This paper examines Simple Network Management Protocol (SNMP). It is intended for readers with minimal prior exposure to SNMP.
Section 1 - A Brief History

Simple Network Management Protocol (SNMP) grew out of an earlier standard, Simple Gateway Management Protocol (SGMP), which only defined a few items such as interfaces type and interface state. A path for SGMP to become a more complete management technology was in the works, but in the mid 1990’s, the inertia shifted to the Internet Engineering Task Force (IETF) SNMP standard. Along with supplying the rudimentary information in SGMP, SNMP also allows the following types of useful system information:

- Interface speed
- System location
- Interface usage
- CPU and memory usage
- Link errors
- Time since last system boot

As well as highly specialized system and service information such as:

- IP routing metrics
- IP telephony configuration
- DHCP service usage
- IP SLA configuration and metrics
- and literally thousands of other parameters

SNMP not only allows the user to view information about managed systems, it also allows for system configuration. SNMP configuration abilities are dependent on the vendor implementation of SNMP on the device. Implementations vary from devices with no SNMP capabilities to devices that are configured and managed entirely by SNMP. Network Management Systems (NMSs) rely heavily on SNMP for exchanging information with managed devices. This includes information on system configuration, status, failures, and performance. Without a good working knowledge of SNMP, it is difficult understand how to properly use the technology.

In recent years internetworking devices have expanded from file sharing systems to automated processing and control systems. SNMP has become rooted into many of these systems that we interact with everyday. These include television broadcast management systems, RFID inventory and sales systems, mobile phone services, electrical power management, accounting and billing systems and medical automation platforms. Because of its relatively simple structure and recent improvements in security, SNMP adoption continues to grow.

Section 2 - SNMP System Structure

At the basic level, an SNMP system consists of three entities, an SNMP manager, a SNMP agent and a Management Information Base (MIB).

- The **SNMP manager** is a software system that communicates using SNMP messages between the manager and **SNMP agents** on managed devices. This is often referred to as a Network Management System (NMS).
- **SNMP agents** are software systems embedded in managed devices. The agent allows the managed device to record data about the device, and communicate with the manager.
- An SNMP MIB is a parsable file, allowing for structured data exchange between the manager and the agent.
These three items and their interactions are depicted below:

![Diagram of SNMP interactions]

The MIB database on the NMS manager is a collection of all the MIBs for all the managed devices. This is described in more detail in the MIBs section below.

Section 3 - SNMP MIBs

While MIBS are central to SNMP functionality, they are often a misunderstood SMNP component. To understand what a MIB is and how it can be used is highly technical, so we’ll cover that first then focus on practical MIB functionality and the rules and guidelines for their use. Communications between computer systems should be as efficient as possible and non-ambiguous. SNMP MIBs meet these two requirements by defining a type of shorthand communication at both ends of the conversation.

An interesting analogy to this is the use of shortwave number stations. If you scan shortwave radio frequencies you will run across some strange communications consisting usually of a single person’s voice, repeating a series of numbers and letters. For example one number station might be transmitting “Yankee, November foxtrot, one, eight, five” repeatedly. To the eavesdropping listener this is meaningless, but if the sender and receiver have a coding/decoding key, then both ends will understand the conversation. While the end goal of number stations is very different than SNMP communications, they both use a coding/decoding key. For SNMP this key is the MIB.
In a very simplified form, SNMP communication using a MIB can be seen in the below diagram.

As long as the manager and the agent have a MIB defining question 2 and the expected format of information exchange, this conversation will work. The actual SNMP request and response will look more like the sample protocol decodes below.

