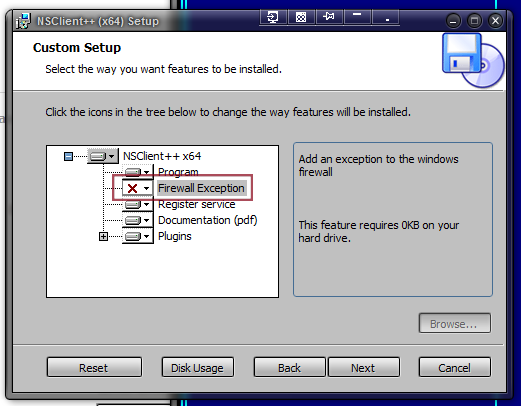
**NAGIOS WINDOWS CLIENT CONGIGURATION**

**Installation**

The installer uses MSI (msiexec) which means it is a standard Microsoft installer (now you should be happy). This in turns means you can use standard features to customize the installer in addition to some defines you can set specific to NSClient++.

Installation of NSClient++ is generally achieved using the graphical interface presented by the MSI package. You can choose which feature you want to install for the application. For example, to install NSClient++ without the firewall exception, you would disable the feature as shown in the following screenshot:

[](http://nsclient.org/nscp/attachment/ticket/376/firewall-exception.png)

On the command line, the graphical interface can be called with the following command:

msiexec /i <<<MSI FILE>>>

**Unattended install**

In the context of deploying the application on computers in a large network, one would like to install NSClient++ automatically and quietly. Using *msiexec*, you can choose which features you want to install. As of now (version 0.3.8), the list of features presented by the package is the following:

* ProductFeature? -- Install the binaries of the application
  + MainProgram? -- Install the main program binary
  + FireWallException? -- Install an exception to the Windows firewall
  + ServiceRegistration? -- Register the NSClient++ service
  + Documentation -- Install the PDF documentation
  + Plugins -- Install plugins to NSClient++
    - CheckPlugins? -- Built-in checks for the system
    - NRPEPlugins -- NRPE listener, lets Nagios query for execution of a remote plugin
    - NSCPlugins -- NSClient listener, adds built-in check remote commands called with *check\_nt* on the Nagios server
    - NSCAPlugin -- Plugin that submits passive check data to Nagios. Implements checks as a push to Nagios instead of a pull
    - SampleScripts? -- Install some sample client-side scripts to use with NRPE
* ALLOWED\_HOSTS -- The IP which is allowed to connect to the client (e.g. the Nagios Server)

To install NSClient++ quietly with all the features enabled, run:

msiexec /i <<<MSI FILE>>> /quiet

Quiet install without restart and with all features but the firewall exception:

msiexec /i <<<MSI FILE>>> /quiet /norestart ADDLOCAL=ALL REMOVE="FireWallException"

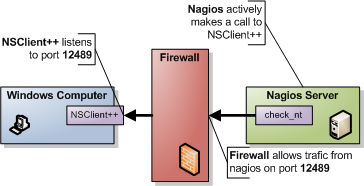
Passive install (displays progress bar) without restart and with several feature exceptions:

msiexec /i <<<MSI FILE>>> /passive /norestart ADDLOCAL="ALL" REMOVE="Documentation,NSCPlugins,NSCAPlugin,SampleScripts"

There is of course more you can do but this is a quick start. If you need more details lookup how to work with MSI. A good starting point would here to install the NSClient++ on a fresh machine via "msiexec /i NSClient++-<TheLatestVersion?>.msi /l\*vx C:\00Install\logfile.txt" and check the logfile for entry's like "PROPERTY CHANGE: Adding ALLOWED\_HOSTS property. Its value is '10.201.1.1'."

**CONFIGURING AND CHECKING**

# Using NSClient++ from nagios with check\_nt

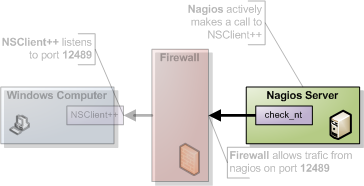
[](http://nsclient.org/nscp/attachment/wiki/doc/usage/nagios/nagios-active-nsclient.png)

**NOTICE** The check\_nt client support is available for compatibility mode and it is not recommended unless you already have an infrastructure around it. There are several features you will not be able to use with this scheme. I would recommend using NRPE instead.

This is the simplest and most locked in way to use NSClient++ you are limited to a handful of checks and there is no way to exploit the power of NSClient++ from here. The good though is that it is very simple to use and setup so it might be a good way to start. It is also the "only" way to have password protection. But note that since there is no encryption the password is sent as clear text so if you are compromised it will be easy to find. Also since check\_nt is distributed in the "normal plugin kit" you undoubtedly already have everything you need on the nagios side.

