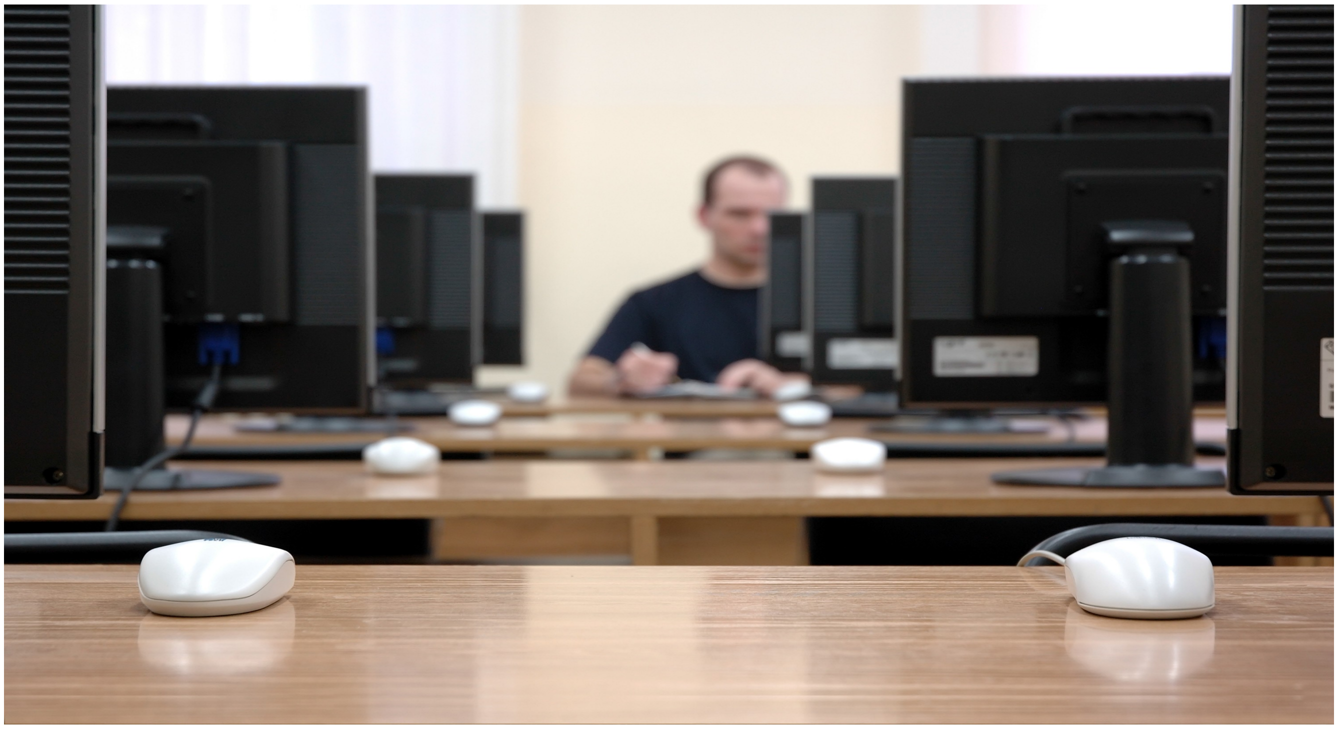
Nagios
Start Up Guide



Nagios is the industry standard for monitoring network infrastructure. The Nagios Start Up Guide provides the foundation for installation and initial configuration of Nagios. In addition, instruction on the monitoring of Linux and Windows machines is provided.

Date of Manual Version: October 5, 2017

Table of Contents

Introduction

Nagios is both a powerful and flexible tool for monitoring devices and applications on those devices. The power of Nagios is in the ability to monitor many different network devices at one time using various methods to monitor those devices. The flexibility of Nagios provides an administrator the tools to monitor just about anything that is connected to a network. In addition, Nagios allows the administrator to monitor both the internals and the application processes on those devices. Monitoring would not be complete without multiple methods for contacting administrators which Nagios also provides.

Nagios Monitoring Solutions

Nagios Core is the foundational application that provides the monitoring and alerting options that Nagios is known for. Nagios Core contains the architecture that enables flexibility in monitoring and in extending the capabilities of Nagios with other applications. The flexibility of Nagios Core allows you to use it to perform and schedule checks, perform event handling and alert administrators as needed. The Nagios web interface which uses CGI by default can be modified to use a MySQL database as the back-end. The front-end can be modified with custom options to provide the look and feel that an organization needs. Nagios Core by design features and supports many different addons that can be used with it. Nagios Core is an OpenSource Software licensed under the GNU GPL V2.

Nagios XI takes the Nagios Core and builds upon it to create an enterprise-class monitoring and alerting solution that is easier to set up and configure. Nagios XI through easy to use network wizards provides infrastructure monitoring of all of an organizations critical hardware, applications, network devices and network metrics. The dashboard feature allows you to view the entire infrastructure visually as you monitor all of these services and devices. You also have the alerting options which communicate to administrators when services and hosts have problems. The trending and hardware capacity limits help you create proactive decisions about the network and devices on the network. The graphical interface is easy to customize to fit the organization needs and by monitoring the graphs will help you predict network, hardware and application problems.

The major differences between these monitoring solutions is that Nagios Core is an OpenSource Software that can be configured manually to perform some of the functions of Nagios XI, but it must be configured from the command line and does not provide the easy to use GUI, configuration wizards or advanced reporting capabilities that XI does. If you are looking for an easy to use and set up interface, Nagios XI is it.

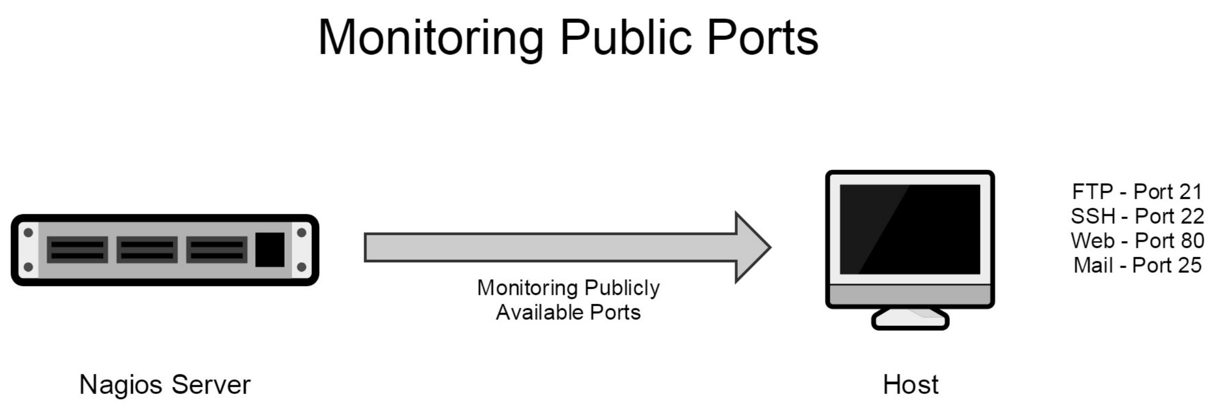
Critical Decisions

These 4 elements represent turning points in how you implement Nagios. Each turning point represents a decision that has implications in how Nagios is used.

1. **Monitor Public Information vs. Private Information**

In order to provide instructions that work across multiple Linux distributions, the Nagios Start Up Guide provides documentation for compiling Nagios and Nagios plugins.

Public ports are ports that are accessible to anyone, like ports for a web server (80), FTP server (21) or mail server (25). When a service is started on a server that is a public service, everyone has access to the public port unless firewall rules prevent it.



The caveat of monitoring public ports is that they are easy to monitor (little to no configuration required) but may not provide all of the detail that is desired. The Nagios server firewall is completely blocked to incoming traffic unless it is related to a connection established by the Nagios server. This means you have greater security for a Nagios server that is accessible from the Internet. It also means the client being monitored is more secure as special access for the Nagios server does not have to be added. In summary, monitoring public ports offer greater security but less information.

In contrast, monitoring internal aspects of a machine requires an *agent* to be installed on the client. The only way to monitor internal aspects of a machine is to install an *agent*, meaning a piece of software that functions as a daemon allowing connections from the Nagios server so that internal plugins or scripts may be executed and that information recorded and provided to the Nagios server on connection. Here are several *agents* that can be used:

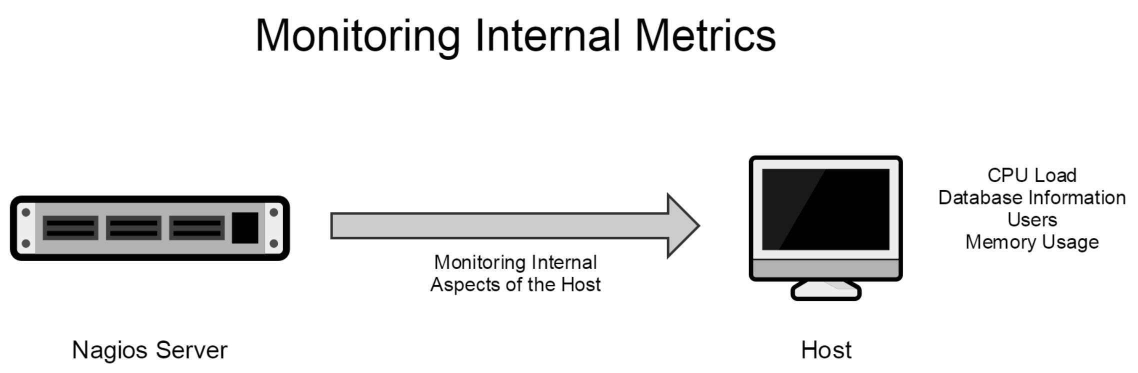
SSH – the daemon allows connections from the Nagios server and returns information generated by plugins or scripts.

NSClient++ – this agent is installed on a Windows server or workstation so that commands executed on the Windows machine can generated information and return that information to the Nagios server when the Nagios server connects to the Windows machine.

NRPE (Nagios Remote Plugin Executor) – The NRPE agent is installed on the remote machine to allow Nagios to connect and obtain information generated by plugins that have executed internally or scripts that have executed.

NCPA (Nagios Cross Platform Agent) – The NCPA agent is installed on the remote machine to allow Nagios to execute plugins and get responses as if it were acting locally.

check_mk – The check_mk agent must be added to the Windows or Linux box in order for check_mk to collect the information and provide it to the Nagios server.

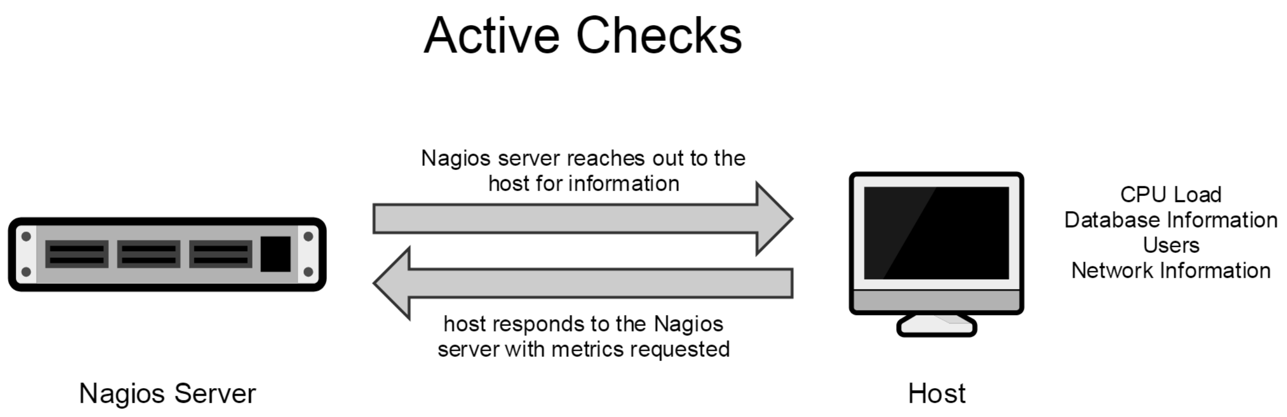

Monitoring Internal Metrics
Monitoring Internal Aspects of the Host
CPU Load
Database Information
Users
Memory Usage
Nagios Server
Host

In summary, monitoring internal aspects of a machine provides greater information but requires an *agent* to be installed as well as security to be altered to allow the Nagios server to connect.

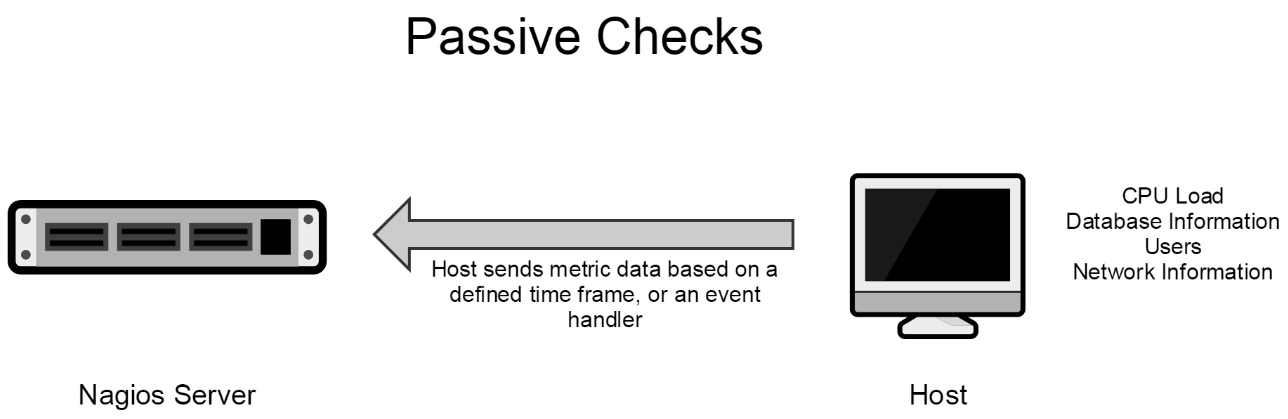
2. Active vs. Passive Checks

There are a few things to consider when deciding whether to monitor your infrastructure with active or passive checks. **Note:** you can use both if you choose.

Active checks are initiated by the Nagios server. This method is also known as polling. The Nagios server determines the time and the frequency for the checks. Active checks that use only public ports require no modification. However, when internal aspects are monitored the server that is monitored must not only have an *agent* installed to initiate any internal plugins or scripts on the machine but the firewall will need to be modified to allow Nagios to connect on the agent ports.


Active Checks
Nagios server reaches out to the host for information
host responds to the Nagios server with metrics requested
CPU Load
Database Information
Users
Network Information
Nagios Server
Host

Passive checks require that the server to be monitored not only initiates the script to check internal aspects but also initiate the connection to the Nagios server. This is the type of connection you would want to use if a security event occurred on the server that is monitored. Any security events would require immediate notification to minimize the impact.

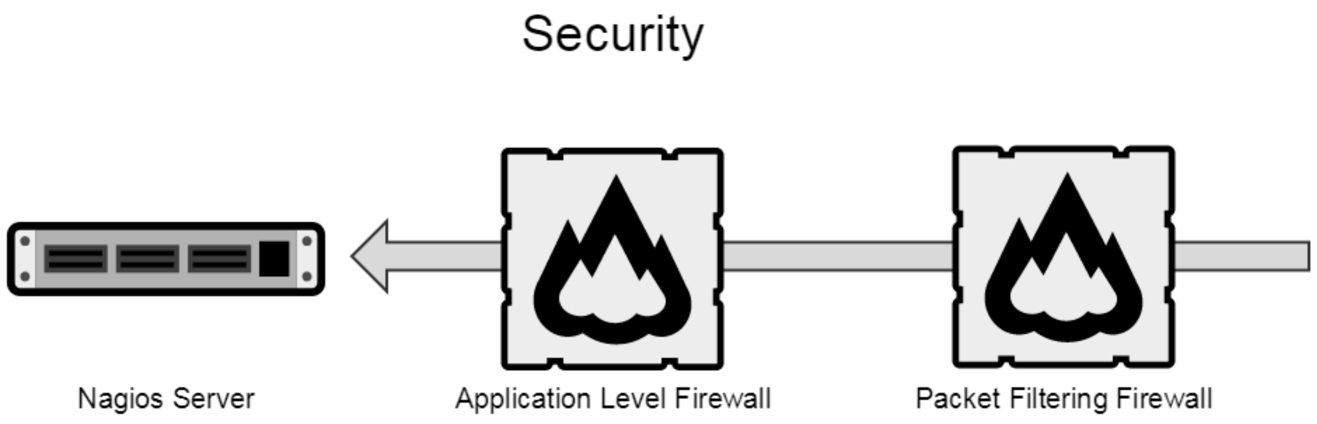


3. Internet Security Measures

Many administrators do not use a firewall on the Nagios server as it is on an internal network, or VPN, so it is not considered necessary. However, we recommend that Nagios servers always have a firewall in place, as a compromised Nagios server can compromise the rest of the network it is monitoring.