Request:

```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>168</td>
<td>6.323</td>
<td>10.110.66.69</td>
<td>10.110.66.124</td>
<td>SNMP</td>
<td>get-request SNMPv2-MIB::sysUpTime.0</td>
</tr>
</tbody>
</table>
```

```
frame 167 (83 bytes on wire, 82 bytes captured)
  Ethernet II, Src: Vmware.3e:6f:d0 (00:0c:29:3e:6f:d0), Dst: Dell.12:4c:65 (00:1a:a0:12:4c:65)
  Internet Protocol, Src: 10.110.66.124 (10.110.66.124), Dst: 10.110.66.69 (10.110.66.69)
  User Datagram Protocol, Src Port: udp-port (3235), Dst Port: snmp (161)
  Simple Network Management Protocol
    version: v2c (1)
    community: public
    data: get-request (0)
      get-request
        request-id: 9395
        error-status: noError (0)
        error-index: 0
        variable-bindings: 1 item
          SNMPv2-MIB::sysUpTime.0 (1.3.6.1.2.1.1.3.0): unspecified
            Object Name: 1.3.6.1.2.1.1.3.0 (snmpv2-mib::sysUpTime.0)
              Scalar Instance Index: 0
```

Response:

```
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>168</td>
<td>6.323</td>
<td>10.110.66.69</td>
<td>10.110.66.124</td>
<td>SNMP</td>
<td>get-request SNMPv2-MIB::sysUpTime.0</td>
</tr>
</tbody>
</table>
```

```
frame 168 (83 bytes on wire, 87 bytes captured)
  Ethernet II, Src: Dell.12:4c:65 (00:1a:a0:12:4c:65), Dst: Vmware.3e:6f:d0 (00:0c:29:3e:6f:d0)
  Internet Protocol, Src: 10.110.66.69 (10.110.66.69), Dst: 10.110.66.124 (10.110.66.124)
  User Datagram Protocol, Src Port: snmp (161), Dst Port: udp-port (3235)
  Simple Network Management Protocol
    version: v2c (1)
    community: public
    data: get-response (2)
      get-response
        request-id: 9395
        error-status: noError (0)
        error-index: 0
        variable-bindings: 1 item
          SNMPv2-MIB::sysUpTime.0 (1.3.6.1.2.1.1.3.0): 12037748
            Object Name: 1.3.6.1.2.1.1.3.0 (snmpv2-mib::sysUpTime.0)
              Scalar Instance Index: 0
              SNMPv2-MIB::sysUpTime.0: 12037748
```
In the request, the manager at 10.110.66.124 is asking the managed device at 10.110.66.69 for a specific piece of information, the managed systems up time (sysUpTime). In the response the managed system is telling the manager that the sysUpTime is 12037748. Most of the text in the above screens is not part of the actual SNMP conversation but is inserted into the protocol analysis by the protocol analyzer, so that the messages are human readable when decoded. The variable-bindings line in the request indicates the MIB and the object of interest, as well as something called the instance. The format to fully describe the information of interest is MIB.Object_Instance. The first two fields, the MIB and object combined; together create what is known as an Object Identifier (OID). Here is the above MIB request for sysUpTime (1.3.6.1.2.1.1.3.0) broken down into the three components.

<table>
<thead>
<tr>
<th>1.3.6.1.2.1.1</th>
<th>.3</th>
<th>.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RFC1213-MIB, mib-2)</td>
<td>object</td>
<td>instance</td>
</tr>
</tbody>
</table>

MIB structure is analogous to a file system structure with the sysUpTime object being the file we want to retrieve and the objects above that being the directory structure, or folders which store that file. When an SNMP manager asks for the value of 1.3.6.1.2.1.1.3, it is giving the SNMP agent the path to follow in order to find the sysUpTime object.
Structurally, a MIB is a hierarchical ordering of OIDs. Below is what this structure looks like graphically. An object name is made of the numeric (or textual) names along this MIB tree from the top down to the object of interest.

As shown above, MIBs and objects can be described numeric format or textual format. The full textual description of the numeric OID detailed above is iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.