Nagios have their own guide for setting this up here [​http://nagios.sourceforge.net/docs/3\_0/monitoring-windows.html](http://nagios.sourceforge.net/docs/3_0/monitoring-windows.html)

## 1. Nagios command line

[](http://nsclient.org/nscp/attachment/wiki/doc/usage/nagios/nagios-active-nsclient-002.png)

Using check\_nt from the command line of your Nagios server is usually the bast place to start. If you are not familiar with it I would recommend you try this out as it will save you a lot of time when you are getting started or trying out new things. It is a good way to eliminate errors and you wont have to bother with restarting/waiting on Nagios when you need to make changes. To access NSClient++ from the Nagios server via the NSClient protocol you use a program (comes with the default plugins) called check\_nt.

check\_nt -H <client ip> -p <port> -v <command> ...

* client ip = the IP of the server you want to monitor (i.e. where NSClient++ i installed).
* port = the port you are using for the NSClientListener (defaults to 12489)
* command = is the various things you can monitor. The various commands all take different additional arguments which are all showed in the help.

To check the CPU load you can for instance run the following (assuming your windows server has 10.0.0.1 as ip address)

check\_nt -H 10.0.0.1 -p 12489 -v CPULOAD -w 80 -c 90 -l 5,80,90,10,80,90

CPU Load 0% (5 min average) 0% (10 min average)

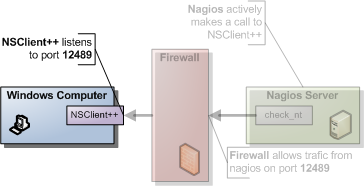
|'5 min avg Load'=0%;80;90;0;100 '10 min avg Load'=0%;80;90;0;100

If you instead got any of the following don't worry, it is because your NSClient++ is not configured properly and, we will solve that in the next section.

CRITICAL - Socket timeout after 10 seconds

Return code of 139 is out of bounds

## 2. NSClient++ configuration

[](http://nsclient.org/nscp/attachment/wiki/doc/usage/nagios/nagios-active-nsclient-001.png)

The first thing you need to do is decide which modules you want to use. NSClient++ is modular by design this means you only use the features you want (and if you want you can use all of them). The modules can be roughly divided into two kinds.

1. check commands
2. protocols (and utility modules).

The first kind is the one you \*use\* it responds to your commands and "finds" monitored data for you. The second kind is the one that allows you to talk to the first kind. When it comes to modules for the NSClient mode you will need the following:

|  |  |  |
| --- | --- | --- |
| Module | Description | Commands |
| [CheckSystem](http://nsclient.org/nscp/wiki/CheckSystem) | Handles many system checks | CPU, MEMORY, COUNTER etc |
| [CheckDisk](http://nsclient.org/nscp/wiki/CheckDisk) | Handles Disk related checks | USEDDISKSPACE |
| NSClientServer | Listens and responds to incoming requests from nagios | N/A |

To enable modules you edit the [modules] section in the nsc.ini file and your section should look something like this:

[/modules]

CheckSystem=enabled

CheckDisk=enabled

NSClientServer=enabled

The other things you need to configure is who is allowed to ask questions (which ip addresses) this is done either under the [Settings] section (globally) or under the [NSClient] (locally). I would recommend using the [Settings] section as it will simplify things when you start using NRPE. The keys you need to change are allowed\_hosts and password. And the value should be:

* allowed hosts = A list of addresses that is allowed to ask questions (i.e. your nagios ip).
* password = The password to use.

The result should look like this (assuming you don't use a password and the nagios ip address is 10.0.0.2):

[/settings/default]

;password=secret-password

allowed hosts=10.0.0.2

Notice that since you don't use a password that key is commented out (;).

### allowed hosts

If you do not configure the allowed hosts directive correctly you might get this in Nagios:

Return code of 139 is out of bounds

And if you check nsclient.log you would see:

2013-03-20 17:23:50: e:D:\source\nscp\trunk\include\check\_nt/server/protocol.hpp:65: Rejected connection from: ::ffff:10.83.14.251

### trying it

**Don't forget to restart NSClient++** after you make changes to the nsclient.ini file.

net stop nscp

net start nscp

Now feel free to try the command line agent again and hopefully things should work out perfectly. Run the following command from your nagios server.

check\_nt -H 10.0.0.1 -p 12489 -v CPULOAD -w 80 -c 90 -l 5,80,90,10,80,90

CPU Load 0% (5 min average) 0% (10 min average)

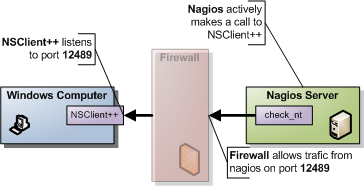
|'5 min avg Load'=0%;80;90;0;100 '10 min avg Load'=0%;80;90;0;100

check\_nt -H 10.0.0.1 -p 12489 -v USEDDISKSPACE -d SHOWALL -l c

c:\ - total: 149.00 Gb - used: 12.93 Gb (9%) - free: 136.07 Gb (91%)

|'c:\ Used Space'=12.93Gb;0.00;0.00;0.00;149.00

## 3. Solving problems

[](http://nsclient.org/nscp/attachment/wiki/doc/usage/nagios/nagios-active-nsclient-003.png)

A good way to find and solve problems is to run nsclient++ in "test" mode this is done by stopping the service and starting it in "test" mode.