If a Nagios server is accessible from the Internet, security should be carefully considered. Several aspects of security must be implemented:

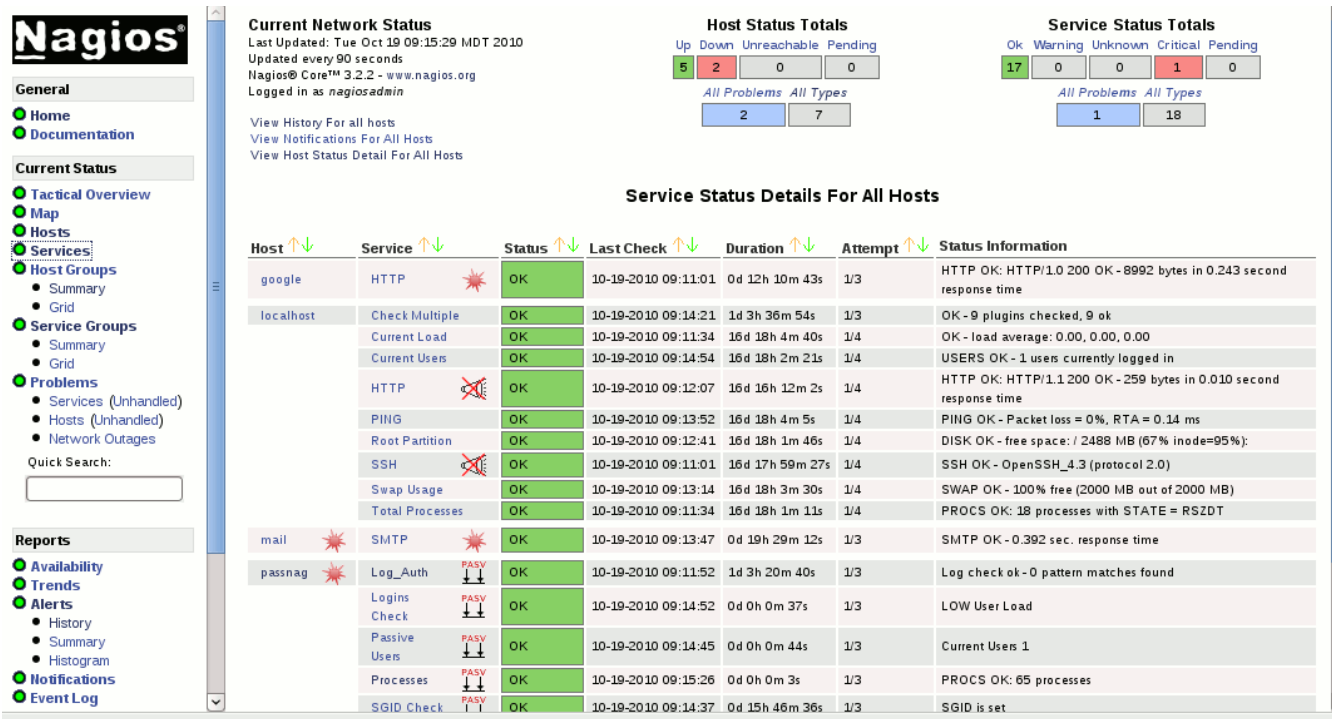
* limited access to the web interface
* limited access to SSH or any other methods of connecting to the server
* SSL for password authentication
* ModSecurity to protect authentication, SQL injection and limit access



4. Nagios Core vs. XI

Nagios Core is the Open Source version of Nagios which can be freely downloaded and implemented in any way the administrator sees fit. The Nagios Core is extremely flexible and provides you access to create, configure and implement at will.

One of the biggest advantages of Nagios Core is that by the time you get plugins working you are able to troubleshoot when problems occur. This advantage cannot be overstated as becoming a "Nagios mechanic" is an extremely valuable asset to any organization. Nagios Core allows the administrator almost unlimited abilities in creating and monitoring devices and services on those devices. Core provides the structure to implement all of the aspects of XI but to do it in a way that fits the organization.



Current Network Status
Last Updated: Tue Oct 19 09:15:29 MDT 2010
Updated every 90 seconds
Nagios® Core™ 3.2.2 - www.nagios.org
Logged in as *nagiosadmin*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
5	2	0	0

All Problems	*All Types*
2	7

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
17	0	0	1	0

All Problems	*All Types*
1	18

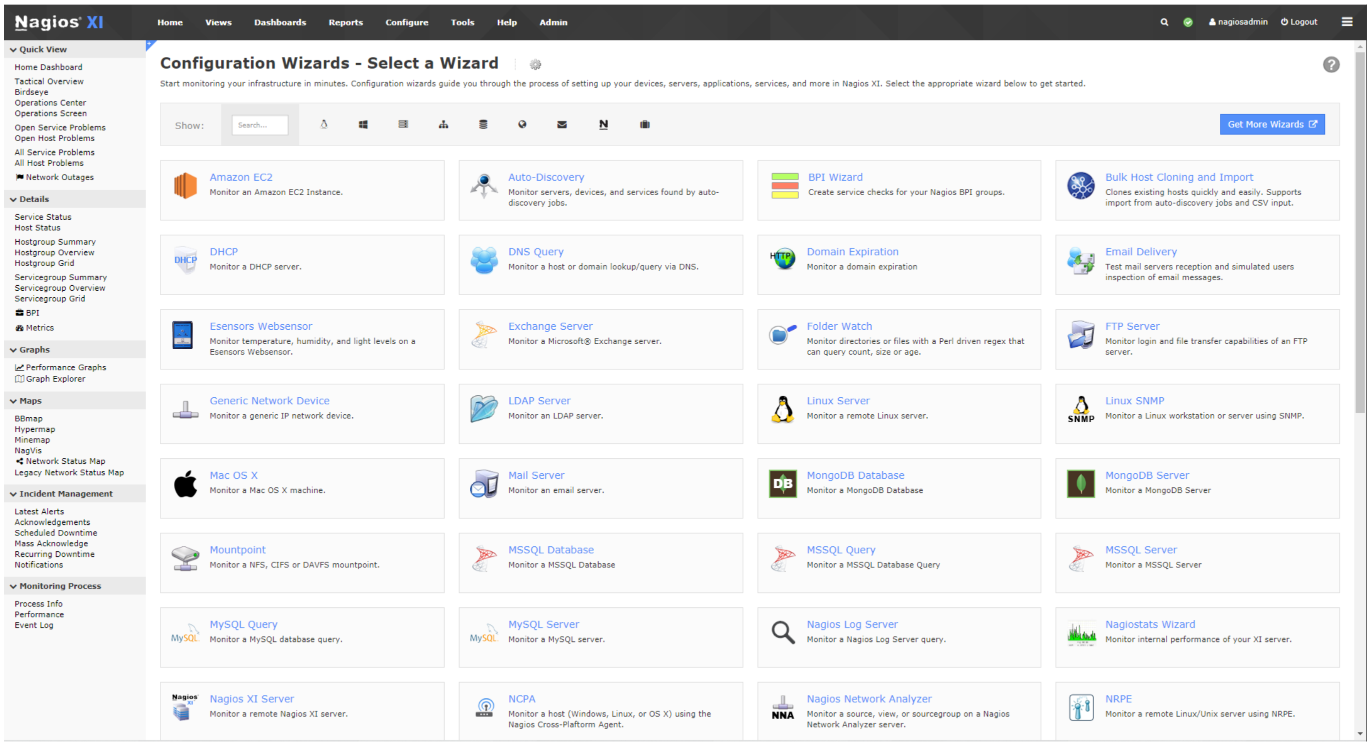
Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
google	HTTP	OK	10-19-2010 09:11:01	0d 12h 10m 43s	1/3	HTTP OK: HTTP/1.0 200 OK - 8992 bytes in 0.243 second response time
localhost	Check Multiple	OK	10-19-2010 09:14:21	1d 3h 36m 54s	1/3	OK - 9 plugins checked, 9 ok
	Current Load	OK	10-19-2010 09:11:34	16d 18h 4m 40s	1/4	OK - load average: 0.00, 0.00, 0.00
	Current Users	OK	10-19-2010 09:14:54	16d 18h 2m 21s	1/4	USERS OK - 1 users currently logged in
	HTTP	OK	10-19-2010 09:12:07	16d 16h 12m 2s	1/4	HTTP OK: HTTP/1.1 200 OK - 259 bytes in 0.010 second response time
	PING	OK	10-19-2010 09:13:52	16d 18h 4m 5s	1/4	PING OK - Packet loss = 0%, RTA = 0.14 ms
	Root Partition	OK	10-19-2010 09:12:41	16d 18h 1m 46s	1/4	DISK OK - free space: / 2488 MB (67% inode=95%):
	SSH	OK	10-19-2010 09:11:01	16d 17h 59m 27s	1/4	SSH OK - OpenSSH_4.3 (protocol 2.0)
	Swap Usage	OK	10-19-2010 09:13:14	16d 18h 3m 30s	1/4	SWAP OK - 100% free (2000 MB out of 2000 MB)
	Total Processes	OK	10-19-2010 09:11:34	16d 18h 1m 11s	1/4	PROCS OK: 18 processes with STATE = RSZDT
mail	SMTP	OK	10-19-2010 09:13:47	0d 19h 29m 12s	1/3	SMTP OK - 0.392 sec. response time
passnag	Log_Auth PASV	OK	10-19-2010 09:11:52	1d 3h 20m 40s	1/3	Log check ok - 0 pattern matches found
	Logins Check PASV	OK	10-19-2010 09:14:52	0d 0h 0m 37s	1/3	LOW User Load
	Passive Users PASV	OK	10-19-2010 09:14:45	0d 0h 0m 44s	1/3	Current Users 1
	Processes PASV	OK	10-19-2010 09:15:26	0d 0h 0m 3s	1/3	PROCS OK: 65 processes
	SGID Check PASV	OK	10-19-2010 09:14:37	0d 15h 46m 36s	1/3	SGID is set

The greatest disadvantage of Nagios Core is developing the necessary skills to monitor the devices and the various aspects of those devices. It takes a great deal of time and meticulous implementation of scripts and plugins to arrive at the working Nagios Core monitoring system that many organizations need. This is compounded by the fact that it is often hard to determine how long it will take to get an implementation up and running before attempting to do so, meaning staff requirements are underestimated.

The XI interface is the commercial version that provides wizards for setting up the plugins making it much easier to set up quickly and also implement checks that you may not totally understand how they work.

The greatest advantages of XI are that it gets an organization up and running quickly, and provides many advanced reporting capabilities that Core does not, such as capacity planning and SLA reports. The wizards work very well and provide intuitive support in implementing NRPE, SNMP, NSCLient++, etc. However, this is not an automatic set up as some basic information and understanding are required.



Configuration Wizards - Select a Wizard

Start monitoring your infrastructure in minutes. Configuration wizards guide you through the process of setting up your devices, servers, applications, services, and more in Nagios XI. Select the appropriate wizard below to get started.

Amazon EC2
Monitor an Amazon EC2 Instance.

Auto-Discovery
Monitor servers, devices, and services found by auto-discovery jobs.

BPI Wizard
Create service checks for your Nagios BPI groups.

Bulk Host Cloning and Import
Clones existing hosts quickly and easily. Supports import from auto-discovery jobs and CSV input.

DHCP
Monitor a DHCP server.

DNS Query
Monitor a host or domain lookup/query via DNS.

Domain Expiration
Monitor a domain expiration

Email Delivery
Test mail servers reception and simulated users inspection of email messages.

Esensors Websensor
Monitor temperature, humidity, and light levels on a Esensors Websensor.

Exchange Server
Monitor a Microsoft® Exchange server.

Folder Watch
Monitor directories or files with a Perl driven regex that can query count, size or age.

FTP Server
Monitor login and file transfer capabilities of an FTP server.

Generic Network Device
Monitor a generic IP network device.

LDAP Server
Monitor an LDAP server.

Linux Server
Monitor a remote Linux server.

Linux SNMP
Monitor a Linux workstation or server using SNMP.

Mac OS X
Monitor a Mac OS X machine.

Mail Server
Monitor an email server.

MongoDB Database
Monitor a MongoDB Database

MongoDB Server
Monitor a MongoDB Server

Mountpoint
Monitor a NFS, CIFS or DAVFS mountpoint.

MSSQL Database
Monitor a MSSQL Database

MSSQL Query
Monitor a MSSQL Database Query

MSSQL Server
Monitor a MSSQL Server

MySQL Query
Monitor a MySQL database query.

MySQL Server
Monitor a MySQL server.

Nagios Log Server
Monitor a Nagios Log Server query.

Nagiostats Wizard
Monitor internal performance of your XI server.

Nagios XI Server
Monitor a remote Nagios XI server.

NCPA
Monitor a host (Windows, Linux, or OS X) using the Nagios Cross-Plaftorm Agent.

Nagios Network Analyzer
Monitor a source, view, or sourcegroup on a Nagios Network Analyzer server.

NRPE
Monitor a remote Linux/Unix server using NRPE.

In summary, if an organization is short on time and has the resources, XI is a great choice. On the other hand, if an organization has the time and the right people the results can be more productive and probably save money in the long run.

Conclusion:
Turning points determine many significant decisions. Give careful thought to how these turning points are selected and consider the long term because it is easier to implement an installation by planning ahead

Nagios Terminology

plugins
Nagios uses plugins, or compiled executables that can used to check services and hosts on your network. Plugins can be developed using Perl, shell scripts, etc. Plugins provide communication between the Nagios daemon and the hosts and service options you want to check. There are many different plugins available. Each plugin must be configured specifically for the host and service you choose to evaluate. Plugins do not come in the nagios package but are provided in a separate package called nagios-plugins. You can download from these locations.

Nagios Plugins

Official Nagios Plugins	http://nagiosplugins.org/
Nagios Plugin Downloads	http://www.nagios.org/download/
NagiosExchange	http://exchange.nagios.org/

Currently the plugins provided in the nagios-plugins package provides about 70 plugins. This certainly provides you with adequate plugins to get started.

If you need to find out more information about a specific plugin you can use this command:

```
    ./check_ping -help
check_ping v1.4.15 (nagios-plugins 1.4.15)
```

```
Copyright (c) 1999 Ethan Galstad <nagios@nagios.org>
Copyright (c) 2000-2007 Nagios Plugin Development Team
      <nagiosplug-devel@lists.sourceforge.net>

Use ping to check connection statistics for a remote host.

Usage:
check_ping -H <host_address> -w <wrta>,<wpl>% -c <crta>,<cpl>%
 [-p packets] [-t timeout] [-4|-6]

Options:
 -h,--help
    Print detailed help screen
 -V, --version
    Print version information
 -4, --use-ipv4
    Use IPv4 connection
 -6, --use-ipv6
    Use IPv6 connection
 -H, --hostname=HOST
    host to ping
 -w, --warning=THRESHOLD
    warning threshold pair
 -c, --critical=THRESHOLD
    critical threshold pair
 -p, --packets=INTEGER
    number of ICMP ECHO packets to send (Default: 5)
 -L, --link
    show HTML in the plugin output (obsoleted by urlize)
 -t, --timeout=INTEGER
    Seconds before connection times out (default: 10)
```

host
A host is a server, switch, router, printer or any other network device that you want to monitor. Nagios requires an IP Address for the host or a FQDN (Fully Qualified Domain Name) to determine the exact location of the device. Each host must also have a unique name that will tie the host name reference to the IP Address of FQDN. The host information is required for the service definition.

service
Services refer to checks that occur on a device which may monitor internal aspects of the device like CPU usage or memory and also refer to checks on applications which exist on the device such as MySQL or Postfix.

contact
Contacts are the individual administrators that are notified by Nagios because of a host or service problem using the contactgroup. The contact information provides a way to communicate to the administrator. Contacts is also a way to manage which administrators can see hosts and services on the web interface as they must be listed as contacts in order to view specific information on devices.

contactgroup
These groups are the connection between detected problems and communications with individuals in the group.

Reachability
Nagios has the ability to determine if a host is in a down state or if it is in an unreachable state. The practical implications of both of these states is the same, stuff does not work. However, the troubleshooting aspect is quite different. If a host is down, then of course the administrator needs to investigate the host specifically. However, if a network device is down or so heavily loaded it restricts communication then the network administrator needs to focus

on the network devices and related issues. So reachability is concerned with the overall network health and how it impacts your monitored hosts.

Nagios is able to discern the network structure and how it alters these down states and unreachable states by understanding the path for data packets on the network. In other words, Nagios needs to know how equipment is connected because that will help determine the situation. This is done by making a reference to the parent/child relationships of connected network devices.

Nagios needs to be able to start tracing the path of the data packet from the Nagios server (hostname) to the next device and the next device. So the first step in setting this up is creating an entry for Nagios in the hosts.cfg file.

```
define host{
        host_name       nagios
}
```

Of course you want to provide the hostname of your Nagios server which you can determine with the command:

```
hostname
```

The next step in configuration is to look at the IP Address and hostname of the next network device. If Nagios is connected to a switch, that switch should be also configured with a host definition. The difference is that you want to tell Nagios that the parent of that switch device is the hostname nagios.

```
define host{
      host_name  ciscoswitch
      parents    nagios
}
```

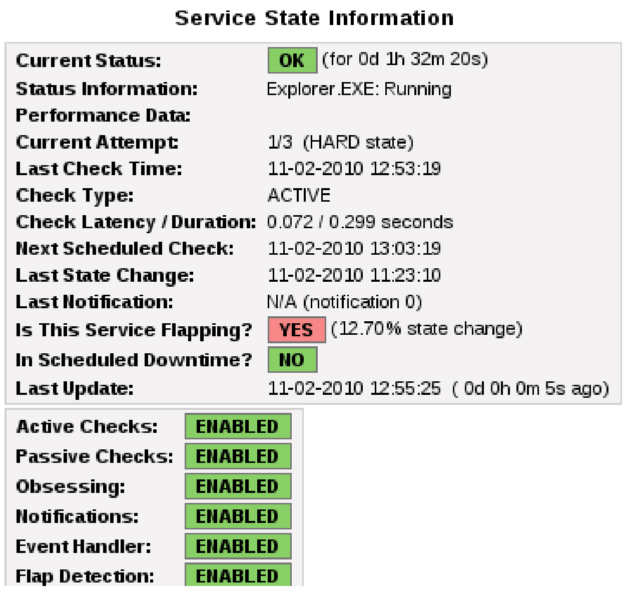
You can only add devices that have the ability to be assigned an IP Address and/or a hostname.