As a side note, the experimental (exp) branch is used for MIBs under development (draft MIBs). When an experimental MIB has been accepted via an RFC, the OIDs are reassigned into the mgmt branch. The directory branch is used for Network Information Services (NIS) OIDs. Both of these branches are beyond the scope of this document. There are also three other entities at the top level, .2 (itu), .3 (joint-iso-itu) at the same level as iso and an unnamed and unnumbered root object above. These branches are also beyond the scope of this document. From here forward, we will focus on the mgmt and private branches.
**MIB Shorthand**

MIBS fall into two general categories, standard MIBS and enterprise MIBS. The split of standard and enterprise MIBs happens directly below the internet level as shown above. When discussing OIDs it’s clumsy to have to start at iso and work your way down the tree to the end object. Because mib-2 can be defined as 1.3.6.1.2.1, the sysUpTime MIB may be referred to as `mib-2.system.sysUpTime` or `mib-2.1.3`. The same is useful in the enterprise branch. For example, if I am interested in Cisco environmental OIDS (voltage, fan state, temperature, etc) I could search down the MIB tree into the enterprise object, then down to the Cisco object (9), and look at all the objects below there for the environmental object. The problem is that there are 21 objects directly below the Cisco object and none of those refer to environmental. I could use an SNMP tool and walk through each of those branches but some of the branches contain thousands of objects.

Instead, I have decided to perform a web search on “Cisco environmental MIB”. There I find a promising link to CISCO-ENVRON-MIB. I click on this and see an option to download the CISCO-ENVRON-MIB. Clicking on this I find the following line, the MIB shorthand to the OID we want:

```
ciscoEnvMonMIB OBJECT IDENTIFIER ::= { ciscoMgmt.13 }
```

The ciscoMgmt branch is not hard to find as it is one of the 21 objects directly below the cisco branch of enterprise. Looking at the MIB browser to the right we can see the path to the CiscoEnvMon MIB.

Starting from the enterprise level the MIB is enterprise.9.9.13. Starting from the cisco level the MIB is cisco.9.13. Normally you would convey this as “cisco enterprise.9.13” or perhaps “cisco management.13”.

The `ciscoEnvMonMIB` object is shown in the MIB browser. The `ciscoEnvMonMIB` object contains sub-objects for environmental monitoring, such as `ciscoEnvMonVoltageStatusTable`, `ciscoEnvMonTemperatureStatusTable`, `ciscoEnvMonFanStatusTable`, and `ciscoEnvMonSupplyStatusTable`. These sub-objects allow for detailed monitoring of environmental conditions.
In the sysUpTime example, the MIB being used is 1.3.6.1.2.1.1 (mib-2), the object is .3 (sysUpTime) and the instance number is 0. The OID is 1.3.6.1.2.1.1.3 (mib-2.system.sysUpTime). The object instance allows for indexed tables for an object. The object described above, system up time, is what is called a scalar object – one that has only a single instance. A single managed device logically has only one value for up time. By default, scalar objects are always marked as instance 0. If we were to request information about the interface traffic on a managed device we would have to specify which interfaces are of interest. This is accomplished by specifying the instance of the OID in the request.

Take a look at the below screen shot of the SolarWinds Engineer's Toolset MIB Browser.

This is a MIB browse of a router at IP 10.199.3.1. What we are seeing is the MIB tree in the left pane and the results of three OID requests on the right pane. The requests are for the interface index table (IfIndex), the interface description table (IfDescr), and the interface in octets table (IfInOctets). Looking in the left pane we can see the tree for the requests expanded down to the interface table area of mib-2. Looking at the top of the right pane we can see the interfaces on the device indexed as indexes 1, 2, 3, 5, 6, 7, and 8. Below that we see the results of the MIB get requests for the names of configured interfaces. Below that we see the number of in octets for each interface (less a shut down interface and a VoIP interface).

