Nagios XI – Best Practices

The Industry Standard in IT Infrastructure Monitoring

Purpose

This document covers various “best practices” which can be applied to your Nagios XI server. It accompanies the presentation from the Nagios World Conference 2015 by Troy Lea.

Target Audience

This document is intended for use by Nagios XI Administrators as a guide to understand how to get the most out of Nagios XI. It covers a range of topics such as how to get the most out of XI, things you wish you knew, and configuration practices.

Summary

Strictly speaking, a best practice is a flexible statement. Depending on your environment and your needs some topics won’t apply and other topics may be exactly what you are after. The information in this talk is a reflection of Nagios XI deployments in the wild.

Additionally, monitoring is not all about metrics and thresholds, it can also be helpful to ensure standards are enforced. If a setting gets changed you can be notified about it instead of having to track it down during one of your troubleshooting adventures.

This document is split into two chapters:

1. Server Internals
    - License entitlements
    - Motoring the Nagios XI server
    - Performance tuning
    - Backups
2. Configuration
    - Check Intervals
    - Dependencies
    - Templates
    - Macros
    - MRTG

Server Internals

Nagios XI License Entitlements

Each XI license entitles 3 instances:

- Production
- Test & Dev (T&D)
- Disaster Recovery (DR)

License activation is tied to the IP Address of each XI host. The caveat to this is that only your Production system can be used for actual monitoring. We are flexible and understand your needs to actively have a test system that runs alongside production. This allows you to implement new checks into production with the confidence that they will work.

What's Monitoring Nagios XI?

- How would you know your XI server died?
  - "Nagios XI Server" Monitoring Wizard
- DR instance monitors production instance
  - Production instance is UP & HEALTHY
- Production Instance monitors DR instance
  - DR instance is UP & HEALTHY

Probably the most important part of a monitoring system is knowing that it's actually working. Nagios XI comes with a "Nagios XI Server" monitoring wizard to ensure everything is A-OK.

However there is no point in monitoring itself, if it's down you're not going to hear about it.

Utilize your DR instance to monitor the production instance. This way, if the production instance goes down, you'll receive alerts about it.

The same applies for monitoring the DR instance, make sure production monitors the DR instance to make sure it's healthy.

By using the "Nagios XI Server" wizard to monitor the other instance, you can have confidence knowing that when something goes wrong, you'll really hear about it.

Montoring the Nagios XI "localhost"

Do you know how your XI server is performing?

- Services you should ideally be monitoring
  - crond, httpd, mysql, ndo2db, npcd, ntpd, postgresql
    - postgresql is not used on fresh installations of XI 5.x onwards. If you upgraded to XI 5 from a previous version postgres is used.
  - snmptrapd and snmptt are not present until you follow the XI SNMP Trap procedure
    - `/usr/local/nagiosxi/scripts/manage_services.sh` is the plugin included with Nagios XI (5)
- File counts - check these folders to make sure the temp files are being processed
  - NPCD Perfdata spool directory

        - `/usr/local/nagios/var/spool/perfdata/`
    - xidpe spool directory
        - `/usr/local/nagios/var/spool/xidpe/`
    - Check results folder
        - `/usr/local/nagios/var/spool/checkresults/`
    - snmptt spool folder
        - `/var/spool/snmptt/`
    - **NOTE:** These locations can vary if a RAMDisk has been implemented
    - **`check_file_count`** is the plugin you can use
        - https://exchange.nagios.org/directory/Plugins/System-Metrics/File-System/check_file_count/details
- Has the nagios user account expired?
    - In some customer installations, it's possible that the nagios user account expires. This isn't always that obvious to troubleshoot, so checking that it hasn't expired is a good precautionary measure.
    - **`check_pass_expire.pl`** is the plugin you can use
        - https://exchange.nagios.org/directory/Plugins/Operating-Systems/Linux/check_pass_expire/details
        - File permissions change:
            - `setfacl -m u:nagios:r-- /etc/shadow`
        - Plugin also needs line **23** changed
        - From:
            - `use lib ".";`
        - To:
            - `use lib "/usr/local/nagios/libexec";`
- root mailbox size
    - If you're not a Linux person then you probably don't know about the system mailbox. This is a local mail system on the linux server where messages are sometimes sent.
    - Certain components used in Nagios XI such as MRTG will send messages to this mailbox when it has a problem. An incorrect MRTG configuration can cause a message to be sent every five minutes as this is when MRTG runs. That's about 288 messages a day. Over time the root mailbox can grow to GB in size causing issues. I wrote a plugin which can monitor this and let you know when it gets too big.
    - **`box293_check_mbox.pl`** is the plugin you can use
        - https://exchange.nagios.org/directory/Plugins/Email-and-Groupware/box293_check_mbox/details
        - Requires file permissions:
            - `setfacl -m u:nagios:r-- /var/spool/mail/root`
- MySQL / MariaDB Databases
    - If the tables are crashed and go undetected, this can have a severe impact on the system and you may not be storing important data and it may cause strange problems.
    - **`box293_check_mysql_table_status`** is the plugin you can use to check this:
        - https://exchange.nagios.org/directory/Plugins/Databases/MySQL/box293_check_mysql_table_status/details
    - Also, another problem can occur if the database engine runs on a different timezone to the local system.