The key in the design is recognizing the network configuration and telling Nagios which is the parent, or network device, directly above the host you are working with. Once your data packet hits your external interface on your router you cannot specify routers on the Internet as the path will vary depending upon best route. So if you were tracing the data packet path from Nagios to a remote device you would need to indicate the IP Address of the external router connecting the device to the Internet.

Volatile Service
A volatile service is a service that will automatically return itself to an "OK" status when it is checked. Or it is a service that needs to be checked by an administrator on each occurrence, like a security event.

flapping
A flapping state is when a service or host changes from an OK state to CRITICAL state rapidly. These changing states will send multitudes of notifications to administrators which can be non-productive. When flapping is detected Nagios will recognize the changing states and move into a state of flapping which provides additional options for an administrator which could allow unwanted notifications.

In order to detect this flapping state Nagios saves in memory 21 checks for each host and service. Nagios reviews the last 20 changes to determine if the host or service is changing states based on a percentage. In this review of states the more recent checks are provided a greater weight than the older checks as this is probably more important to an administrator. Nagios also provides two thresholds for a service and a host so that an administrator can set an upper and lower threshold which means that when the service or host goes above the upper threshold Nagios recognizes this as state flapping which means notifications will be stopped, an entry in the log is created and a comment is placed in the web interface so it can be reviewed by administrators. Once the percentage goes below the lower limit the comment is removed and the service is returned to a normal state with notifications enabled. This process takes a period of time or occur. Here is an example of a service that is flapping. If you look closely you can see the percentage of state change.


Service State Information

Current Status:	OK (for 0d 1h 32m 20s)
Status Information:	Explorer.EXE: Running
Performance Data:	
Current Attempt:	1/3 (HARD state)
Last Check Time:	11-02-2010 12:53:19
Check Type:	ACTIVE
Check Latency / Duration:	0.072 / 0.299 seconds
Next Scheduled Check:	11-02-2010 13:03:19
Last State Change:	11-02-2010 11:23:10
Last Notification:	N/A (notification 0)
Is This Service Flapping?	YES (12.70% state change)
In Scheduled Downtime?	NO
Last Update:	11-02-2010 12:55:25 (0d 0h 0m 5s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Notifications for this service are being suppressed because it was detected as having been flapping between different states (24.4% change >= 20.0% threshold). When the service state stabilizes and the flapping stops, notifications will be re-enabled.

To make changes to the settings for flap detection, first access the nagios.cfg file which provides global settings. The first setting that can be altered is that an administrator can turn flapping off by changing the value to “0”. The thresholds may be modified to meet specific requirements for the organization. Remember these thresholds are percentages so the low end is 5%, or one state change and the upper end is 20% which equals five state changes.

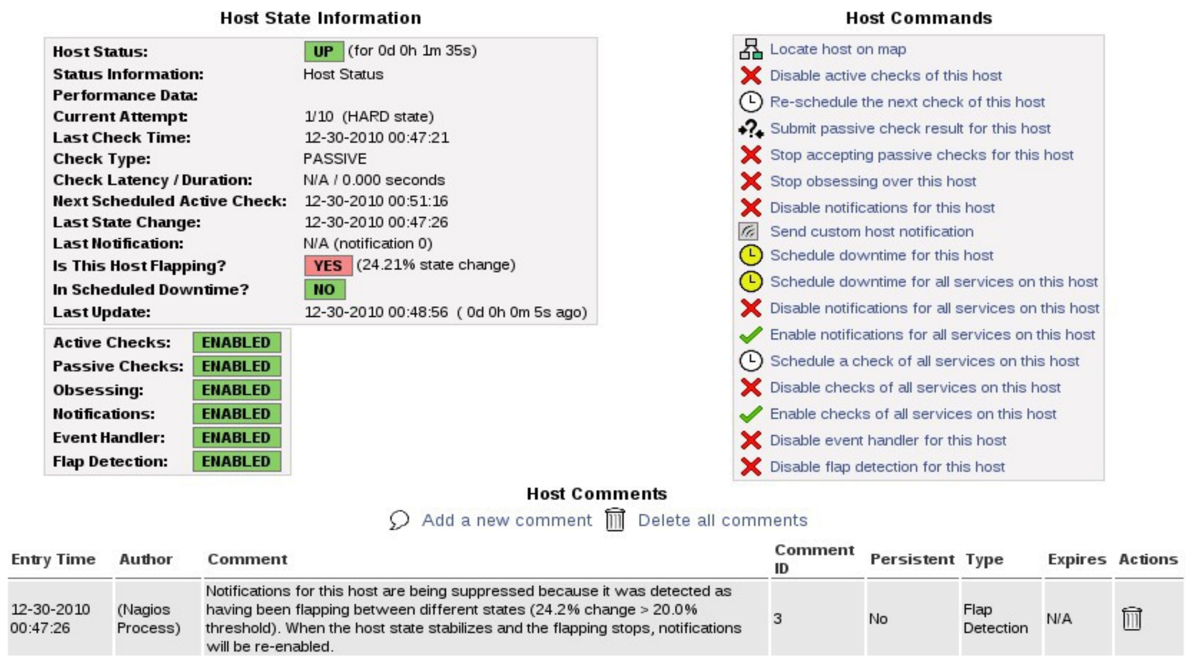
```
enable_flap_detection=1

low_service_flap_threshold=5.0
high_service_flap_threshold=20.0
low_host_flap_threshold=5.0
high_host_flap_threshold=20.0
```

Specific changes could be made with the specific service as well. The “flap_detection_enabled” must be included to allow the override of the global settings. The two thresholds then may be modified to meet the needs of the service.

```
define service{
        use                             generic-service
        host_name                       centos
        service_description             SMTP
        check_command                   check_smtp
        flap_detection_enabled          1
        low_flap_threshold              10.0
        high_flap_threshold             30.0
}
```

There is another option that is available with flapping. This option allows an administrator to control which states indicated flapping. The states available are o(OK), w(WARNING), c(CRITICAL) and u(UNKNOWN). States that are not listed are not taken into account to determine flapping.

```
flap_detection_options                  o,w,c,u
```



Host State Information

Host Status:	UP (for 0d 0h 1m 35s)
Status Information:	Host Status
Performance Data:	
Current Attempt:	1/10 (HARD state)
Last Check Time:	12-30-2010 00:47:21
Check Type:	PASSIVE
Check Latency / Duration:	N/A / 0.000 seconds
Next Scheduled Active Check:	12-30-2010 00:51:16
Last State Change:	12-30-2010 00:47:26
Last Notification:	N/A (notification 0)
Is This Host Flapping?	YES (24.21% state change)
In Scheduled Downtime?	NO
Last Update:	12-30-2010 00:48:56 (0d 0h 0m 5s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Host Commands

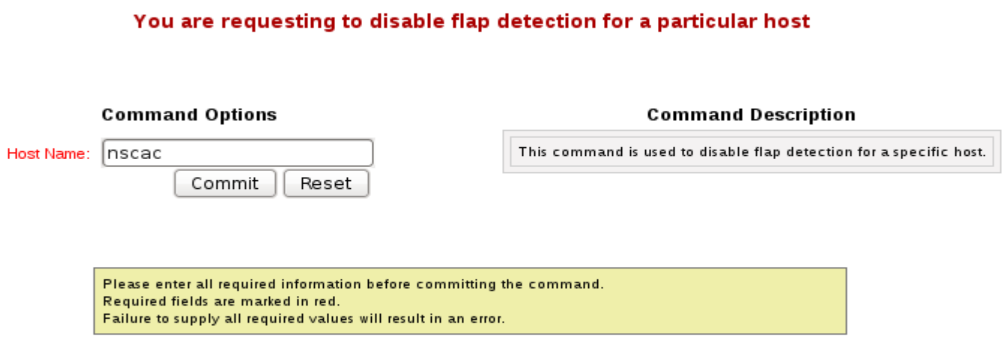
- Locate host on map
- Disable active checks of this host
- Re-schedule the next check of this host
- Submit passive check result for this host
- Stop accepting passive checks for this host
- Stop obsessing over this host
- Disable notifications for this host
- Send custom host notification
- Schedule downtime for this host
- Schedule downtime for all services on this host
- Disable notifications for all services on this host
- Enable notifications for all services on this host
- Schedule a check of all services on this host
- Disable checks of all services on this host
- Enable checks of all services on this host
- Disable event handler for this host
- Disable flap detection for this host

Host Comments

Add a new comment Delete all comments

Entry Time	Author	Comment	Comment ID	Persistent	Type	Expires	Actions
12-30-2010 00:47:26	(Nagios Process)	Notifications for this host are being suppressed because it was detected as having been flapping between different states (24.2% change > 20.0% threshold). When the host state stabilizes and the flapping stops, notifications will be re-enabled.	3	No	Flap Detection	N/A	🗑

As you can see in this illustration you can also "Disable flap detection for this host" under the "Host Commands". This provides the option to just perform the task as it happens. Here is the verification before you commit the change.



You are requesting to disable flap detection for a particular host

Command Options

Host Name: nscac

Commit Reset

Command Description

This command is used to disable flap detection for a specific host.

Please enter all required information before committing the command.
Required fields are marked in red.
Failure to supply all required values will result in an error.

Service and Host Check Options

Public Service Checks

There are a number of protocols that exist which allow the Nagios server to test them externally. For example the common port 80 is available to be check on any web server.

FTP	- port 21
SSH	- port 22
Web	- port 80
SMTP	- port 25
Secure Web	- port 443

These public services allow Nagios to not only check to see if the port is open but to verify the correct application is running on the specific port. This can be done because each of these public services run specific protocols which provide the information needed to monitor them correctly and to differentiate them from other services on the same port.

Checks Using SSH
Nagios can connect to a client server using SSH and then execute a local plugin to check internal functions of the server like CPU load, memory, processes, etc. The advantage of using SSH is that checks are secure in the connection and the transfer of information. The disadvantage for SSH checks are that they take more resources than other check types.

Nagios Remote Plugin Executor
NRPE, Nagios Remote Plugin Executor, executes plugins internally on the client and then returns that information to the Nagios server. The Nagios server connects on port 5666 in order to execute the internal check. NRPE is protected by the xinetd daemon on the client so that an administrator can restrict the connections to the NRPE plugins.

Monitoring with SNMP
SNMP, Simple Network Management Protocol, is used extensively in network devices, server hardware and software. SNMP is able to monitor just about anything that connects to a network, that is the advantage. The disadvantage is that it is not easy to work with. The complexity of SNMP is made even worse by the fact that vendors write proprietory tools to monitor SNMP that are not easily accessed using Nagios. SNMP can be monitored directly using Nagios plugins or the device itself can monitory SNMP and send information to SNMP traps which can be located on the Nagios server. The difficulties are further aggrevated when using traps as the SNMP trap information must be translated into data that Nagios can understand.

Nagios Service Check Acceptor
NSCA, Nagios Service Check Acceptor, employs a daemon on the Nagios server which waits for information generated by passive checks which execute independently on the client being monitored by Nagios. The advantage of NCSA is that services are monitored locally independent of the Nagios server and then sent to the Nagios server so this is a good option when a firewall between the Nagios server and the client prevent other types of communication. The disadvantage is that passive checks use plugins but often require scripts to execute on the client.

Communication can be encrypted between the client and the Nagios server and a password will be required to complete communication.

Another use for NSCA is distributed monitoring. Distributed monitoring allows a wide geographical base of network devices to be monitored by multiple Nagios servers which use NSCA to send service checks and host checks to a central Nagios server.

Basic Nagios Configuration

The manual provides step-by-step instructions for setting up a CentOS /RHEL based server. Note that the file locations and the names of files may be different depending upon how you install Nagios. **This manual is based on compiling Nagios on a CentOS server.**

Installing From Source

Installation from source is a process where the source code that was developed by the programmer is converted into a binary format that the server can run. Compiling Nagios is not as difficult as it may sound. It may require a few extra steps in setting up Nagios but there are several advantages over using a RPM repository or a DEB repository. The biggest advantage of installing from source is that the installation process can be repeated on almost any Linux distribution. This aspect is even more important when you consider that whether you install from a RPM repository (CentOS) or from a DEB repository (Ubuntu) the file names and locations for files are different in each case. The implications for documentation are that you must translate any documentation to the installation method that was chosen.

Another significant advantage of compiling from source is that you have more options so the configuration may be altered to meet specific requirements. Of course, any changes to the defaults mean that the documentations and other dependencies must be evaluated per the changes from the default.

The installation of Nagios must be performed as root. In order for all of the following commands to work, become root by entering the following command:

```
	su -
```

Install from source by first moving into the directory that you want to make the installation from. This should be a directory that you can clean up when you have compiled Nagios. The sources files from this directory can be removed once the installation is complete. For this reason many users create download directories, use the /opt directory or the /tmp directory. In this example the /usr/local/src directory is used.

```
	cd /usr/local/src
```

The source code that is downloaded is in the form of a tarball and compressed so it is in the form of a tar.gz file. The wget command is used to pull the source code down from the web site.

```
	wget http://sourceforge.net/projects/nagios/files/nagios-3.x/nagios-
3.2.3/nagios-3.2.3.tar.gz/download

	wget_
http://sourceforge.net/projects/nagiosplug/files/nagiosplug/1.4.15/nagios-plugins-
1.4.15.tar.gz/download
```

Prerequisites to compile.
When you compile software it will require a compiler like GCC. In order to compile an application it requires the source code. This source code is what the programmer has developed in an editor. The compiler takes the source code and converts it into binary code that the server can use. Or to put it another way, the source code is taken and built into object code which can then be executed from the computer hardware. It is typical that the source code will have dependencies as well. Dependencies are applications that are required to be installed before the source code will

work properly. Several of the files installed with yum in this example are dependencies that must be available. Note that depending on the Linux distribution these dependency applications may be called by different names.

```
      yum install -y httpd php gcc glibc glibc-common gd gd-devel
```

Add the required users and groups.

```
      useradd nagios
      groupadd nagcmd

      usermod -a -G nagcmd nagios
```

The tarballs are compressed so in order to compile these must be expanded into the directories that contain the source code.

```
      tar zxvf nagios-3.2.3.tar.gz
      tar zxvf nagios-plugins-1.4.15.tar.gz
```

Move into the directory created when the Nagios source was uncompressed and run the configure script using the group that was created earlier.

```
      cd nagios-3.2.3
      ./configure -with-command-group=nagcmd
```

The make command will compile the Nagios source code.

```
      make all
```

Now make will install the binaries, the init script, the config files, set the permissions on the external command directory and verify the web configuration files are installed. The semicolons allow you to run all the commands from one line.

```
      make install; make install-init; make install-config; make install-
commandmode; make install-webconf
```

Edit the contacts.cfg and and add the email for the primary nagios administrator, nagiosadmin.

```
      vi /usr/local/nagios/etc/objects/contacts.cfg
```

Create a password for the nagiosadmin which will be needed in order to login to the web interface.

```
      htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

Nagios Plugins
Move into the directory created when the Nagios plugins source was uncompressed and run the configure script using the group that was created earlier. **Note: If you want to use check_snmp be sure to install net-snmp before you compile the plugins.**

Either compile net-snmp (see the SNMP chapter) or install it with yum.

```
      yum install -y net-snmp
      cd /usr/local/src
      cd nagios-plugins-1.4.15
      ./configure -with-nagios-user=nagios -with-nagios-group=nagios
```

Now make will install the binaries.

```
      make
      make install
```

Initial Set Up

The first step is to add a contact email for the nagiosadmin. The user nagiosadmin by default is the only user able to access the whole web interface. This can be changed but the default user is nagiosadmin.

Change the Contact Information

Edit `/usr/local/nagios/etc/objects/contacts.cfg (RPM repository` /etc/nagios/objects/contacts.cfg).

Place your email in the email location.

```
define contact{
        contact_name                    nagiosadmin             ; Short name of user
        use                             generic-contact         ; Inherit default values
        alias                           Nagios Admin            ; Full name of user
        email                           your_email  ; <<***** CHANGE THIS TO YOUR EMAIL
        }
```

Pre-Flight Check

The pre-flight check provides a way to verify all of the configuration files which exist in the /usr/local/nagios/etc/objects directory. This command reads and verifies the initial set up.

```
      nagios -v /usr/local/nagios/etc/nagios.cfg
```

```
Nagios 3.0.6
Copyright (c) 1999-2008 Ethan Galstad (http://www.nagios.org)
Last Modified: 12-01-2008
License: GPL

Reading configuration data...

Running pre-flight check on configuration data...