So by requesting 1.3.6.1.2.1.2.1.10.1 (IfInOctets.1) we are requesting the current value for the in octets for the FastEthernet0/0 interface. The MIB browser is able to retrieve these values and interpret them because it is acting as an SNMP manager and has a local MIB database. The same is true for the protocol analyzer - it also interprets the SNMP responses using its own MIB database.
Section 4 - SNMP Versions, MIB Versions and Communications

We have seen an example of an SNMP request and response in the top of the MIBs section. Here we'll discuss some of the other methods of exchanging information using SNMP. For SNMP managers and agents to communicate, they need to agree on the message protocol to use, as well as some passwords or community strings to include in messages. One or more managed devices and one or more managers are known as an SNMP community. In SNMP versions 1 and 2 community strings are used as low level passwords to qualify the management community. There are three types of strings used.

- Read-only. Used only to request (read) information.
- Read-write. Used to request information and/or to provide information to be stored (written) by the managed device.
- Trap. The trap community string is optional in many systems today.

Community string security is considered a weak security method as the strings are passed in plain text. SNMP version 3 uses community strings but adds a much more complete and secure authentication. One of the weaknesses in SNMPv1 and v2c is the use of community string security alone. Not only are community strings passed in plain text by SNMPv1 and 2c, but they are often not changed from the read-only (public) and read-write (private) defaults. Because many systems can be configured using SNMP, if a device becomes exposed to the Internet it is simple for a hacker to alter the device to interfere with its operation or gain access to other devices. The main emphasis behind SNMP v3 is hardened security.

SNMP v3 agents and manager have a shared security database for valid users and shared security keys for each user. All SNMP v3 messages use the shared key to create a hash of the message and include the hashed value in the message. The receiver uses the same key to create a hash of the message and if the hashes match the message is validated. Message time checks are also employed to make sure SNMP messages are not being stored and replayed. The variable bindings can also optionally be encrypted. Access control is used to allow for user based access to specific objects as well as read or read-write abilities on an object by object basis. SNMP v3 also allow for the use of management groups so users can be simply added to groups with the proper access.

**SNMP Communications**

SNMP version 1 (SNMPv1) defines five types of messages between the manager and the agent(s).

- From manager to agent these are get, getnext and set.
- From agent to manager they are getresponse and trap.

A get request is a request for a single OID. Getnext requests are similar but allow a manager to ask an agent to send successive responses to the manager. This is particularly useful when the manager needs to retrieve several pieces of similar information such as the utilization of several interfaces on one device. As we saw earlier the interfaces are indexed so the manager can send a series of getnext requests to walk the interface utilization tables. SNMPv1 is widely used today.

SNMP version 2c (SNMPv2c) is the only version of SNMPv2 used today. Except for the trap format, SNMPv2c uses the message types of SNMPv1 and introduces some new message types.

From manager to agent these are getbulk, inform, v2trap and report.

From agent to manager they are getbulkresponse, v2trap, inform and report.
Getbulk was created to address a couple of issues with streams of get and getnext requests. One issue is the amount of network overhead for a large series of get and getnext requests. The second issue is the load multiple sequential requests place on a device. This load can cause an agent to return an error and no data values. Using getbulk, the agent is allowed to send the manager what it is able to send along with information allowing the manager to request remaining values. Inform messages are a lot like trap messages but allow the manager a method to acknowledge that the message was received. SNMP does not provide managers with a method to acknowledge traps. Report messages were implemented for SNMPv2 but were not used because the prescribed security improvements of v2 were scraped when it was made a standard. Report is used in SNMP v3. Getbulkresponse allows for the return of getbulk data requests and information on values that were requested but not returned. V2trap messages are functionally similar to v1 trap but use different encoding.

**MIB versions and SNMP Versions**

MIBs are sometimes referred to as SNMPv1 MIBs or SNMPv2c MIBs. This is actually a misnomer. MIBs are written in one of two Structure of Management Information (SMI) syntaxes, SMIv1 or SMIv2. SMIv1 was introduced about the same time as the messaging protocol SNMP v1, with the same being true for SMIv2 and SNMPv2c. Because of this timing, SMI versions and the SNMP messaging protocols versions are often confused. Data defined in a SMIv2 MIB can be carried by SNMPv1, v2c and v3 protocols with the exception of 64bit counter information, which requires SNMPv2c or 3. The SMI syntax version of a MIB should not be confused with the SNMP messaging protocols. Both SMI MIB versions are widely used. Cisco releases all their MIBS in SMIv1 and SMIv2.