- **box293_check_mysql_date** is the plugin you can use to check this:
  - https://exchange.nagios.org/directory/Plugins/Databases/MySQL/box293_check_mysql_date/details
- Overall Load
  - This service is included by default in Nagios XI
  - **check_load** is the plugin included with Nagios XI (5)
- Memory Free – Physical
  - Make sure your XI server doesn't run out of memory
  - **check_memory.sh** is the plugin you can use
    - https://exchange.nagios.org/directory/Plugins/Operating-Systems/Linux/check_memory-2Esh/details
- Swap Usage
  - If the system runs out of physical memory and starts swapping to disk, the system performance will be greatly impacted.
  - This service is included by default in Nagios XI
  - **check_swap** is the plugin included with Nagios XI (5)
- Disk Free
  - Disk free space is very important.
  - This service is included by default in Nagios XI however it only monitors /
  - If you have different volumes mounted then you should be monitoring each one of these.
  - **check_disk** is the plugin included with Nagios XI (5)

Date and Timezone

Make sure your XI server has its timezone correctly defined

- Configure Timezone
  - Admin > Manage System Config

When using NTP make sure it's synchronized with a trusted time source like `pool.ntp.org`.

If this is a virtual machine, don't sync its time with the hypervisor (ESXi, HyperV etc.)!

- Can be the source of confusing problems such as:
  - Apply configuration throwing errors
  - Performance data not being processed

CPU

Having the right amount of CPU cores is important but so too is the speed of those cores. Not all plugins and processes are multi-threaded, so a higher speed CPU is going to benefit. A 3.4GHz CPU will do a lot more than a 2.2GHz one.

Refer to the XI Hardware Requirements guide:

- https://assets.nagios.com/downloads/nagiosxi/docs/Nagios-XI-Hardware-Requirements.pdf

Memory

How much memory do you need on an XI system? When all the hosts and services in XI are healthy, the amount of memory used is far less compared to a major system outage. When XI fires off event handlers they consume memory, if there is a major outage and a lot of event handlers are being executed, a lot of memory is being consumed. It doesn't take long for 6GB of memory to be used.

Generally speaking you should have at least 50% more memory than needed.

Refer to the XI Hardware Requirements guide:

- https://assets.nagios.com/downloads/nagiosxi/docs/Nagios-XI-Hardware-Requirements.pdf

RAM Disk

Configuring Nagios XI with a RAM Disk is highly recommended as the number of monitored objects increase. The more things you are monitoring the more disk I/O occurs. By directing this traffic a RAM Disk, the time it takes for that I/O operation to complete is drastically faster.

- Reduces disk I/O & load
- Speeds up processing of performance data
- Speeds up processing of spooled check results
- Speeds up nagios restarts

Refer to the XI RAM Disk procedure:

- https://assets.nagios.com/downloads/nagiosxi/docs/Utilizing_A_RAM_Disk_In_NagiosXI.pdf

Solid State Disk (SSD)

Greatly improves overall performance

- Compliments RAM Disk
- Helps read/writes with:
  - Logs
  - Database
  - Performance Graphs
  - Reports

SSD vs RAID ?

RAID allows for much larger disk capacities than SSD can provide, however it would be very hard for a spinning disk RAID set to beat the performance of SSD.

Keep in mind if you implement SSD you should implement RAID1 sets for redundancy purposes.

rrdcached

rrdcached is a way of accumulating the received performance data and then processing it in a batch job. It helps with larger installations and can reduce I/O, however it can also result with performance graphs lagging behind the realtime results.

Refer to the XI rrdcached procedure:

- https://assets.nagios.com/downloads/nagiosxi/docs/Using_rrdcached_with_Nagios_XI.pdf

Offloaded MySQL / MariaDB

On larger installations there can be a lot more data being written to the databases, which in turn can result in a lot of CPU usage directed away from actual monitoring.

Offloading to a separate server will remove this CPU usage from your monitoring server.

Of course, make sure you monitor the offloaded server!

- Disk/CPU/Memory/Tables/Service
  - Refer to earlier notes about services

Refer to the XI Offloaded DB procedure:

- https://assets.nagios.com/downloads/nagiosxi/docs/Offloading_MySQL_to_Remote%20Server.pdf

Mod-Gearman

Mod-Gearman is a way of offloading the plugin execution to separate workers instead of the monitoring engine doing it. A worker can be on the XI server itself OR on external hosts.