Checking services...
```

```
      Checked 8 services.
Checking hosts...
      Checked 1 hosts.
Checking host groups...
      Checked 1 host groups.
Checking service groups...
      Checked 0 service groups.
Checking contacts...
      Checked 1 contacts.
Checking contact groups...
      Checked 1 contact groups.
Checking service escalations...
      Checked 0 service escalations.
Checking service dependencies...
      Checked 0 service dependencies.
Checking host escalations...
      Checked 0 host escalations.
Checking host dependencies...
      Checked 0 host dependencies.
Checking commands...
      Checked 24 commands.
Checking time periods...
      Checked 5 time periods.
Checking for circular paths between hosts...
Checking for circular host and service dependencies...
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

By default it should run and you should be able to login to the web interface after you create the nagiosadmin user.

```
      htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
```

Now login to the web interface with http://ip_address/nagios

Eliminating an HTTP Error
When you set up the Nagios server and either review your log files in /var/log/nagios/nagios.log or review the web interface you may initially see an error related to the web server. The error is related to the fact that you do not have a an index.html file that exists. **Note: If you do not see the error it is because you have the necessary files so you can skip this step.** Here is what it will look like in the log.

```
WARNING: HTTP/1.1 403 Forbidden-5240 bytes in 0.001 second response time
Sep 26 10:00:18 nagios nagios: SERVICE ALERT: localhost;HTTP;WARNING;HARD;4;HTTP
```

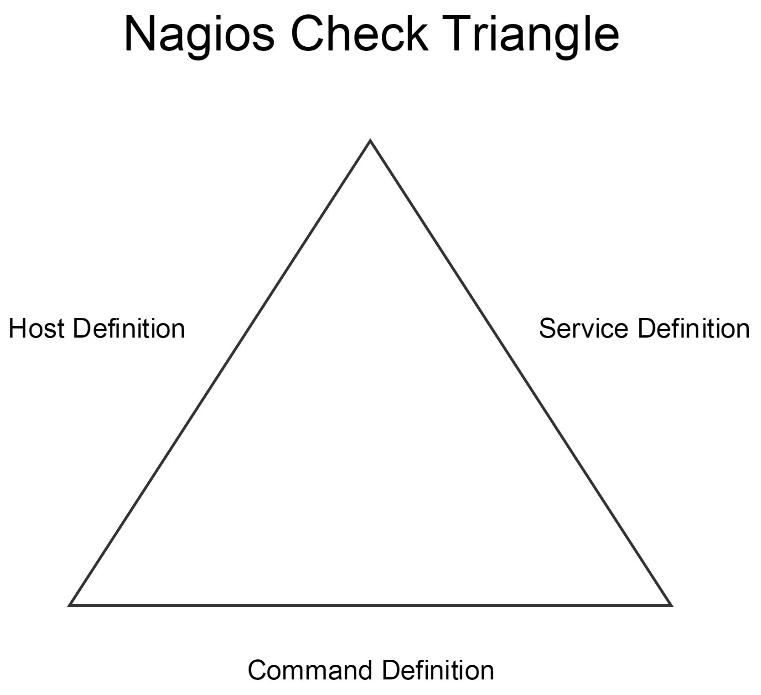
Here is what it will look like in the web interface.

You can easily eliminate the error by creating an index.html file. Create a simple HTML.

```
      vi /var/www/html/index.html

<HTML>
<BODY>
Nagios Server
</BODY>
</HTML>

      chmod 755 /var/www/html/index.html
      chown apache:apache /var/www/html/index.html
```

Nagios Check Triangle

One of the major concepts of creating checks is to remember that all plugins with Nagios will require three elements to be configured. There must be a host definition, a service definition and a command definition. Think of it as a triangle each time you want to use a plugin.



These three definitions are all located in three separate files, hosts.cfg, services.cfg and commands.cfg. You may need to create hosts.cfg and services.cfg as they are not created by default. These files must be located in:

/usr/local/nagios/etc/objects

Host Defintion
Nagios needs to know an IP Address of the host you want to check. This is configured in the hosts.cfg file. The hosts.cfg file does not exist initially so you will need to create it. In this example the host_name is "win2008" and it is tied to the address "192.168.3.114". This is the information Nagios must have to know where to point a request and how to record information for a specific host.

Create the file, hosts.cfg, in /usr/local/nagios/etc/objects

```
define host{
       use                             windows-server
       host_name                       win2008
       alias                           Windows Server
       address                         192.168.3.114
}
```

Service Definition
The second part of the triangle is the service definition. Nagios needs to know what service you want to check, so that service or plugin must be defined. In this example the host "win2008", which Nagios knows now is tied to the IP Address 192.168.3.114, is being checked with the ping plugin. So you can see the host_name determines which host the plugin acts upon and then the service_description is really the text that shows up in the web interface. The check_command, defines the parameters of the plugin. Here you can see that "check_ping" is the plugin and it is followed by two different sections of options divided by "!". The first section, "60.0,5%", provides a warning level if packets are take longer than 60 milliseconds or if there is greater than a 5% loss of packets when the ping command is performed. The second section is the critical level where a CRITICAL state will b e created if packets take longer than 100 milliseconds or if there is more than 10% packet loss.

Create the file, services.cfg, in the /usr/local/nagios/etc/objects directory.

```
define service{
        use                             generic-service
        host_name                       win2008
        service_description             Ping
        check_command                   check_ping!60.0,5%!100.0,10%
}
```

Command Definition
The command definitions are located in the commands.cfg file which is created by default in the objects directory. Many commands are already defined so you do not have to do anything. The check_ping command is one example that has been defined. The command_name, "check_ping", is what is part of the service definition. The command_line specifically defines where the plugin is located with the "$USER1$ macro. This is equal to saying that the plugin check_ping is located in /usr/local/nagios/libexec (if you compiled). The other 4 options include the host, using the $HOSTADDRESS$ macro, a warning level (-w) using the $ARG1$ macro, the critical level (-c) using the $ARG2$ macro and the number of pings to use by default (-p 5).

Edit this file, /usr/local/nagios/etc/objects/commands.cfg as it will be created by default.

```
# 'check_ping' command definition
define command{
      command_name    check_ping
      command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5
        }
```

In each of the elements of the Nagios triangle you can see the importance of the term “definition” as each element must be clearly defined and each element is dependent upon the other definitions.

Important:
You will have created two configuration files which did not exist previously. You must create a path to those files in the main nagios configuration file found at:

/usr/local/nagios/etc/nagios.cfg

cfg_file=/usr/local/nagios/etc/objects/hosts.cfg
cfg_file=/usr/local/nagios/etc/objects/services.cfg

You will see other paths have been also created. Any time you create a new configuration file this should be entered in the nagios.cfg file.

Run the pre-flight check to verify all of the configuration files which exist in the /usr/local/nagios/etc/objects directory. This command reads and verifies the initial set up.

```
      nagios -v /usr/local/nagios/etc/nagios.cfg
```

Important Paths to Note when you compile Nagios on a CentOS server.

NAGIOS	**Program Location**	**Configuration File**	**Plugins**
Compile	/usr/local/nagios/bin/nagios	/usr/local/nagios/etc/nagios.cfg	/usr/local/nagios/libexec
NRPE	**Program Location**	**Configuration File**	
Compile	/usr/local/nagios/bin/nrpe	/usr/local/nagios/etc/nrpe.cfg	/usr/local/nagios/libexec
NSCA	**Program Location**	**Configuration File**	
compile	/usr/local/nagios/bin/nsca	/usr/local/nagios/etc/nsca.cfg	
WEB	**Web Pages**	**cgi Configuration**	**cgi Files**
Compile	/usr/local/nagios/share	/usr/local/nagios/etc/cgi.cfg	
Web Server	**Program Location**	**Web Server Configuration**	**Nagios Web Config**
CentOS	/usr/sbin/httpd	/etc/httpd/conf/httpd.conf	/etc/httpd/conf.d/nagios.cfg
	htpasswd Database		
Compile	/usr/local/nagios/etc		

Administration Tasks

Authentication

Authentication is the process that allows users to access the web interface. Authentication is controlled by the use of a database using the htpasswd command. The database, called htpasswd.users, is located in the /usr/local/nagios/etc directory. The name and location of the database is determined by the configuration options found in /etc/httpd/conf.d/nagios.conf. In this example, from a CentOS install, you can see that several directories require authentication from this database.

```
ScriptAlias /nagios/cgi-bin "/usr/local/nagios/sbin"

<Directory "/usr/local/nagios/sbin">
#  SSLRequireSSL
   Options ExecCGI
   AllowOverride
   None Order
   allow,deny Allow
   from all
#  Order deny,allow
#  Deny from all
#  Allow from 127.0.0.1
   AuthName "Nagios
   Access" AuthType Basic
   AuthUserFile /usr/local/nagios/etc/htpasswd.users
   Require valid-user
</Directory>

Alias /nagios "/usr/local/nagios/share"

<Directory "/usr/local/nagios/share">
#  SSLRequireSSL
   Options None
   AllowOverride
   None Order
   allow,deny Allow
   from all
#  Order deny,allow
#  Deny from all
#  Allow from 127.0.0.1
   AuthName "Nagios
   Access" AuthType Basic
   AuthUserFile /usr/local/nagios/etc/htpasswd.users
   Require valid-user
</Directory>
```

Access is maintained through the database but the permissions a user has once they authenticate are determined by contacts, contact groups and cgi permissions determined from the cgi.cfg file. An important point to remember when setting up permissions is that the contact is only able to see the host or service that they are responsible for. Make sure contact names match the user created for access to the web interface.

These settings represent the default settings in the /usr/local/nagios/etc/cgi.cfg file for permissions to the web interface. The user "nagiosadmin" is the default nagios user with access and unlimited permissions to the web interface. The defaults demonstrate why it is so important to correctly set up the nagiosadmin user as part of the initial configuration.

```
use_authentication=1
use_ssl_authentication=0
#default_user_name=guest
authorized_for_system_information=nagiosadmin
authorized_for_configuration_information=nagiosadmin
authorized_for_system_commands=nagiosadmin
authorized_for_all_services=nagiosadmin
authorized_for_all_hosts=nagiosadmin
authorized_for_all_service_commands=nagiosadmin
authorized_for_all_host_commands=nagiosadmin
#authorized_for_read_only=user1,user2
```

Scenario: Turn Off All Authentication
Turning off all authentication is **not recommended under any circumstances**. It is only demonstrated here in order to aid in the understanding of how Nagios authentication works. These changes allow anyone to make changes to the Nagios interface, hosts and services.

There are two steps required to turn off all security. Edit the cgi.cfg file located in /usr/local/nagios/etc and change the "use_authentication" to a "0".

```
use_authentication=0
```

The second step required is to access the /etc/httpd/conf.d/nagios.conf file and comment out the lines that require authentication for the Nagios directories.

```
ScriptAlias /nagios/cgi-bin "/usr/local/nagios/sbin"

<Directory "/usr/local/nagios/sbin">
#  SSLRequireSSL
   Options ExecCGI
   AllowOverride
   None Order
   allow,deny Allow
   from all
```

```
#  Order deny,allow
#  Deny from all
#  Allow from 127.0.0.1
#   AuthName "Nagios Access"
#   AuthType Basic
#   AuthUserFile /usr/local/nagios/etc/htpasswd.users
#   Require valid-user
</Directory>

Alias /nagios "/usr/local/nagios/share"

<Directory "/usr/local/nagios/share">
#  SSLRequireSSL
   Options None
   AllowOverride
   None Order
   allow,deny Allow
   from all
#  Order deny,allow
#  Deny from all
#  Allow from 127.0.0.1
#   AuthName "Nagios Access"
#   AuthType Basic
#   AuthUserFile /usr/local/nagios/etc/htpasswd.users
#   Require valid-user
</Directory>
```

Restart Nagios and the web server.

Scenario: Create a View Only Account
This scenario will create a user that can view all hosts and services but not be allowed to make any changes to those hosts or services. This is typically the settings you may choose for management to review the status of hosts and services.

Create the user in the htpasswd.users database.

```
htpasswd htpasswd.users management New
password:
Re-type new password:
```

Make modifications to the /usr/local/nagios/etc/cgi.cfg file by adding the user separated by a comma, without spaces. The user has global access, which means they are not required to be listed as contacts for hosts and services. The user is also added to the read only list.

```
authorized_for_all_services=nagiosadmin,management
authorized_for_all_hosts=nagiosadmin,management
authorized_for_read_only=management
```

Restart Nagios and the web server.

Scenario: Create System Administrator with No Contact Information
In this scenario the settings will allow a user to have full access to all settings on all hosts and services just like the

nagiosadmin user. However, this user is not associated with any contact information so will not be notified at any time. This account is strictly administration only.

```
htpasswd htpasswd.users john New
password:
Re-type new password:
```

Edit the cgi.cfg file and add john to each of the lists indicated below.

authorized_for_system_information=nagiosadmin,john
authorized_for_configuration_information=nagiosadm,john
authorized_for_system_commands=nagiosadmin,john
authorized_for_all_services=nagiosadmin,john
authorized_for_all_hosts=nagiosadmin,john
authorized_for_all_service_commands=nagiosadmin,john
authorized_for_all_host_commands=nagiosadmin,john

Restart Nagios and the web server.

Scenario: Create an Administrator with Limited Access
This user will only be allowed to access the hosts and services that they are associated with via contact information. This may be the type of settings used when an organization has divided responsibilities for routers, Windows servers and Linux servers for example.

```
htpasswd htpasswd.users sue New
password:
Re-type new password:
```

Create a new contact entry in contacts.cfg and specify the contact_name, alias and email contact information for the user.

```
define contact{
        contact_name                            sue
        use                                     generic-contact
        alias                                   Router Admin
        email                                   sue@example.com
        }
```

Add the user to a group or create a new group in the contacts.cfg file. This example shows a user added to a new contact group called router-admins. By creating a new group it enables an administrator to assign that group to a series of devices, like routers.

```
define contactgroup{
        contactgroup_name       router-admins
        alias                   Router Administrators
        members                 sue
        }
```

At this point you will need to edit the hosts and services and add the “contact_groups router-admins” which will override the default settings in the template. This will enable only those users in this contact group access to these hosts and services unless they have global access from the cgi.cfg file.

```
define host{
        use                     generic-switch
        host_name               cisco
        alias                   cisco router
        address                 192.168.5.220
        contact_groups          router-admins
}
define service{
        use                     generic-service
        host_name               cisco
        service_description     PING
        check_command           check_ping!200.0,20%!600.0,60%
        normal_check_interval   5
        retry_check_interval    1
        contact_groups          router-admins
}
```

Restart Nagios and the web server.

Scheduled Downtime

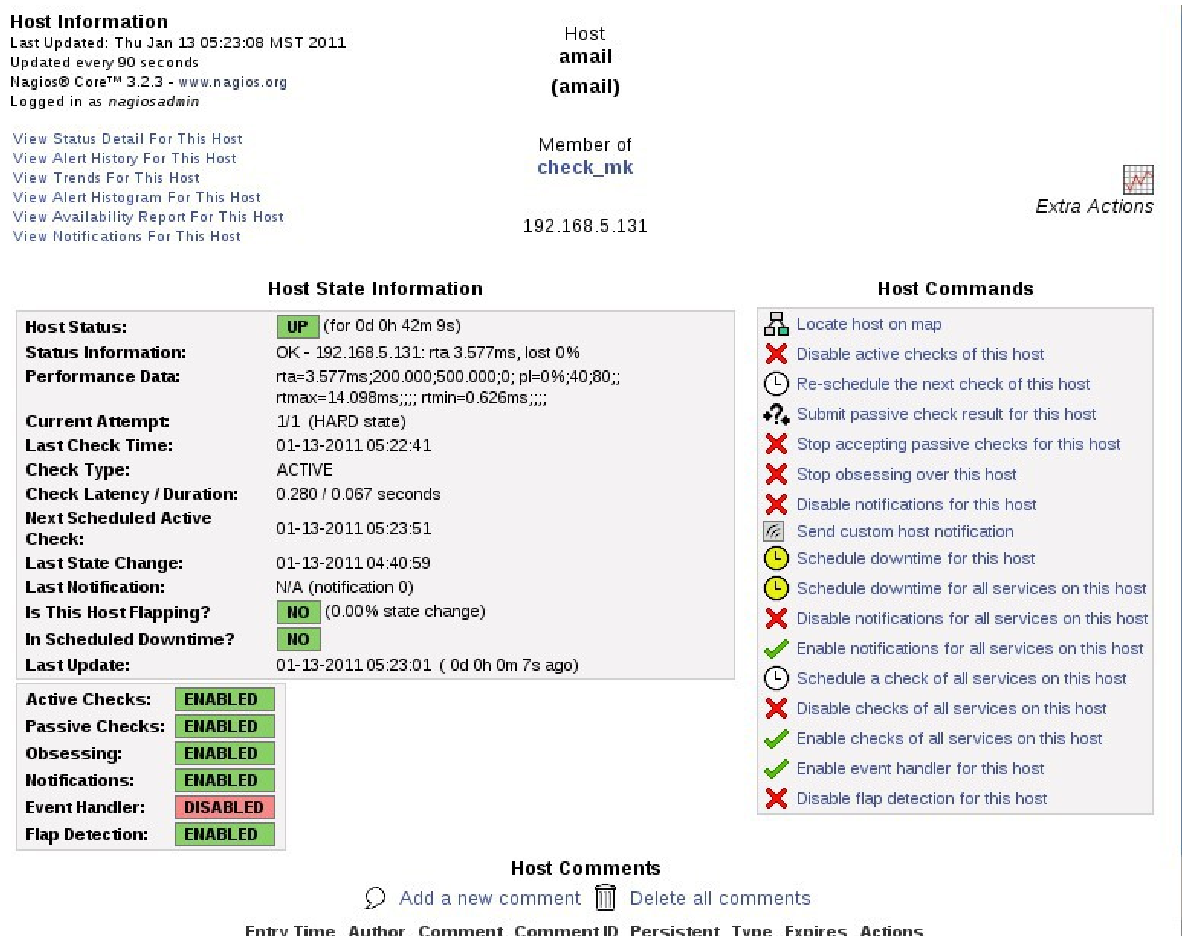
If you are going to work on a server or device and need to schedule downtime so Nagios does not notify administrators that can be performed at the web interface. When you select the host or service that will be down you have an option to schedule downtime. When downtime is scheduled Nagios will place a comment in the web interface in order to communicate the fact to all administrators who access the web interface.