For SNMP communication, messages can be translated from SMIv2 to v1 with the exception of Counter64 data types. SNMPv1 uses UDP port 161 for requests and replies and UDP port 162 for traps. SNMPv2c also uses these UDP ports and defines connection-oriented transport via TCP.
**SNMP Operations**

We have seen an example of an SNMP get request above and discussed the other request types, getnext and get bulk. Here we’ll look at a getbulk request in more detail.

The above is a capture of a getbulk request with some of the more pertinent parts of the request highlighted. These are:

- UDP port 161 being used to send the request.
- SNMPv2c defined.
- Type of request set as getbulk (type 5).
- The request ID used to link responses to the data requested.
- A maximum of 10 data points has been set. This prevents an overload of replies.
- The beginning object requested, enterprises.9.9.109.1.1.1.1.7.
What is being requested is a portion of the ciscoProcessMIB at enterprises.9.9.109. Below is the response.

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The highlighted portions of the reply are:

- UDP port 161 used again for the reply.
- SNMPv2c indicated.
- The request ID of 65045 indicating this data is in reply to the request seen above
- The type of data (gauge 32) and the data value (variable binding) of 91 for the first data point

Nine other data points are directly below this one filling the maximum repetitions indicated in the request. What is happening here is that the manager at 10.110.66.81 is collecting bulk information from the Cisco process MIB in the managed device. Rather than using a request for each OID needed, the manager uses the getbulk and tells the agent where to start and how much to send back. The above is one in a series of getbulk requests retrieving the entire contents of the process MIB.

Getnext is another method of retrieving multiple variable bindings. Managers can send successive getnext requests until the agent returns an end on MIB message. Getbulk is much more efficient.

**SNMP Data Types**

In the above example we saw that the agent returned gauge 32 data. In SNMP communications it is required to indicate the data type as well as the OID value (variable binding). If someone asks you if you are going out to lunch today and you answer is 0, they will probably be confused. The expected data responses in this question are along the lines of “yes” or “no” (although “affirmative” is an acceptable answer from an air traffic controller). SNMP uses several data types as described below.

- **Integer** – Any whole number between – 2147483648 and 2147483648. This data type would be used for status variable bindings such as unknown status = 1, running status = 2, warning status = 3.
- **Integer32** – Basically the same as Integer but a maximum bit size is indicated.
Introduction to SNMP

- **Unsigned32** – Used like the above integer types but includes only non-negative integers only from 0 to 4294967295. Unsigned does not refer to any security method but the fact the this can only represent a positive integer.
- **Gauge** – An unsigned 32 bit integer with range restrictions. These work like a car’s gas gauge. It has a present value and a range limit (empty to full).
- **Gauge32** – Same as Gauge but defines max bit size.
- **Counter** – Unsigned 32 bit integers that “roll over” after reaching the maximum value, sometimes called rolling counters. These work like a car’s odometer when it reaches its maximum value it returns to zero miles and starts again.
- **Counter32** - Same as Counter but defines maximum bit size (32).
- **Counter64** - Same as Counter but defines maximum bit size (64).
- **TimeTicks** – Integer 32 where each tick equally 1/100 second.
- **OCTET STRING** – Data is a string of octets.
- **OBJECT IDENTIFIER** – An OID is specified.
- **IpAddress** – Octet string limited to four octets.
- **NetworkAddress** – Not normally used. IpAddress is used instead.
- **Opaque** – Rarely used, similar to OCTET STRING in function.
- **BITS** – Also rarely used.