On external hosts, it requires all the plugins to be installed that are going to be executed.

Also, be aware of plugins that create temporary files, these don't work well if the plugins are moving about the workers.

You can also use host groups for directing checks to only be executed by specific workers, this is handy for multi-site setups.

- Defined on the XI server in:
  - `/etc/mod_gearman/`**`mod_gearman_neb.cfg`**
  - By using the `hostgroups=` and `servicegroups=` directives
- The workers need to be configured to use these groups:
  - `/etc/mod_gearman/mod_gearman_worker.conf`
  - By using the `hostgroups=` and `servicegroups=` directives

Mod Gearman by default puts all hosts and service checks into the queues called `host` and `service`

- You can stop workers from processing checks in these queues by using these options:
  - `/etc/mod_gearman/mod_gearman_worker.conf`
  - By using the `hosts=no` and `services=no` directives
  - This is required when you have workers at dedicated sites

You will need to exclude the Nagios XI localhost checks from being executed by the gearman workers by defining that host/service objects into host/service groups

- Defined on the XI server in:
  - `/etc/mod_gearman/`**`mod_gearman_neb.cfg`**
  - By using the `localhostgroups=` and `localservicegroups=` directives

XI 2014 uses Core 4.x which now has its own workers. Using Mod-Gearman on an XI 2014 server just for the purpose of a local worker is not required, however if you need external workers then Mod-Gearman is the solution for you.

Refer to the XI Mod-Gearman procedure:

- https://assets.nagios.com/downloads/nagiosxi/docs/Integrating_Mod_Gearman_with_Nagios_XI.pdf

Disaster Recovery

Define is what is import to you in a disaster. Once you have clearly defined goals and outcomes you can plan appropriately and test.

These presentations cover DR options:

- Andy Brist: High Availability and Failover Solutions for Nagios XI
  - https://www.youtube.com/watch?v=KW5Qkl8brcA
- Jeremy Rust & Devin Vance: Scaling Across Data Centers Using High Availability
  - https://www.youtube.com/watch?v=EVMbTbh9zV4

Backups

Have you scheduled your backups in Nagios XI?

- Admin > System Backups
- Schedule backups of XI
  - Location can be local, FTP, SSH
  - **Remote** location recommended. Storing them on storage that is not local to the XI file system is important - make sure you can get to your backups if your XI server dies.
- Manual Backups
  - Local Backup Archives via Admin menu
  - `/usr/local/nagiosxi/scripts/backup_xi.sh`

Restoring Backups

The backup and restore procedure is very straight-forward and allows for a full recovery of your Nagios XI system.

Another good use of it is to migrate XI from one server to another.

Refer to the XI Backup and Restore procedure:

- https://assets.nagios.com/downloads/nagiosxi/docs/Backing-Up-And-Restoring-Nagios-XI.pdf

Configuration

Intervals - Host vs Service

- Host down HARD = service notifications suppressed
  - When a host goes down HARD, it will prevent service notifications from being sent, saving unnecessary alerts.
- What happens when host and services use the same check intervals?
  - Unnecessary Notifications get sent
  - A common mistake when setting up your monitoring intervals is to leave the host intervals the same as the service intervals.
  - What this can lead to is the hosts service's going into a HARD critical state before the HOST does.
- Make host go down HARD quicker than it's services!
  - By making sure the HOST goes into a HARD down state before services ensures the service notifications will be suppressed.

This is easier to explain in a scenario with examples:

```
define host{
       use windows-server
       host_name host1
       alias host1
       address 10.25.14.51
       check_interval 5
       retry_interval 1
       max_check_attempts 5
       notification_interval 30
       }

define service{
       use local-service
       host_name host1
       service_description Memory Usage
       check_command check_nrpe!CheckMem!ShowAll type=physical MinWarn=512M MinCrit=256M
       check_interval 5
       retry_interval 1
       max_check_attempts 5
       notification_interval 30
       }
```

Here is a scenario based on this configuration:

- 13:10:10
  - Nagios HOST check for host1 executed
  - Result = OK
  - HARD state
  - Attempt 1/5
  - Next scheduled check 13:15:10
- 13:10:30
  - host1 dies
  - Nagios does not know about this yet
- 13:10:50

- Nagios SERVICE check Memory Usage for host1 executed
- Check timed out after 30 seconds as host is down
- Result = CRITICAL
- SOFT state
- Attempt 1/5
- No notification sent
- Next scheduled check 13:12:20