There are two types of downtime. **Fixed downtime** allows for and exact start and end time when the host or service will be unavailable. **Flexible downtime** allows for a start time but an open ended startup time as the exact time cannot be determined based on the nature of the situation.

Triggered downtime is when the downtime of a parent will trigger downtime for all of it's children. In other words, the downtime for a switch, will impact all of the devices connected to it.

Scheduling Downtime for a Host

In order to schedule downtime for a host, select host details from the web interface. On the right hand side you will notice the “yellow clocks” permit scheduling for host or services. Select the host option.



Host Information

Last Updated: Thu Jan 13 05:23:08 MST 2011
Updated every 90 seconds
Nagios® Core™ 3.2.3 - www.nagios.org
Logged in as *nagiosadmin*

View Status Detail For This Host
View Alert History For This Host
View Trends For This Host
View Alert Histogram For This Host
View Availability Report For This Host
View Notifications For This Host

Host
amail
(amail)

Member of
check_mk

192.168.5.131

Extra Actions

Host State Information

Host Status:	UP (for 0d 0h 42m 9s)
Status Information:	OK - 192.168.5.131: rta 3.577ms, lost 0%
Performance Data:	rta=3.577ms;200.000;500.000;0; pl=0%;40;80;; rtmax=14.098ms;;;; rtmin=0.626ms;;;;
Current Attempt:	1/1 (HARD state)
Last Check Time:	01-13-2011 05:22:41
Check Type:	ACTIVE
Check Latency / Duration:	0.280 / 0.067 seconds
Next Scheduled Active Check:	01-13-2011 05:23:51
Last State Change:	01-13-2011 04:40:59
Last Notification:	N/A (notification 0)
Is This Host Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	01-13-2011 05:23:01 (0d 0h 0m 7s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	DISABLED
Flap Detection:	ENABLED

Host Commands

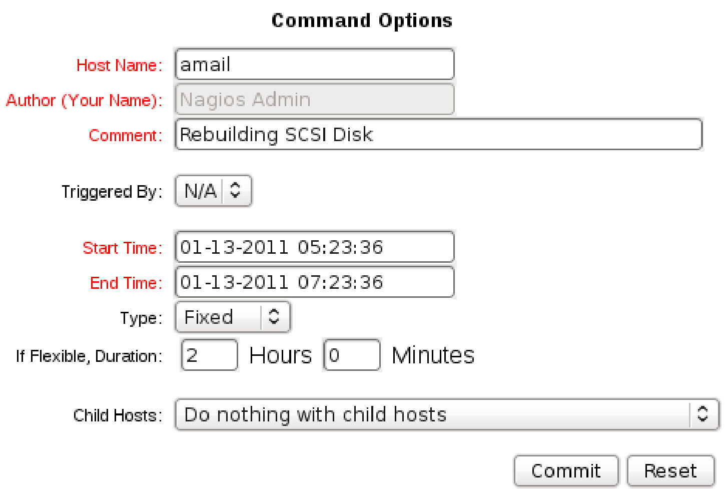
- Locate host on map
- Disable active checks of this host
- Re-schedule the next check of this host
- Submit passive check result for this host
- Stop accepting passive checks for this host
- Stop obsessing over this host
- Disable notifications for this host
- Send custom host notification
- Schedule downtime for this host
- Schedule downtime for all services on this host
- Disable notifications for all services on this host
- Enable notifications for all services on this host
- Schedule a check of all services on this host
- Disable checks of all services on this host
- Enable checks of all services on this host
- Enable event handler for this host
- Disable flap detection for this host

Host Comments

Add a new comment   Delete all comments

Entry Time	Author	Comment	Comment ID	Persistent	Type	Expires	Actions

Once you have selected the host, "Command Options" appears and provides a place to explain why the downtime to other administrators in the comment area, which is a good idea in most situations. If you select a "Fixed" time you will enter the start and end of the downtime. If this machine that provided network connection with other devices you may want to notify downstream devices with a "triggered by" option that is created by this device going down. Or you may choose to do nothing.



Command Options

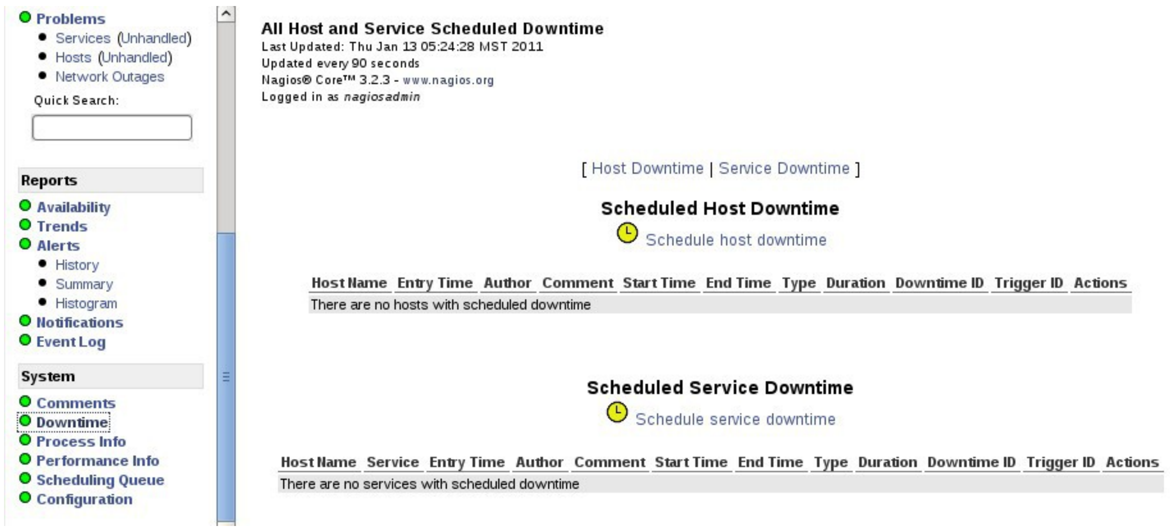
Host Name: amail
Author (Your Name): Nagios Admin
Comment: Rebuilding SCSI Disk

Triggered By: N/A

Start Time: 01-13-2011 05:23:36
End Time: 01-13-2011 07:23:36
Type: Fixed
If Flexible, Duration: 2 Hours 0 Minutes

Child Hosts: Do nothing with child hosts

On the Nagios interface on the left menu, if you select "Downtime" you will see a list of all scheduled downtimes for hosts and services. Remember it may take a few minutes to allow the devices to show up.



All Host and Service Scheduled Downtime
Last Updated: Thu Jan 13 05:24:28 MST 2011
Updated every 90 seconds
Nagios® Core™ 3.2.3 - www.nagios.org
Logged in as *nagiosadmin*

[Host Downtime | Service Downtime]

Scheduled Host Downtime

Schedule host downtime

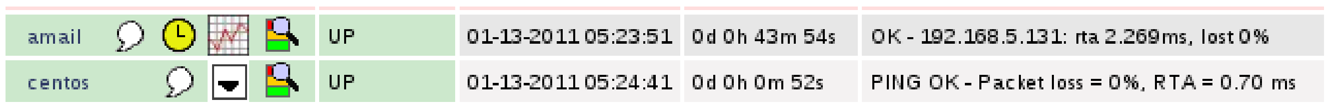
Host Name	Entry Time	Author	Comment	Start Time	End Time	Type	Duration	Downtime ID	Trigger ID	Actions
There are no hosts with scheduled downtime										

Scheduled Service Downtime

Schedule service downtime

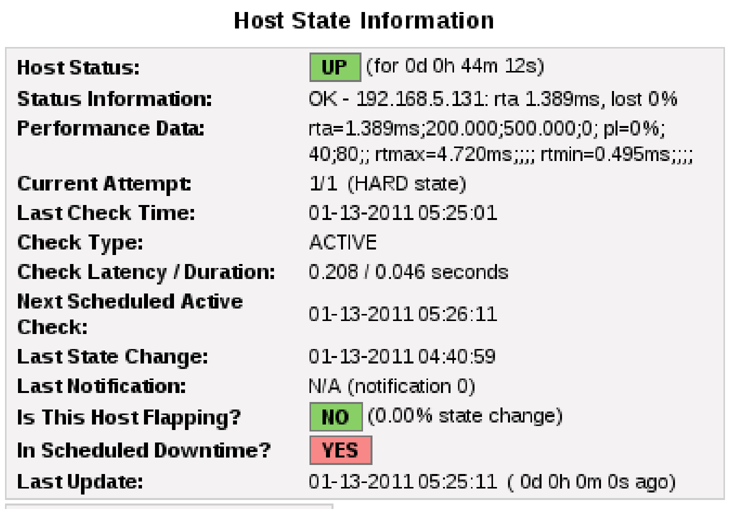
Host Name	Service	Entry Time	Author	Comment	Start Time	End Time	Type	Duration	Downtime ID	Trigger ID	Actions
There are no services with scheduled downtime											

Here is how the host looks with downtime (this is the exfoliation frontend), note the "yellow clock" which is an indicator of scheduled downtime.



amail	🕒	UP	01-13-2011 05:23:51	0d 0h 43m 54s	OK - 192.168.5.131: rta 2.269ms, lost 0%
centos		UP	01-13-2011 05:24:41	0d 0h 0m 52s	PING OK - Packet loss = 0%, RTA = 0.70 ms

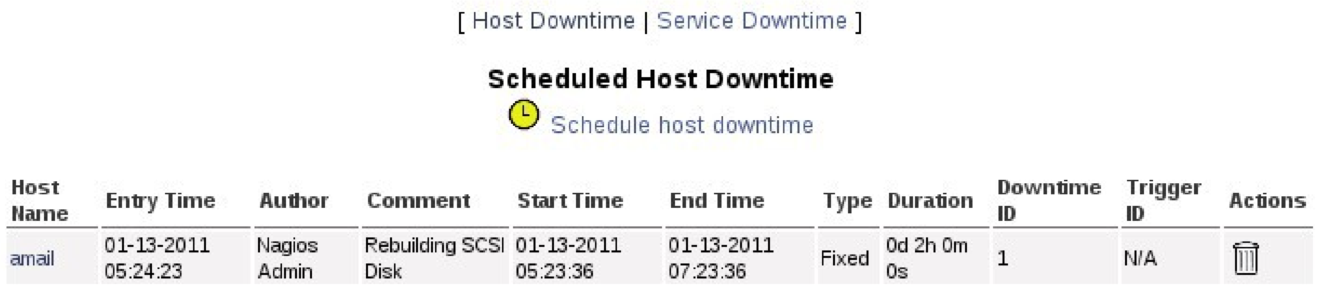
If you select the clock you will see the details on the host list it as being in a scheduled downtime.



Host State Information

Host Status:	UP (for 0d 0h 44m 12s)
Status Information:	OK - 192.168.5.131: rta 1.389ms, lost 0%
Performance Data:	rta=1.389ms;200.000;500.000;0; pl=0%; 40;80;; rtmax=4.720ms;;;; rtmin=0.495ms;;;;
Current Attempt:	1/1 (HARD state)
Last Check Time:	01-13-2011 05:25:01
Check Type:	ACTIVE
Check Latency / Duration:	0.208 / 0.046 seconds
Next Scheduled Active Check:	01-13-2011 05:26:11
Last State Change:	01-13-2011 04:40:59
Last Notification:	N/A (notification 0)
Is This Host Flapping?	NO (0.00% state change)
In Scheduled Downtime?	YES
Last Update:	01-13-2011 05:25:11 (0d 0h 0m 0s ago)

At this point it will be listed in the “Downtime” menu. Note you can cancel by deleting the downtime.



[Host Downtime | Service Downtime]

Scheduled Host Downtime

Schedule host downtime

Host Name	Entry Time	Author	Comment	Start Time	End Time	Type	Duration	Downtime ID	Trigger ID	Actions
amail	01-13-2011 05:24:23	Nagios Admin	Rebuilding SCSI Disk	01-13-2011 05:23:36	01-13-2011 07:23:36	Fixed	0d 2h 0m 0s	1	N/A	🗑

Notifications and Downtime

Notifications for downtime should stop in the downtime period. If the notifications do not stop verify that you do not have the “d” option set for your contacts. The “d” option will send notifications on downtime.

Host Groups

Often you will want to create a group of devices that have similar monitoring needs. The hostgroup allows you to then create service checks that monitor all of the devices in the hostgroup. Specifically what this means is that the services defined for the group will be available for all hosts in the group without making individual configurations. Nagios will also list the hosts together in the web interface if they are in the same hostgroup.

Define Each Host

In order to set up a hostgroup, each server must be defined as a host. In this example, 3 Ubuntu servers are defined.

```
define host{
       use                             linux-server
       host_name                       ub
       alias                           Ubuntu Server
       address                         192.168.5.180
}
define host{
       use                             linux-server
       host_name                       ub1
       alias                           Ubuntu Server
       address                         192.168.5.181
}
define host{
       use                             linux-server
       host_name                       ub3
       alias                           Ubuntu Server
       address                         192.168.5.183
}
```

Define Host Groups

Create hostgroups.cfg in the objects directory and create an entry in nagios.cfg to the location of hostgroups.cfg.

```
cfg_file=/usr/local/nagios/etc/objects/hostgroup.cfg
```

Define the hostgroup, in this example the hostgroup ubuntu_servers is defined with the three members that were defined in hosts.cfg file.

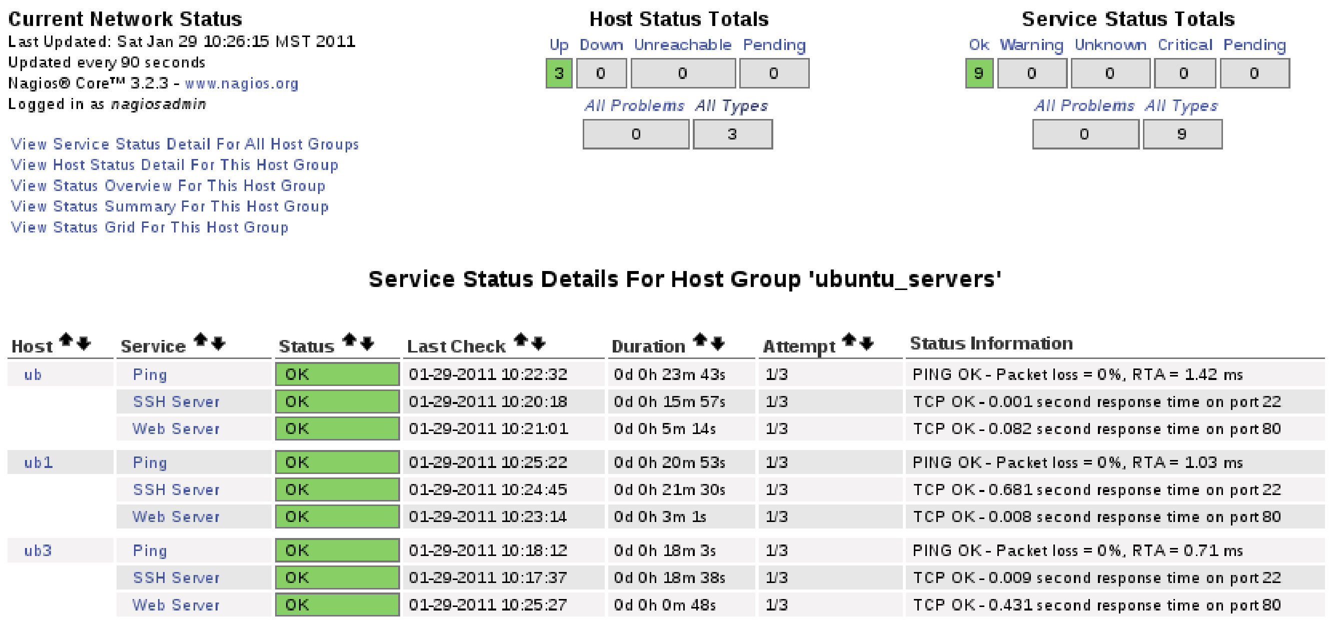
```
define hostgroup {
        hostgroup_name          ubuntu_servers
        alias                   Ubuntu Servers
        members                 ub,ub1,ub3
}
```

Define Services for the Group

The advantage of the hostgroup is that you can create one service definition and add that to the whole group of servers. This is exactly the same as a regular service definition except you use hostgroup_name instead of host.

```
define service{
        use                             generic-service
        hostgroup_name                  ubuntu_servers
        service_description             Ping
        check_command                   check_ping!60.0,5%!100.0,10%
}
define service{
        use                             generic-service
        hostgroup_name                  ubuntu_servers
        service_description             SSH Server
        check_command                   check_tcp!22
}
define service{
        use                             generic-service
        hostgroup_name                  ubuntu_servers
        service_description             Web Server
        check_command                   check_tcp!80
}
```

Now if you go to the web interface and select "Hostgroups" you will have a group of servers that are all related with the same service checks.