The data type is normally inconsequential to network administrators except for the Counter, Counter32 and Counter64 types. These rolling counters are used most commonly for data such as interface utilization. Low speed interfaces or interfaces with little traffic can use 32 bit counters without issue. With busy, high speed interfaces an issue can occur where 32 bit counters roll over or “wrap”. This is normally seen as an SNMP counter returning a lower value than the previous value. All counters will wrap eventually as they have a defined maximum value, but for fast interfaces these wraps may happen too frequently for the resulting data to be useful.

Imagine if your car’s odometer only had two digits for whole miles. By definition, the odometer would wrap every 100 miles. If you tried to calculate your gas mileage by writing down the mileage when you filled the tank, you might get a sequence of numbers for five tanks of gas like 13, 89, 62, 44, and 9. Trying to calculate MPG would be impossible. As interface speeds increased in the mid 1990's the same issue became frequent for high speed interfaces. This is why Counter64 is used.

MIBs not only have to convey the data type to expect but also the units involved. An OID which uses Integer32 data could return a value such as 1215. The MIB will define the units to be associated with the number, such as “seconds” so the manager understands this is 1215 seconds. MIBs also define the maximum access for the OID. Some OIDs are only to be read by managers (read-only), some are also writeable (read-write) and some allow no access (not-accessible). Not accessible OIDs are used by the MIB to store agent values for calculations or such and provide no direct value to a manager.

The data an SNMP agent uses to answer an SNMP request can actually come from a variety of sources. The OID only tells the agent what data is requested. The agent use several methods including reading the data from the agent MIB, retrieving the data via API or calculating the data from multiple sources. Calculated data in an SNMP reply is often called cooked data. An example of cooked data is the five minute average CPU utilization OID for a Cisco router. The router records data points periodically for the CPU utilization and calculates the five minute data. Non-calculated data is often referred to as raw data.
Section 5 - SNMP Traps

The previous sections have described SNMP information which is requested by the SNMP manager, or the NMS. NMS’s use polling cycles to schedule regular SNMP requests and record the results. This polling method is very useful for determining trends or usage patterns but the data is only received when requested. Imagine if the local fire department were to call each house and business on a regular schedule and ask if there was a fire at that location. Obviously this is the wrong way to collect this data. SNMP allow for the equivalent of a 9-1-1 call using SNMP trap messages. Traps are used to communicate exception events from the managed device to the manager.

Managed devices have trap MIBs with defined trap conditions. When using SNMP traps, it is critical that the manager has compiled MIBs for any device that may send a trap. Remember that the MIB is a decoding key for SNMP communications so if a device sends a trap and the manager does not have the MIB for that device, the manager will have no way to interpret the event triggering the trap. The SNMPv2-MIB and the IF-MIB contain some standard trap definitions, also called generic traps. The definitions include cold start, warm start, link down, link up, and EGP neighbor loss. Below is an example of a cold start trap.

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10:11:00.00</td>
<td>10.110.00.09</td>
<td>10.110.00.74</td>
<td>SNMP</td>
<td>get-response trap SNMPv2-MIB::snmpTraps 94</td>
</tr>
</tbody>
</table>

This trap is telling the manager that the device cold started and by providing the sysUpTime the manager can calculate the time the device came back up. The managed device must be configured to send traps to the manager(s). Traps are sometimes called events or alarms but this terminology is incorrect. Events may include traps and traps may trigger alarms but the three are not equivalent.
Section 6 – Review

- SNMP has three primary components:
  - Managers
  - MIBs
  - Agents

- MIBS are used as a key to define SNMP message content.
- SNMP messages are encoded in SMIv1 or SMIv2 syntax. The syntax numbers are independent of SNMP version numbers.
- Managers are often called NMS’s (Network Management Systems).
- Enumerated names for managed objects are called Object Identifiers (OIDs).
- Tabular objects are identified by an OID index.
- Agents are software residing on managed devices and contain MIBs.
- Managers have copies of all MIBs used by the managed devices in the manager’s MIB database.
- SNMP managers use two methods of obtaining SNMP data, polling and trap reception.
- Traps are unsolicited messages sent from agents to managers.
- Managers can use several methods to request information the most common being get, getnext and getbulk.
- Community strings are used as a weak authentication method in SNMPv1 and v2c.
- SNMP v3 adds several security features including message encryption.
- Agents send the requested data (variable binding) along with the data type. Common data types are Counter, Counter32, Counter64 and Gauge32.
Related SolarWinds Products

