

About NSClient++ 0.4.x

NSClient++ has always attempted to be a generic agent built to support any ones needs. The original concept was to merge "nscient" with "nrpe_nt". Due to inherent design issue sthe case was always a bit cramped and much was based around the inner working of NRPE. This has in 0.4.0 changed **drastically** one could argue that finally the idea behind a generic agent has finally become reality something which can be seen in the number of supported protocols. At launch of 0.4.0 we will support the following protocols with many more to come.

- Distributed NSCP
- Graphite
- NRPE
- NSCA
- NSClient (check_nt)
- NSCP
- SMTP
- Syslog

Message Exchange Concepts (paradigms)

To create a generic client it was important to find a few paradigms which are generic and extensible enough to facilitate various uses. In NSClient++ we currently support three main paradigms:

- query (status check) A **Synchronous query** MEP used to query system health (or other things)
- submissions (status report) A **fire and forget** MEP used to report status (or other things)
- execution (script execution) A **Synchronous? execution** MEP used to modify and change system aspects (or other things)

The idea is that these are more message exchange patterns and can be used for many things. For instance query could just as well be used for inventory and execution could easily be used to modify configuration and/or system properties.

Query

This is the common idea used in NRPE the main difference is that all *limitations* have been removed. It is now possible to send any length payload, any number of queries in one go as well as attach payload data etc etc.

Submission

This is the common idea used in NSCA which the main difference is that all *limitations* have been removed. It is now possible to send any length payload, any number of queries in one go as well as attach payload data etc etc.

Execution

This is very similar to query and the main difference is how it is used.

Summary

It is important to understand that these are **not protocols** they are paradigms. For instance internally there is no payload limit but if you check from Nagios using `check_nrpe` you are still stuck with the 1024 char limitation. So currently the "wonders of no limits" are not really useful to you (but this will change). But the main idea is to show the community that it is possible and hopefully as time goes on new protocols will emerge. It is also important to understand that a lot of these things can still be used with great benefits even now as you can use all this internally inside scripts and what not. So dont think that it is unimportant just because NRPE still has a payload limit.

Configuration Concepts

(Again) Since the goal was to make something which is flexible enough for everyone the settings subsystem allows you to do whatever you want (almost). We currently support five main different settings stores.

- old (legacy format used in 0.3.x versions)
- ini New version of the ini file wich supports all new options
- registry Windows registry (proper one this time) support.
- http (Currently very crude way to support loading remote configuration)
- dummy In-memory store which cannot be saved. (mainly used for scripts and such which will create a temporary run which does not require saving)

The "cool" thing about the whole concept is that on store can include another this means that if I want to keep my legacy "old" store I can simply create a include section which includes a ini file and thus have another file where I can place all new configuration settings. Or if I want to have the bulk of the config in the registry but allow an administrator the ability to override some keys locally we can (from the registry) include a file and vice verse.

The actual settings are key value pairs store in a hierarchical tree (much like the registry in that regards logical grouping). But of course we can still store this in a file thus the end result is:

```
[/settings/NSCA/client/targets/default]
host=192.168.0.1
encryption=aes
```

Where you can notice the `/settings/NSCA/client/targets/default` which is the hierarchical path **host** is the key and **192.168.0.1** is the value.

For more details about configuration see the configuration section [doc/configuration/0.4.x](#).

TODO This page is a quick place holder for a more in depth version which I will write at some point :)