Current Network Status
Last Updated: Sat Jan 29 10:26:15 MST 2011
Updated every 90 seconds
Nagios® Core™ 3.2.3 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Host Status Detail For This Host Group
View Status Overview For This Host Group
View Status Summary For This Host Group
View Status Grid For This Host Group

Host Status Totals

Up	Down	Unreachable	Pending
3	0	0	0

All Problems	All Types
0	3

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
9	0	0	0	0

All Problems	All Types
0	9

Service Status Details For Host Group 'ubuntu_servers'

Host	Service	Status	Last Check	Duration	Attempt	Status Information
ub	Ping	OK	01-29-2011 10:22:32	0d 0h 23m 43s	1/3	PING OK - Packet loss = 0%, RTA = 1.42 ms
	SSH Server	OK	01-29-2011 10:20:18	0d 0h 15m 57s	1/3	TCP OK - 0.001 second response time on port 22
	Web Server	OK	01-29-2011 10:21:01	0d 0h 5m 14s	1/3	TCP OK - 0.082 second response time on port 80
ub1	Ping	OK	01-29-2011 10:25:22	0d 0h 20m 53s	1/3	PING OK - Packet loss = 0%, RTA = 1.03 ms
	SSH Server	OK	01-29-2011 10:24:45	0d 0h 21m 30s	1/3	TCP OK - 0.681 second response time on port 22
	Web Server	OK	01-29-2011 10:23:14	0d 0h 3m 1s	1/3	TCP OK - 0.008 second response time on port 80
ub3	Ping	OK	01-29-2011 10:18:12	0d 0h 18m 3s	1/3	PING OK - Packet loss = 0%, RTA = 0.71 ms
	SSH Server	OK	01-29-2011 10:17:37	0d 0h 18m 38s	1/3	TCP OK - 0.009 second response time on port 22
	Web Server	OK	01-29-2011 10:25:27	0d 0h 0m 48s	1/3	TCP OK - 0.431 second response time on port 80

If you want to add individual service checks for one of the servers in the hostgroup that would be done as a regular service definition using the host.

Service Groups

Nagios combines devices that are checking the same services into group in order to make the set up faster and more efficient. This allows an administrator to group machines based on services. Each of these services must be configured as service checks for each host. Once that is complete the services may be grouped in the servicegroups.cfg. The other major advantage is that the administrator may manage all those in the service group with servicegroup commands in the web interface.

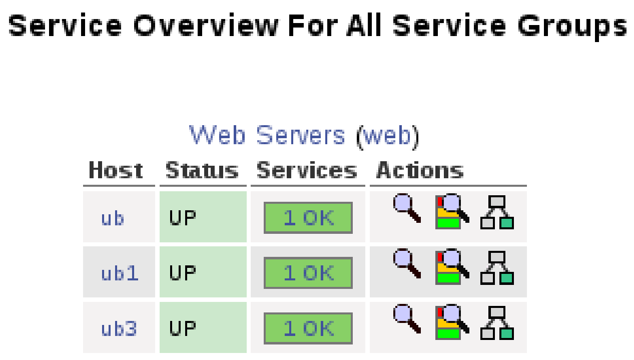
You will need to create a file called servicegroups.cfg and put an entry in nagios.cfg to indicate where it is. Note the entries are in pairs (first host, then service) "host,service, host2,service2".

```
define servicegroup{
        servicegroup_name       web
        alias                   Web Servers
        members                 ub, HTTP ,ub1, HTTP ,ub3, HTTP
}
```

Define each host with a normal service check.

```
define service{
        use                             generic-service
        host_name                       ub
        service_description             HTTP
        check_command                   check_http
}
define service{
        use                             generic-service
        host_name                       ub1
        service_description             HTTP
        check_command                   check_http
}
define service{
        use                             generic-service
        host_name                       ub3
        service_description             HTTP
        check_command                   check_http
}
```

This now allows the administrator to group these services and view them as a group when "ServiceGroups" is selected in the web interface.



Service Overview For All Service Groups

Web Servers (web)

Host	Status	Services	Actions
ub	UP	1 OK	
ub1	UP	1 OK	
ub3	UP	1 OK	

Monitoring Public Ports

Each of the plugins that monitors a specific service.

Each plugin will evaluate the situation and return a status value to Nagios. There are four status values that Nagios interprets.

0	OK	the stats is as expected
1	WARNING	a warning limit has been reached
2	CRITICAL	a critical limit has been reached
3	UNKNOWN	the status is unknown, misconfiguration

In order for Nagios to provide these four levels of status settings, warning and critical limits must be established. An important aspect of setting these limits is that each network will have different equipment and varying needs so these settings should reflect the individual network. Another setting

Typical Options

-h,	--help	Print detailed help screen
-V,	--version	Print version information
-H	--hostname=ADDRESS	Host name, IP Address, or unix socket (must be an absolute path)
-w	--warning=DOUBLE	Response time to result in warning status (seconds)
-c	--critical=DOUBLE	Response time to result in critical status (seconds)
-t	--timeout=INTEGER	Seconds before connection times out (default: 10)
-v	--verbose	Show details for command-line debugging (Nagios may truncate output)
-4	--use-ipv4	Use IPv4 connection
-6	--use-ipv6	Use Ipv6 connection

check_tcp, check_udp

-p	--port=INTEGER	Port number (default: none)

-E	--escape	Can use \n, \r, \t or \ in send or quit string. Must come before send or quit option Default: nothing added to send, \r\n added to end of quit
-s	--send=STRING	String to send to the server
-e	--expect=STRING	String to expect in server response (may be repeated)
-A	--all	All expect strings need to occur in server response. Default is any
-q	--quit=STRING	String to send server to initiate a clean close of the connection
-r	--refuse=ok|warn|crit	Accept TCP refusals with states ok, warn, crit (default: crit)
-M	--mismatch=ok|warn|crit	Accept expected string mismatches with states ok, warn, crit (default: warn)
-j	--jail	Hide output from TCP socket
-m	--maxbytes=INTEGER	Close connection once more than this number of bytes are received
-d	--delay=INTEGER	Seconds to wait between sending string and polling for response
-D	--certificate=INTEGER	Minimum number of days a certificate has to be valid.
-S	--ssl	Use SSL for the connection.

check_ping

Ping is a standard method of checking to see if a network device is up.

Uniq Options

-p	--packets=INTEGER	number of ICMP ECHO packets to send (Default: 5)

Here is a service definition with a warning level of 60 milliseconds or 5% packet loss and a critical level of 100 milliseconds or 10% loss. This demonstrates that the settings need to be specific to the device or the network as networks vary. The default is 5 packets in the ping.

```
define service{
        use                                generic-service
        host_name                          centos
        service_description                Ping
        check_command                      check_ping!60.0,5%!100.0,10%
}
```

The command definition can include the settings for warning and critical level if you want to make them standard for all uses of ping on a network.

```
define command{
        command_name    check-host-alive
        command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,80%
-c 5000.0,100% -p 5
        }
```

check_tcp

This plugin will provide the flexibility you need if you need to monitor a port just to verify that the port is available. Here is an example of portmap service checks.

```
define service{
        use                                generic-service
        host_name                          centos
        service_description                Portmap
        check_command                      check_tcp! 111
}
```

```
define
    command{ command_name
                    check_tcp
    command_line    $USER1$/check_tcp -H $HOSTADDRESS$ -p $ARG1$ $ARG2$
    }
```

Note that a common problem with check_tcp is that often the “-p” is added to the service definition. This will create the error “Port must be a positive integer” if the command definition already has the “-p”.

If you have any problems run the command from the command line to experiment.

```
        ./check_tcp -H 192.168.5.1 -p 111
TCP OK - 0.000 second response time on port 111|
time=0.000386s;;;0.000000;10.000000
```

check_http

A common public port that often is check is port 80, http. There are a significant number of options with this plugin to get out of it as much as possible.

-I	--IP-address=ADDRESS	IP address or name (use numeric address if possible to bypass DNS lookup).
-p	--port=INTEGER	Port number (default: 80)
-S	--ssl	Connect via SSL. Port defaults to 443
--sni		Enable SSL/TLS hostname extension support (SNI)
-C	--certificate=INTEGER	Minimum number of days a certificate has to be valid. Port defaults to 443
-e, --expect=STRING		Comma-delimited list of strings, at least one of them is expected in the first (status) line of the server response (default:HTTP/1.) If specified skips all other status line logic (ex: 3xx, 4xx, 5xx processing)
-s	--string=STRING	String to expect in the content
-u	--url=PATH	URL to GET or POST (default: /)
-P	--post=STRING	URL encoded http POST data
-j	--method=STRING	(HEAD, OPTIONS, TRACE, PUT, DELETE) Set HTTP method.
-N	--no-body	Don't wait for document body: stop reading after headers.
-M	--max-age=SECONDS	Warn if document is more than SECONDS old. the number can also be of the form "10m" for minutes, "10h" for hours, or "10d" for days.
-T	--content-type=STRING	specify Content-Type header media type when POSTing
-l	--linespan	Allow regex to span newlines (must precede -r or -R)
-r	--regex, --ereg=STRING	Search page for regex STRING
-R	--eregi=STRING	Search page for case-insensitive regex STRING
--invert-regex		Return CRITICAL if found, OK if not
-a	--authorization=AUTH_PAIR	Username:password on sites with basic authentication
-b	--proxy-authorization=AUTH_PAIR	Username:password on proxy-servers with basic authentication
-A	--useragent=STRING	String to be sent in http header as "User Agent"
-k	--header=STRING	Any other tags to be sent in http header. Use multiple times for additional headers
-L	--link	Wrap output in HTML link (obsoleted by urlize)
-f	--onredirect=<ok|warning|critical|follow|sticky|stickyport>	
-m, --pagesize=INTEGER<:INTEGER>		Minimum page size required (bytes) : Maximum page size required (bytes)

This is the standard way to use the check_http. It checks to verify communication is available on port 80 of a web server. This is in fact, a better check on the server than the check_ping which can only determine if the server is up. This simple check provides some peace of mind and a place to start.

```
define service{
        use                             generic-service
        host_name                       centos
        service_description             HTTP
        check_command                   check_http
}
```

These two checks are related to the SSL options with the web server. Note that the checks change to port 443 if you use the --ssl option, they are testing to see if the web server can serve secure pages and if the web server certificate is valid for the next 21 days. The first check will test for a response within a limited time frame, 5 seconds for a warning or more than 10 seconds for a critical state.

```
define service{
        use                             generic-service
        host_name                       centos
        service_description             Secure HTTP
        check_command                   check_http! -w 5-c 10 --ssl
}
```

This check is focused on the certificate. In this example, if the certificate is good for more than 21 days an “OK” is returned. A warning state is triggered if the certificate has less than 21 days before it expires. A critical state is triggered when the certificate has expired.

```
define service{
        use                             generic-service
        host_name                       centos
        service_description             Certificate
        check_command                   check_http! -C 21
}
```

Both of the service checks above will return the following output in the Nagios web interface.

```
OK - Certificate will expire on 05/25/2012 23:59.
```

This usage of check_http allows you to check to see if a directory requiring authorization with username and password is working correctly. Note that the check_http has been redefined to check_http_auth so that additional arguments can be used. The service definition includes the IP Address of the server, the directory that requires authentication (-u/sales) and the username and password required to access the directory. Each is separated by a “!”. Note the command definition included.

```
define service{
        use                             generic-service
        host_name                       centos
        service_description             Sales Authorization
        check_command                   check_http_auth!192.168.5.1 -u/sales!tom!
user_password
}


define command{
        command_name    check_http_auth
        command_line    $USER1$/check_http -H $ARG1$ -a $ARG2$:$ARG3$
        }
```

If the user login is not correct warning will be issued with the "401 Authorization Required". This enables you to verify password changes and integrity. However, leaving a plain text password in the Nagios config files is not the best idea.

Monitor Linux with NRPE

The Nagios Remote Plugin Executor or NRPE allows you to execute programs for monitoring purposes on the remote server. One advantage of NRPE is that it does not require a login to perform the tests on the remote server.

NRPE allows you to monitor internal aspects of a Linux server from the Nagios server. When you monitor public ports like HTTP you can determine if the web server is running by using these service checks, but you are not able to monitor other aspects of the server which you may need information on, which is why you will want to use NRPE.

Set Up the Host to be Monitored with NRPE

The first thing to do with the host to be monitored by Nagios is to install the required applications.

NRPE From Source

These instructions pertain to the installation of the daemon and the plugins which are both required for the client to be monitored. This is different than setting up the Nagios server.

```
      cd /usr/local/src
      wget http://sourceforge.net/projects/nagios/files/nrpe-2.x/nrpe-2.12/nrpe- 2.12.tar.gz/download
      tar zxvf nrpe-2.12.tar.gz
      cd nrpe-2.12

You will need to install support for ssl, xinetd and compiling tools.
      yum install -y mod_ssl openssl-devel xinetd gcc make

      ./configure --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib

*** Configuration summary for nrpe 2.12 03-10-2008 ***:

 General Options:
 -----------------------------------
 NRPE port:    5666
 NRPE   user:   nagios
 NRPE  group:  nagios
 Nagios  user: nagios
 Nagios        group:
 nagios

      make
      make install
      make install-plugin
      make install-daemon
      make install-daemon-config
      make install-xinetd
```

Install the Daemon xinetd
The xinetd superdaemon has replaced inetd on most Linux distributions today. xinetd has become more popular because of security restrictions that can be placed on those who access the daemons managed by xinetd. xinetd also provides better protection from denial of service attacks, better log management, and more flexibility. Both inetd and xinetd only work with daemons that provide connections over a network.

How to Protect the NRPE Daemon
Server daemons must be protected to be effective. There is no perfect or complete option, but there are definite ways to minimize the risk.

1. **Limit Connections to Daemons**
   Connections to daemons can be limited by using several powerful tools. Iptables firewall is probably the most flexible and powerful tool than an administrator has access to. However, it is at the same time the most complex. Tcp_wrappers is a tool that is easy to use and works with most daemons to limit access to daemons to specific subnets or IP Addresses.
2. **Limit the Number of Connections**
   Your network and hardware can only handle a limited number of connections safely. When connections push your resources to the limit you will often see vulnerabilities appear that would not normally exist. When resources begin to fail some options and security programs cannot function to their full extent.

You will need to install xinetd and make sure you have a file in /etc/xinetd.d called nrpe on the client and it looks like this:

```
# default: off
# description: NRPE (Nagios Remote Plugin Executor)
service nrpe
{
        flags           = REUSE
        type            = UNLISTED
        port            = 5666
        socket_type     = stream
        wait            = no
        user            = nagios
        group           = nagios
        server          = /usr/sbin/nrpe
        server_args     = -c /usr/local/nagios/etc/nrpe.cfg --inetd
        log_on_failure  += USERID
        disable         = no
        only_from       = 127.0.0.1 192.168.5.50
}
```

These are the two most important lines. By default all daemons monitored by xinetd are disabled so the default line says “disable = yes”. The “only_from” line allows you to determine which machines can monitor this server using NRPE, this is where you will enter the IP Address for the Nagios server as well as the localhost.

```
      disable         = no
      only_from       = 127.0.0.1 192.168.5.50
```

Edit /etc/services and add this line:

```
nrpe            5666/tcp                        # Nagios Remote Monitoring
```

Restart xinetd and view the log at /var/log/daemon.log

```
      service xinetd restart
      tail /var/log/daemon.log
```

Look for errors to correct.

Edit the /usr/local/nagios/etc/nrpe.cfg .
Change your allowed_hosts address to reflect the nagios monitoring server. You should also allow the localhost so that you can do testing if necessary.

```
allowed_hosts=127.0.0.1 192.168.5.180
```

The basic plugins that are running for you initially are these listed below.

```
command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c 10
command[check_load]=/usr/local/nagios/libexec/check_load -w 15,10,5 -c 30,25,20
command[check_hda1]=/usr/local/nagios/libexec/check_disk -w 20 -c 10 -p /dev/hda1
command[check_zombie_procs]=/usr/local/nagios/libexec/check_procs -w 5 -c 10 -s Z
command[check_total_procs]=/usr/local/nagios/libexec/check_procs -w 150 -c 200
```

Change ownership on the /usr/local/nagios/etc/nrpe.cfg

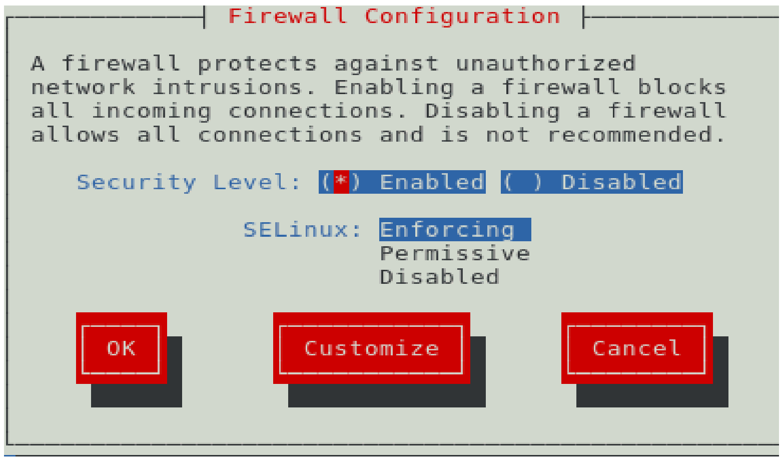
```
      chown nagios /usr/local/nagios/etc/nrpe.cfg*
```

Firewall
You will need to verify that the firewall will allow your Nagios server to access the Linux server you are testing on port 5666.