net stop nscp

nscp test

... test mode ... (quit with: exit)

net start nscp

When in test mode you will get a lot of interesting log messages when things are happening so it is fairly simple to figure out what is wrong. To try this out do the following:

net stop nscp

nscp test

What you will see is the following output (or something similar):

Launching test mode - client mode

d NSClient++.cpp(1106) Enabling debug mode...

d NSClient++.cpp(494) Attempting to start NSCLient++ - 0.3.7.7 2009-07-05

d NSClient++.cpp(897) Loading plugin: CheckSystem...

d NSClient++.cpp(897) Loading plugin: NSClient server...

d \PDHCollector.cpp(66) Autodetected w2k or later, using w2k PDH counters.

l NSClient++.cpp(600) NSCLient++ - 0.3.7.7 2009-07-05 Started!

d \PDHCollector.cpp(103) Using index to retrive counternames

d \Socket.h(675) Bound to: 0.0.0.0:12489

l NSClient++.cpp(402) Using settings from: INI-file

l NSClient++.cpp(403) Enter command to inject or exit to terminate...

d \PDHCollector.cpp(123) Found countername: CPU: \Processor(\_total)\% processortid

d \PDHCollector.cpp(124) Found countername: UPTIME: \System\Tid sedan systemstart

d \PDHCollector.cpp(125) Found countername: MCL: \Minne\Dedikationsgrõns

d \PDHCollector.cpp(126) Found countername: MCB: \Minne\Dedicerade byte

Then when you run the check from Nagios again:

check\_nt -H 10.0.0.1 -p 12489 -v USEDDISKSPACE -d SHOWALL -l c

c:\ - total: 149.00 Gb - used: 12.93 Gb (9%) - free: 136.07 Gb (91%)

|'c:\ Used Space'=12.93Gb;0.00;0.00;0.00;149.00

If you check the log from NSCLient++ you should see (amongst other):

d \NSClientListener.cpp(146) Data: None&2&5

d \NSClientListener.cpp(171) Data: 5

d NSClient++.cpp(1034) Injecting: checkCPU: 5, nsclient

d NSClient++.cpp(1070) Injected Result: OK '0'

d NSClient++.cpp(1071) Injected Performance Result: ''

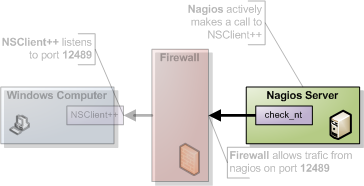
When you are don you can exit NSClient++ using the exit command:

exit

Then don't forget to start NSClient++ again:

net start nscp

## 4. Nagios configuration

[](http://nsclient.org/nscp/attachment/wiki/doc/usage/nagios/nagios-active-nsclient-002.png)

Nagios comes pre-configured for many of the NSClient checks. in windows.cfg you will find many entries along the lines of:

define service{

use generic-service

host\_name winserver

service\_description NSClient++ Version

check\_command check\_nt!CLIENTVERSION

}

The interesting part here is: **'check\_nt!CLIENTVERSION**' which will run a check against check\_nt. In commands.cfg the check\_nt command is defined like so:

# 'check\_nt' command definition

define command{

command\_name check\_nt

command\_line $USER1$/check\_nt -H $HOSTADDRESS$ -p 12489 -v $ARG1$ $ARG2$

}

So you can see most things are already setup for you so it is quite simple to get started. The more "advanced" checks (which takes parameters) looks like this if you recall the CPULOAD we tried from the command line:

define service{

use generic-service

host\_name winserver

service\_description CPU Load

check\_command check\_nt!CPULOAD!-l 5,80,90

}

the command is now defined as **'check\_nt!CPULOAD!-l 5,80,90**' which translates directly into:

<plugin dir>/check\_nt -H <ip of client> -p 12489 -v CPULOAD -l 5,80,90

which if you recall is exactly what we used when we tried the command from the command line. If you want to add a password the simplest way is to add it in command.cfg (if you want to have the same password on all your clients) like so:

# 'check\_nt' command definition

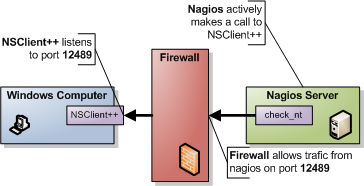
define command{

command\_name check\_nt

command\_line $USER1$/check\_nt -H $HOSTADDRESS$ -p 12489 -s <password> -v $ARG1$ $ARG2$

}

## 5. Final words

[](http://nsclient.org/nscp/attachment/wiki/doc/usage/nagios/nagios-active-nsclient.png)

As I stated initially using check\_nt is limited and many checks (for instance EventLog?) wont work this way. So a good idea is probably to start checking out the [NRPE Guide](http://nsclient.org/nscp/wiki/doc/usage/nagios/nrpe) as well.

**And remember** if you experience problems don't "debug" from nagios, run your command from the command line while having nsclient++ running in /test mode and you should be fine!

[Last modified](http://nsclient.org/nscp/wiki/doc/usage/nagios/nsclient?action=diff&version=16) [21 months ago](http://nsclient.org/nscp/timeline?from=2013-03-20T21%3A33%3A55%2B01%3A00&precision=second) Last modified on 03/20/13 21:33:55