SolarWinds Orion Network Performance Monitor
Orion NPM enables you to quickly detect, diagnose, and resolve network outages and performance issues. It offers network-centric views that are designed to deliver the critical information you need most. And Orion NPM is the easiest product of its kind to use and maintain, meaning you will spend more time managing networks, not supporting Orion NPM. NPM:

- Monitors and analyzes real-time, in-depth, network performance statistics for routers, switches, wireless access points, servers, and any other SNMP-enabled devices
- Simplifies network issue investigation with drill down maps and Top 10 views of your global network
- Gets you up and running in less than an hour with do-it-yourself deployment
- Scales to accommodate growth and management needs with a hot standby engine, multiple polling engines, and additional web servers
- Enables advanced alerting for correlated events, sustained conditions, and complex combinations of device states
- Monitors the energy consumption of Cisco® EnergyWise-enabled network devices and displays policies that regulate energy consumption
- Extends management capabilities to NetFlow traffic analysis, IP SLA monitoring, IP address management, network configuration management, and application and server performance

http://www.solarwinds.com/products/orion

SolarWinds Engineer’s Toolset
Once you’ve experienced Engineer’s Toolset, you’ll never troubleshoot your network the same way again. It includes a collection of powerful network management tools, all of which can be easily accessed through the new Workspace Studio to quickly resolve issues right from your desktop!

With Engineer’s Toolset version 10:

- Cut troubleshooting time in half using the Workspace Studio, which puts the tools you need for common situations at your fingertips.
- Monitor and alert in real time on network availability and health with tools including Real-Time Interface Monitor, SNMP Real-Time Graph, and Advanced CPU Load. View Monitoring Tools
- Perform robust network diagnostics for troubleshooting and quickly resolving complex network issues with tools such as Ping Sweep, DNS Analyzer, and Trace Route. View Diagnostic Tools
- Deploy an array of network discovery tools including Port Scanner, Switch Port Mapper, and Advanced Subnet Calculator. View Discovery Tools
- Manage Cisco® devices with specialized tools including Real-time NetFlow Analyzer, Config Downloader, and Config Compare. View Cisco Tools

http://www.solarwinds.com/products/toolsets/
About SolarWinds

SolarWinds is rewriting the rules for how companies manage their networks. Guided by a global community of network engineers, SolarWinds develops simple and powerful software for managing networks, small or large. Our company culture is defined by passion for innovation and a philosophy that network management can be simplified for every environment.

SolarWinds products are used by more than one million network engineers to manage IT environments ranging from ten to tens of thousands of network devices. Comprised of fault and performance management products, configuration and compliance products, and tools for engineers, the SolarWinds product family is trusted by organizations around the globe to design, build, maintain, and troubleshoot complex network environments.

SolarWinds is headquartered in Austin, Texas, with sales and product development offices around the world. Join our online community of experts at thwack.com!

About the Author

Andy McBride is a Technical Specialist for SolarWinds focusing on making knowledge of networking and network management accessible to customers and prospects of all levels. The “New to Networking” series is specifically written for an audience with limited prior exposure to these technologies. His technical background includes seven years at International Network Services (INS) as a Network Engineer and Managing Consultant, three years as a Novell Certified Instructor and five years as a Network Performance Products Manager with BT-Infonet. Prior to entering technology, Andy worked in aerospace on projects such as the SR-71, F-117, F-22, L-1011, F-18, and the space shuttle main engine and has a degree in Chemistry from California State University Northridge. Go Matadors!