successfully deleted.

**Conclusion**

At this point, you should have a fully configured client-server setup what can communicate securely using the SNMP protocol. You can easily add additional daemons on other hosts and configure account access across your entire infrastructure.

In the next guide, we’ll go over some of [the basic usage of the net-snmp tools](https://www.digitalocean.com/community/tutorials/how-to-use-the-net-snmp-tool-suite-to-manage-and-monitor-servers) that we have been working with. We will demonstrate how to retrieve values one-by-one or by bulk and how to modify data.