If the Linux server to be monitored is CentOS it probably has the lokkit interface to manage the firewall. At the command line type:

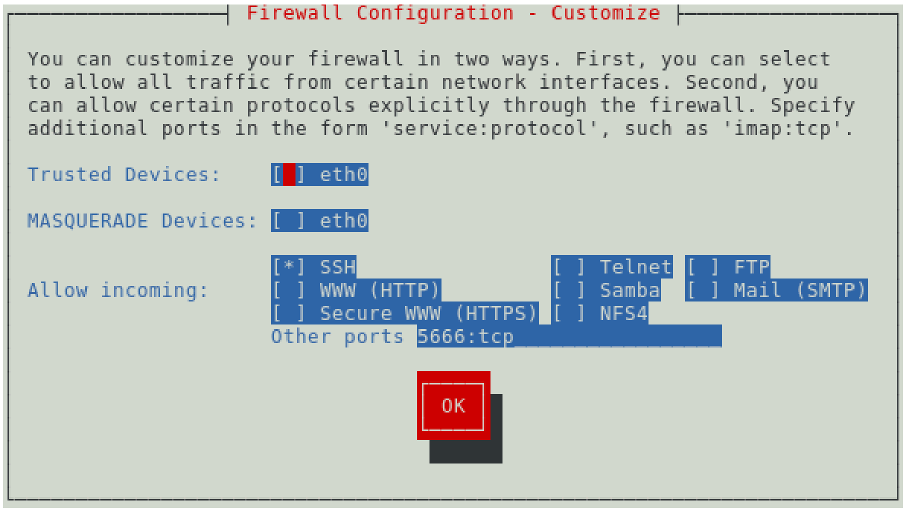
```
      lokkit
```

The firewall interface will open so you can manage the ports that are open on the Linux machine. Use the tab to go to the "Customize" option.



```
Firewall Configuration

A firewall protects against unauthorized
network intrusions. Enabling a firewall blocks
all incoming connections. Disabling a firewall
allows all connections and is not recommended.

   Security Level: (*) Enabled ( ) Disabled

             SELinux: Enforcing
                      Permissive
                      Disabled
```

The ports you want to enter that are not in the default options can be added by using the port number followed by a colon and whether it is tcp or udp. In this example 5666:tcp has been added to enable the Nagios server access on this port.



```
Firewall Configuration - Customize

You can customize your firewall in two ways. First, you can select
to allow all traffic from certain network interfaces. Second, you
can allow certain protocols explicitly through the firewall. Specify
additional ports in the form 'service:protocol', such as 'imap:tcp'.

Trusted Devices:     [ ] eth0

MASQUERADE Devices: [ ] eth0

                     [*] SSH                  [ ] Telnet [ ] FTP
Allow incoming:      [ ] WWW (HTTP)           [ ] Samba  [ ] Mail (SMTP)
                     [ ] Secure WWW (HTTPS)   [ ] NFS4
                     Other ports 5666:tcp
```

Save your changes.

tcp_wrappers
Now set up your tcp_wrappers.

Edit the /etc/hosts.allow file first and make sure that you maintain your SSH connection to manage the server and then add a line for NRPE for your Nagios server to have access.

```
# hosts.allow  This file describes the names of the hosts which are
#              allowed to use the local INET services, as decided
#              by the '/usr/sbin/tcpd' server.
#
SSHD:   192.168.5.51
```

```
NRPE:   192.168.5.51
```

Now edit /etc/hosts.deny. Use the one line to deny ALL hosts and ALL services. This will then only allow what is in /etc/hosts.allow.

```
# The portmap line is redundant, but it is left to remind you that
# the new secure portmap uses hosts.deny and hosts.allow.  In particular
# you should know that NFS uses portmap!
ALL:    ALL
```

This completes the basic configuration of the host that you will monitor.

Set Up the Nagios Server

One you have the remote host set up you will need to set up the Nagios monitoring server. First install the nrpe plugin.

NRPE From Source
These instructions pertain to the installation of the plugin only which is different that for the client to be monitored. NRPE plugins only need to be installed on the Nagios server.

```
      cd /usr/local/src
      wget http://sourceforge.net/projects/nagios/files/nrpe-2.x/nrpe2.12/nrpe-
2.12.tar.gz/download
      tar zxvf nrpe-2.12.tar.gz
      cd nrpe-2.12

You will need to install support for ssl, xinetd and compiling tools.
      yum install -y mod_ssl openssl-devel xinetd gcc make

      ./configure --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib

*** Configuration summary for nrpe 2.12 03-10-2008 ***:

 General Options:
 --------------------------------
 NRPE port:    5666
 NRPE   user:  nagios
 NRPE  group:  nagios
 Nagios  user: nagios
 Nagios        group:
 nagios

      make
      make install
      make install-plugin
```

Do a Manual Check of the Remote Host
In order to verify that the remote host is functioning correctly do a manual check. Remember to allow port 5666/tcp on the remote host. Use the full path to check if the Nagios server can contact the remote host.

```
/usr/local/nagios/libexec/./check_nrpe -H 192.168.5.49 -c check_users
USERS OK - 2 users currently logged in | users=2;5;10;0
```

If you see output that is similar it is functioning correctly.

Create the Host Files
In order to monitor remote Linux boxes you will need to set up your template called "linux-box" or use a template that is already available. Then you will need to create a host entry for each remote box you will monitor.

```
define host{
        name                    linux-box
        use                     generic-host
        check_period            24x7
        check_interval          5
        retry_interval          1
        max_check_attempts      10
        check_command           check-host-alive
        notification_period     24x7
        notification_interval   30
        contact_groups          admins
        register                0
        }
define host{
        use                     linux-box
        host_name               dg
        alias                   Base
        address                 192.168.5.178
        }
```

Configure Services
Each service you want to monitor on the remote host must be entered individually. Here is an example of monitoring CPU load on the host "dg". Note: The "service_description" should be entered carefully as you may decide to use other addons for Nagios that are case sensitive to the names of the services. The check_nrpe command is used to access the remote server and then execute the Nagios plugin that is on the remote server and retrieve the information.

```
define service{
        use                     generic-service
        host_name               dg
        service_description     CPU Load
        check_command           check_nrpe!check_load
        }
```

Once this is complete you must restart your nagios server with:

```
      service nagios restart
```

If you get errors correct them.

Now you can check your connection by running the following command and using the IP Address of the remote box you want to monitor. You should get the return "NRPE" and version number if all is working.

```
/usr/local/nagios/libexec/./check_nrpe -H 192.168.5.178
NRPE v2.12
```

If you get this return then you have communication between the Nagios monitoring server and the remote host.

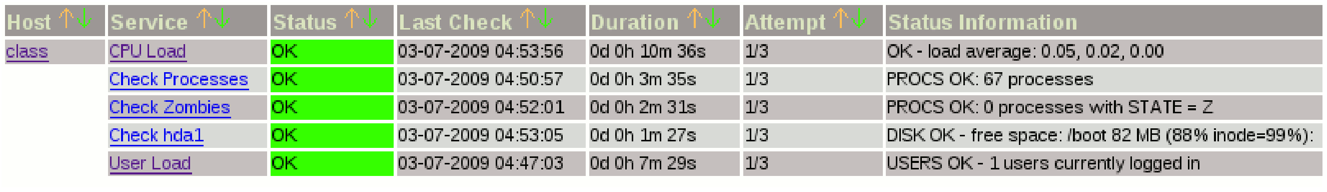
Create the NRPE Command Definitions
Before you can execute commands for NRPE on the Nagios server you will need to edit the commands.cfg and define the commands for NRPE. Here are two examples that you can use.

```
# NRPE Commands

define command{
      command_name    check_nrpe
      command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
      }
define command{
      command_name    check_nrpe2
      command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$ -a $ARG2$
      }
```

Configure the Checks
On the Nagios server you can monitor all of the defaults by placing the information in your services file.

```
define service{
        use                     generic-service
        host_name               class
        service_description     CPU Load
        check_command           check_nrpe!check_load
        }
define service{
        use                     generic-service
        host_name               class
        service_description     User Load
        check_command           check_nrpe!check_users
        }
define service{
        use                     generic-service
        host_name               class
        service_description     Check hda1
        check_command           check_nrpe!check_hda1
        }
define service{
        use                     generic-service
        host_name               class
        service_description     Check Zombies
        check_command           check_nrpe!check_zombie_procs
        }
define service{
        use                     generic-service
        host_name               class
        service_description     Check Processes
        check_command           check_nrpe!check_total_procs
        }
```

Once you have added these to your server restart Nagios and you should see that they are working.



Host	Service	Status	Last Check	Duration	Attempt	Status Information
class	CPU Load	OK	03-07-2009 04:53:56	0d 0h 10m 36s	1/3	OK - load average: 0.05, 0.02, 0.00
	Check Processes	OK	03-07-2009 04:50:57	0d 0h 3m 35s	1/3	PROCS OK: 67 processes
	Check Zombies	OK	03-07-2009 04:52:01	0d 0h 2m 31s	1/3	PROCS OK: 0 processes with STATE = Z
	Check hda1	OK	03-07-2009 04:53:05	0d 0h 1m 27s	1/3	DISK OK - free space: /boot 82 MB (88% inode=99%):
	User Load	OK	03-07-2009 04:47:03	0d 0h 7m 29s	1/3	USERS OK - 1 users currently logged in

Monitoring Windows with NSClient++

The Windows client NSClient++ can be used to both monitor a Windows machine with NSClient++ using the check_nt command or using NRPE. Because the configuration for both aspects involves the NSClient++ they are viewed together. The first step in setting up NRPE for Windows is to download a client for the Windows machine.

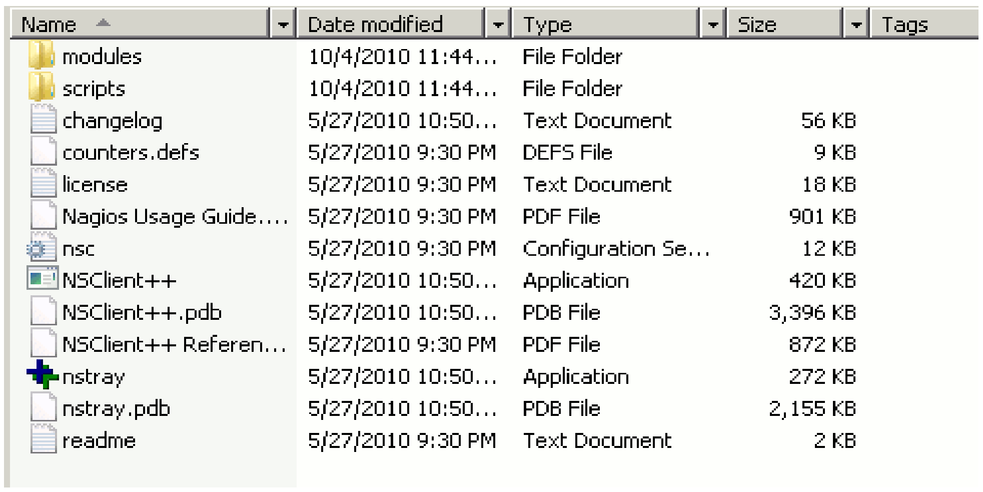
Download the NSCLient++ from http://sourceforge.net/projects/nscplus

This will provide a .zip file which you can unzip and it will provide the NSClient++-Win32-x.x.x folder.

Installation of NSClient++

Login as the Administrator to the server.
Create a directory under the C:\ drive and download NSClient++ into that drive. Unzip the file and enter the directory that was created.

Once you install you will have to make a note of the location of the install directory path.
Here is the contents of the directory.



Name	Date modified	Type	Size	Tags
modules	10/4/2010 11:44...	File Folder		
scripts	10/4/2010 11:44...	File Folder		
changelog	5/27/2010 10:50...	Text Document	56 KB	
counters.defs	5/27/2010 9:30 PM	DEFS File	9 KB	
license	5/27/2010 9:30 PM	Text Document	18 KB	
Nagios Usage Guide....	5/27/2010 9:30 PM	PDF File	901 KB	
nsc	5/27/2010 9:30 PM	Configuration Se...	12 KB	
NSClient++	5/27/2010 10:50...	Application	420 KB	
NSClient++.pdb	5/27/2010 10:50...	PDB File	3,396 KB	
NSClient++ Referen...	5/27/2010 9:30 PM	PDF File	872 KB	
nstray	5/27/2010 10:50...	Application	272 KB	
nstray.pdb	5/27/2010 10:50...	PDB File	2,155 KB	
readme	5/27/2010 9:30 PM	Text Document	2 KB	

On the Windows machine place the path for the .exe file in the run command and install the program.

```
C:\NSClient++-Win32-0.3.8\NSClient++.exe /install
```

You will see a security warning but continue the install.

Now start the program, note your path may be different.

```
C:\NSClient++-Win32-0.3.8\NSClient++.exe /start
```

In order to stop the program use this command.

```
C:\NSClient++-Win32-0.3.8\NSClient++.exe /stop
```

You can test with:

```
C:\NSClient++-Win32-0.3.8\NSClient++.exe /test
```

If you make any changes to the configuration, stop the service and restart it.

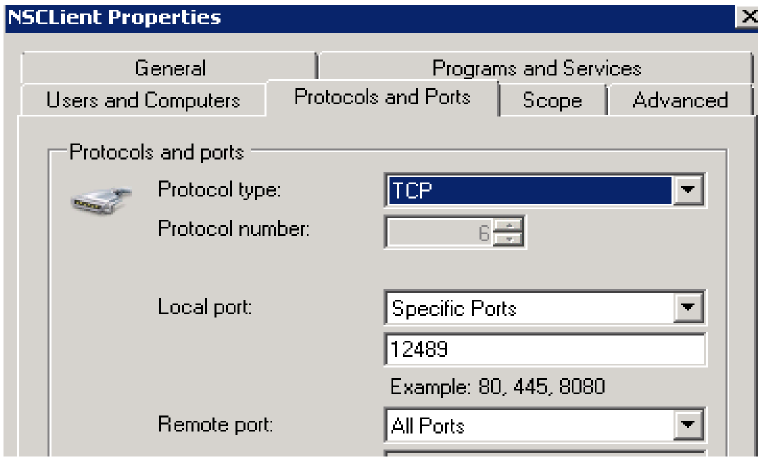
Edit the NSC.ini file that is in the NSClient directory. Note that the file is divided by keywords placed in brackets. First go to the [modules] section and edit the checks that you want to use. Uncomment the lines that you see below. The FileLogger.dll will log the activities of the NSClient++. CheckDisk.dll will check for file size and hard disk use. The CheckSystem.dll will check for memory, uptime, service stats and processes. You will also need to uncomment the NSClientListener.dll and the NRPEListener.dll in order to communicate with Nagios.

In order to use some of the options available with NSClient++ you need to allow to additional features. There are characters that need to be used with commands |`&<>'"\[]{} that you will want allow, "nasty_meta_chars". The "allow_arguments" will allow NRPE parameters to be passed along. Now there some security issues with enabling this option so you need to consider that factor.

Go to the global section, [Settings], and be sure to limit the access to the Windows server that you are going to monitor. Under the Allowed Hosts section enter the local host and any other connections that you want to enable. These addresses will be separated by a comma.

```
allowed_hosts=127.0.0.1/32,192.168.5.50
```

In the Windows firewall open two ports, 5666 for NRPE and 12489 for NSClient++. Both are TCP ports. You can see in the example how it should look when you review the firewall.



Protocols and ports

Protocol type: TCP

Protocol number: 6

Local port: Specific Ports

12489

Example: 80, 445, 8080

Remote port: All Ports

Limit access to these ports to the Nagios server only.

NSClient++ and NRPE

The NSC.ini file contains several settings for using NRPE. Look for two sections that relate to NRPE and the modules section.

```
[modules]
FileLogger.dll
CheckSystem.dll
CheckDisk.dll
NRPEListener.dll
CheckEventLog.dll

[NRPE]
allow_arguments=1
allow_nasty_meta_chars=1
allowed_hosts=127.0.0.1/32,192.168.5.50
Port=5666
```

This of course assumes you will open port 5666 on the Windows machine. If you can, limit the access to this port only to the Nagios server for security. If you see this output in your web interface make sure that port 5666 is open and that you have started the client.

FileLogger.dll
FileLogger provides an internal log of NSClient++ but does not actually provide any checks.

CheckSystem.dll
This dll allows for checks of the CPU,memory, uptime, services, and process states.

CheckDisk.dll
CheckDisk allows checks for file size, and hard drive usage.

NRPEListener.dll
The NRPEListener is the key to providing functionality to NRPE.

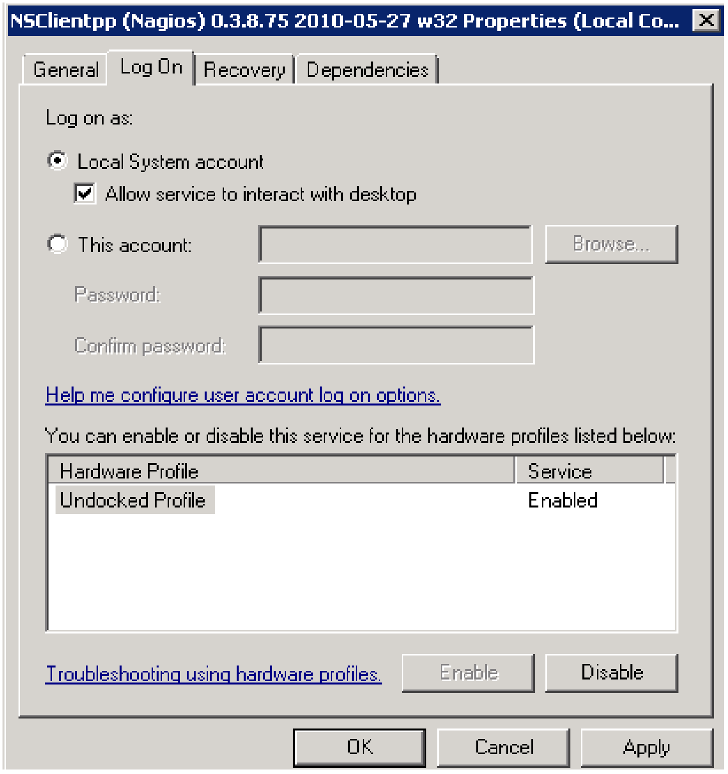
SysTray.ddl
The SysTray installs an icon to use for access to NSClient++. You will need to install NSClient first with:

```
      nsclient++ /install
```

Then you will need to run this command:

```
      nsclient++ -noboot SysTray install
```

Finally, open the services manager and edit the NSClientpp service to allow it to interact with the desktop.


NSClientpp (Nagios) 0.3.8.75 2010-05-27 w32 Properties (Local Co...
General
Log On
Recovery
Dependencies
Browse...
Help me configure user account log on options.
Troubleshooting using hardware profiles.
Enable
Disable
OK
Cancel
Apply

Log on as:

[✓ Local System account]

[✓] Allow service to interact with desktop

[ ] This account:

Password:

Confirm password:

You can enable or disable this service for the hardware profiles listed below:

Hardware Profile	Service
Undocked Profile	Enabled

Note: The SysTray feature only works with XP and older machines!

```
CheckEventLog.dll
```

The NRPE Handlers represent the actual commands that will be used.

```
[NRPE Client Handlers]
command[check_users]=/usr/local/nagios/libexec/check_users -w 5 -c 10
check_disk1=/usr/local/nagios/libexec/check_disk -w 5 -c 10
check_disk_c=inject CheckFileSize ShowAll MaxWarn=1024M MaxCrit=4096M
```

Once you have the remote host set up you will need to set up the Nagios monitoring server. First install the nrpe plugin.

NRPE From Source on Nagios Server

These instructions pertain to the installation of the plugin only which is different that for the client to be monitored. NRPE plugins only need to be installed on the Nagios server.

```
      cd /usr/local/src
      wget http://sourceforge.net/projects/nagios/files/nrpe-2.x/nrpe-2.12/nrpe- 2.12.tar.gz/download
      tar zxvf nrpe-2.12.tar.gz
      cd nrpe-2.12
```

```
You will need to install support for ssl, xinetd and compiling tools.
      yum install -y mod_ssl openssl-devel xinetd gcc make

      ./configure --with-ssl=/usr/bin/openssl --with-ssl-lib=/usr/lib

*** Configuration summary for nrpe 2.12 03-10-2008 ***:

 General Options:
 ---------------------------------
 NRPE port:    5666
 NRPE  user:   nagios
 NRPE  group:  nagios
 Nagios  user: nagios
 Nagios        group:
 nagios

      make
      make install
      make install-plugin
```

Once it is up an running check your connection.

```
      /usr/local/nagios/libexec/./check_nrpe -H 192.168.5.14
I (0.3.5.1 2008-09-24) seem to be doing fine...
```

If you see errors you will need to correct them, use the log for locating the errors.

Internal NSClient ++ Functions

There are a number of internal functions that can be called with the inject command and NRPE and are usually combined with check_nt. The check_nt plugin makes it easy to use these functions. However, if you want to fine tune the options that are available you may want to use NRPE and the inject command. Following is a list of modules with their functions.

```
CheckDisk         - CheckFileSize, CheckDriveSize
CheckSystem       - CheckCPU, CheckUpTime, ChekServiceState,
CheckProcState, CheckMem, CheckCounter
CheckeventLog     - CheckEventLog
CheckHelpers      - CheckAlwaysOk, CheckAlwaysCRITICAL,
CheckAlways,WARNING, CheckMultiple
```

You can use aliases with external commands to do checks. The advantage of setting up the aliases is not so much the alias by itself but it will allow you to use the CheckMultiple function if you want to. Check to see if you can get it to work from the command line on the Nagios server first. If that works you can proceed.

```
      ./check_nrpe -H 192.168.5.14 -c CheckCPU -a warn=80 crit=90 time=20m
time=10s time=4
OK CPU Load ok.|'20m'=0%;80;90; '10s'=0%;80;90; '4'=0%;80;90;
```

You will need to define Service checks on Nagios server as usual. Note NRPE is used to make the connection and then run the alias that you will set up.

```
define service{
      use                    generic-service
      host_name              winserver
      service_description    CPU Load
      check_command          check_nrpe!alias_cpu
}
define service{
      use                    generic-service
      host_name              winserver
      service_description    Check Services
      check_command          check_nrpe!alias_service
}
define service{
      use                    generic-service
      host_name              winserver
      service_description    Free Space
      check_command          check_nrpe!alias_disk
}
```

Once the Windows server you will need to edit the “External Alias” section and create or uncomment the aliases that are there with the levels.

```
[External Alias]
alias_cpu=checkCPU warn=80 crit=90 time=5m time=1m time=30s
alias_disk=CheckDriveSize MinWarn=10% MinCrit=5% CheckAll FilterType=FIXED
alias_service=checkServiceState CheckAll
```

You also need to verify that the “modules” section has the uncommented “CheckExternalScripts.dll ” as you see below so checks can be made.

```
[modules]
CheckExternalScripts.dll
```

Restart your NSCLient++ on the Windows server and nagios on the Nagios server.

If you wanted to perform multiple checks at one time, thus saving network and server resources, you could use the CheckMultiple function. The CheckMultiple function will become an alias for any number of commands that you will want to run. The format should be like this:

alias=alias_name command=   command=   command=

Remove the aliases that you may have had previously and place them all on the CheckMultiple alias.

```
[External Alias]
alias_multiple=CheckMultiple command=checkCPU warn=80 crit=90 time=5m time=1m
time=30s command=CheckDriveSize MinWarn=10% MinCrit=5% CheckAll FilterType=FIXED
command=checkServiceState CheckAll
```

You will need to set up a service on the Nagios server to reflect your settings in the nsc.ini on the windows server.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     Multiple
        check_command           check_nrpe!alias_multiple
}
```

Here you can see the second line down the Multiple check is running. It checks the number of checks you have entered and then the "bad news rises to the top". In other words as you can see any issues with one check can trigger the "CRITICAL" state. If you look closely the text specifically says the other checks are OK.

NSClient++ and check_nt

The check_nt plugin is a standard plugin that is available and ready to go. It can be extended with The key to getting this to work is to uncomment NSClientListener.dll and to open the port 12489/TCP.

```
[modules]
FileLogger.dll
CheckSystem.dll
CheckDisk.dll
NSClientListener.dll
CheckEventLog.dll
```

[NSClient]
allowed_hosts=192.168.4.3

Security Tip: Use the "allows_hosts" option to protect your Windows server so only the Nagios server can access this daemon.

If allowed hosts is used in this section it will take precedence over the "Settings" where there is also an option to enter "allowed_hosts".

check_net plugin
The check_nt plugin is the standard plugin that is used with NSClient++ and is a plugin included in the nagios-plugins install.

-H	host address
-v	command that is executed
-p	port, this port is often changed to 12489
-w integer	warning integer
-c integer	critical integer
-l	use a parameter
-d	option, the -d SHOWFAIL option shows only checks that fail, the -d SHOWALL will show all
-s	password sent to Windows server
-t	timeout, default is 10 seconds

There are a number of easy to use service definitions. Here are some basic ones to get started. Each of these services using check_nt show that the check_nt plugin is separated from the service with “!”. This is also seen in the default check_net commands definition in commands.cfg. Note in this example the port is determined with “-p 12489”.

```
# 'check_nt' command definition
define command{
        command_name    check_nt
        command_line    $USER1$/check_nt -H $HOSTADDRESS$ -p 12489 -v $ARG1$ $ARG2$
        }
```

The first check to try, which is actually the easiest to get started is the the test for the clientversion. Try this one first and once it is running then you will know that communication is working correctly.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     NSClient++ Version
        check_command           check_nt!CLIENTVERSION
        }
```

Monitor the uptime of the Windows server with UPTIME.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     Uptime
        check_command           check_nt!UPTIME
        }
```

Create a service for monitoring CPU load. When you define this service the "-l" is a parameter that has three settings. The first setting "5" is the average load over 5 minutes. Of course, you can adjust that for your needs. The second and third settings are the warning level 80% load and the critical level 90% load. Again, these must be averages over the time period of 5 minutes.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     CPU Load
        check_command           check_nt!CPULOAD!-l 5,80,90
        }
```

You can modify this check so that you can evaluate averages on various time intervals. These time intervals will need to be added in three entries. In the example below you can see "5,80,90" and "15,75,87". The first entry are averages for 5 minutes and the second entry is the averages fro 15 minutes. You can also see the averages are lower in the second entry. This is typically how you would want to evaluate CPU Load as spikes over a short period of time are not a concern but high averages over a longer period are certainly problematic.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     CPU2 Load
        check_command           check_nt!CPULOAD!-l 5,80,90,15,75,87
        }
```

This check is to evaluate memory use on the server with a warning when it reaches 80% and a critical level at 90%.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     Memory
        Usage
        check_command           check_nt!MEMUSE!-w 80 -c 90
        }
```

The C:/ drive is typically the installation drive for a Windows machine. Of course this will be one drive or partition that you will want to monitor. The example show next has the parameter(-l) for first the drive "c" and then the warning level "-w 80" and the critical level "-c 90". Adjust the parameters to your needs.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     C:\ Drive
        Space
        check_command           check_nt!USEDDISKSPACE!-l c -w 80 -c 90
       }
```

If you wanted to monitor another partition, in this example drive "e", then just substitute the drive letter you want to monitor.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     E:\ Drive
        Space
        check_command           check_nt!USEDDISKSPACE!-l e -w 80 -c 90
        }
```

You can use check_nt to monitor any service on the Windows machine. There are two options you can use to specifically monitor a service. The "-d" provides the option to either use SHOWFAIL option shows only checks that fail or the SHOWALL that will show all services. Now if you are only monitoring one service with the check you will want to use "SHOWALL". If you were trying to monitor all services with one check then you would probably want "SHOWFAIL".

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     W3SVC
        check_command           check_nt!SERVICESTATE!-d SHOWALL -l W3SVC
        }
```

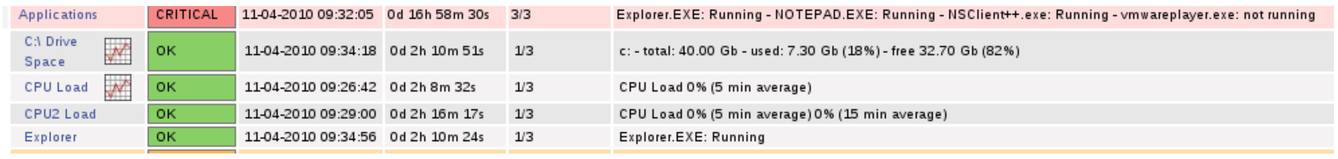
Here is an example of monitoring explore.exe, vmplayer.exe and notepad.exe. Simply by changing the exe on the parameter you can choose specific applications.

```
define service{
        use                     generic-service
        host_name               winserver
        service_description     Explorer
        check_command           check_nt!PROCSTATE!-d SHOWALL -l Explorer.exe
        }
define service{
        use                     generic-service
        host_name               winserver
        service_description     VMware
        check_command           check_nt!PROCSTATE!-d SHOWALL -l vmplayer.exe
        }
define service{
        use                     generic-service
        host_name               winserver
        service_description     Notepad
        check_command           check_nt!PROCSTATE!-d SHOWALL -l notepad.exe
        }
```

You also have the option to include all of the mission critical applications in one check. Here you want to make sure to list each application separated by a command as you can see. If you use the option "SHOWALL" it will list both those that are running as well as those that are not. The bad news rises to the top so if one is not running the check will be in the critical state.   If you just want to know which ones are not running then use "SHOWFAIL".

```
define service{
        use                     generic-service
        host_name               win2008,exchange
        service_description     Applications
        check_command          check_nt!PROCSTATE!-d SHOWALL -l
explorer.exe,notepad.exe,nsclient++.exe,vmwareplayer.exe
        }
```

Here you can see that the Critical state is listed because only one application out of the list is not running. Bad news rises up to the top.



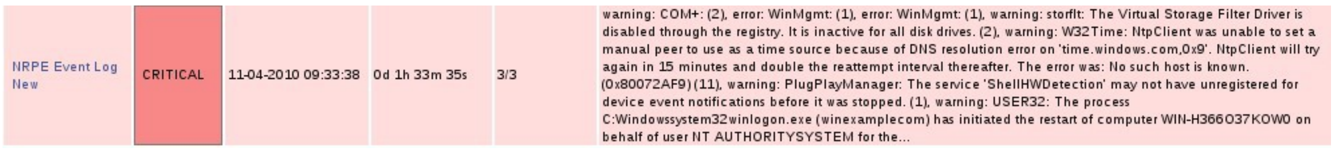
Applications	CRITICAL	11-04-2010 09:32:05	0d 16h 58m 30s	3/3	Explorer.EXE: Running - NOTEPAD.EXE: Running - NSClient++.exe: Running - vmwareplayer.exe: not running
C:\ Drive Space	OK	11-04-2010 09:34:18	0d 2h 10m 51s	1/3	c: - total: 40.00 Gb - used: 7.30 Gb (18%) - free 32.70 Gb (82%)
CPU Load	OK	11-04-2010 09:26:42	0d 2h 8m 32s	1/3	CPU Load 0% (5 min average)
CPU2 Load	OK	11-04-2010 09:29:00	0d 2h 16m 17s	1/3	CPU Load 0% (5 min average) 0% (15 min average)
Explorer	OK	11-04-2010 09:34:56	0d 2h 10m 24s	1/3	Explorer.EXE: Running

event_log Monitoring

The event log on a Windows server can be a critical aspect of locating information on the server. Here is an example of the service check using NRPE and the alias_event_log.

```
define service{
        use                    generic-service
        host_name              win2008
        service_description    NRPE Event Log New
        check_command          check_nrpe!alias_event_log
        }
```

Note that if one element has a problem it will create a critical state as you see here.



NRPE Event Log New	CRITICAL	11-04-2010 09:33:38	0d 1h 33m 35s	3/3	warning: COM+: (2), error: WinMgmt: (1), error: WinMgmt: (1), warning: storflt: The Virtual Storage Filter Driver is disabled through the registry. It is inactive for all disk drives. (2), warning: W32Time: NtpClient was unable to set a manual peer to use as a time source because of DNS resolution error on 'time.windows.com,0x9'. NtpClient will try again in 15 minutes and double the reattempt interval thereafter. The error was: No such host is known. (0x80072AF9) (11), warning: PlugPlayManager: The service 'ShellHWDetection' may not have unregistered for device event notifications before it was stopped. (1), warning: USER32: The process C:Windowssystem32winlogon.exe (winexamplecom) has initiated the restart of computer WIN-H366O37KOW0 on behalf of user NT AUTHORITYSYSTEM for the...

Here is the actual output that you can find in the logs of the Nagios server.

```
Nov  4 09:13:43 nag2 nagios: SERVICE NOTIFICATION: nagiosadmin;win2008;NRPE
Event Log New;CRITICAL;notify-service-by-email;warning: COM+:    (2), error:
WinMgmt: (1), error: WinMgmt:                            (1), warning: storflt:
The Virtual Storage Filter Driver is disabled through the registry. It is
inactive for all disk drives. (2), warning:
W32Time: NtpClient was unable to set a manual peer to use as a time source
because of DNS resolution error on time.windows.com,0x9. NtpClient will try again
in 15 minutes and double the reattempt interval thereafter. The error was: No
such host is known. (0x80072AF9) (11), warning: PlugPlayManager: The service
ShellHWDetection may not have unregistered for device event notifications before
it was stopped. (1), warning: USER32: The process
C:\Windows\system32\winlogon.exe (winexamplecom) has initiated the restart of
computer WIN-H366O37KOW0 on behalf of user NT AUTHORITY\SYSTEM for the...
```

NSCLient++ Password

The password feature allows you to create a password that will be used by Nagios to log into the Windows server. This password has several options. First you can enter the password in plain text in the nsc.ini and in the command definition for check_nt as you see below.

```
[Settings]
password=your_password
```

When you use the password option in nsc.ini, you will need to modify the check_nt command so the password can be transferred. Edit the commands.cfg

```
command_line  check_nt -H $HOSTADDRESS$ -p 12489 -s your_password -v $ARG1$
$ARG2$
```

The use of the obfuscated_password option seems to be broken. In order to create the password go to the command line on the Windows machine and execute this command:

```
NSClient++ /encrypt
```

You will be asked to enter you password and it will obfuscate not encrypt the password. The shorter the word the shorter the password that is created. This method is both unreliable and undocumented. You are better off using plain text than this method as at least you know what is going on.