- 13:12:20
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 2/5
  - No notification sent
  - Next scheduled check 13:13:50
- 13:13:50
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 3/5
  - No notification sent
  - Next scheduled check 13:15:20
- 13:15:10
  - Nagios HOST check for host1 executed
  - Check timed out as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 1/5
  - No notification sent
  - Next scheduled check 13:16:10
- 13:15:20
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 4/5
  - No notification sent
  - Next scheduled check 13:16:50
- 13:16:10
  - Nagios HOST check for host1 executed
  - Check timed out as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 4/5
  - No notification sent
  - Next scheduled check 13:17:10
- 13:16:50
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - HARD state
  - Attempt 5/5
  - Notification SENT
  - Next scheduled check 13:22:20
  - Next notification 13:47:20

- 13:17:10
  - Nagios HOST check for host1 executed
  - Check timed out as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 3/5
  - No notification sent
  - Next scheduled check 13:18:10
- 13:18:10
  - Nagios HOST check for host1 executed
  - Check timed out as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 4/5
  - No notification sent
  - Next scheduled check 13:19:10
- 13:19:10
  - Nagios HOST check for host1 executed
  - Check timed out as host is down
  - Result = CRITICAL
  - HARD state
  - Attempt 5/5
  - Notification SENT
  - SERVICE notifications for host1 suppressed as host1 is in a HARD down state
  - Next scheduled check 13:24:10
  - Next notification 13:49:10
- 13:22:20
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - HARD state
  - Attempt 5/5
  - NO Notification sent as host1 is in a HARD down state
  - Next scheduled check 13:27:50

What you can see here is that Nagios did not know that they host was down until about 4 minutes after the host died. In that 4 minutes, the Memory Usage service went into a hard state and triggered the retry_interval to be used for that service, effectively beating the host check to reaching that HARD state and a notification was send.

Now let's look at a revised example with different check settings for host and service:

```
define host{
       use windows-server
       host_name host1
       alias host1
       address 10.25.14.51
       check_interval 2
       retry_interval 1
       max_check_attempts 2
       notification_interval 30
       }

define service{
       use local-service
       host_name host1
       service_description Memory Usage
       check_command check_nrpe!CheckMem!ShowAll type=physical MinWarn=512M MinCrit=256M
```

```
check_interval 5
retry_interval 1
max_check_attempts 5
notification_interval 30
}
```

Here is a scenario based on this configuration:

- 13:10:10
  - Nagios HOST check for host1 executed
  - Result = OK
  - HARD state
  - Attempt 1/2
  - Next scheduled check 13:12:10
- 13:10:30
  - host1 dies
  - Nagios does not know about this yet
- 13:10:50
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 1/5
  - No notification sent
  - Next scheduled check 13:12:20
- 13:12:10
  - Nagios HOST check for host1 executed
  - Check timed out as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 1/2
  - No notification sent
  - Next scheduled check 13:13:10
- 13:12:20
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 2/5
  - No notification sent
  - Next scheduled check 13:13:50
- 13:13:10
  - Nagios HOST check for host1 executed
  - Check timed out as host is down
  - Result = CRITICAL
  - HARD state
  - Attempt 2/2
  - Notification SENT
  - SERVICE notifications for host1 suppressed as host1 is in a HARD down state
  - Next scheduled check 13:15:10
  - Next notification 13:43:10
- 13:13:50
  - Nagios SERVICE check Memory Usage for host1 executed
  - Check timed out after 30 seconds as host is down
  - Result = CRITICAL
  - SOFT state
  - Attempt 3/5
  - No notification sent

    - Next scheduled check 13:15:20
- 13:15:10
    - Nagios HOST check for host1 executed
    - Check timed out as host is down
    - Result = CRITICAL
    - HARD state
    - Attempt 2/2
    - No Notification send as the Next notification is at 13:43:10
    - Next scheduled check 13:17:10
- 13:15:20
    - Nagios SERVICE check Memory Usage for host1 executed
    - Check timed out after 30 seconds as host is down
    - Result = CRITICAL
    - SOFT state
    - Attempt 4/5
    - No notification sent
    - Next scheduled check 13:16:50
- 13:16:50
    - Nagios SERVICE check Memory Usage for host1 executed
    - Check timed out after 30 seconds as host is down
    - Result = CRITICAL
    - HARD state
    - Attempt 5/5
    - NO Notification sent as host1 is in a HARD down state
    - Next scheduled check 13:22:20

In this example you can see Nagios identified that host1 was down (HARD non-OK state) BEFORE the Memory Usage service reached a HARD non-OK state, because of this the service notifications were suppressed.

Service Dependencies

When a host goes down, the services still get executed and can result in services in an unknown or critical state. Nagios suppresses any notifications however they still appear as critical in the interface.

Sometimes a host can be up but the monitoring agent can be down. An example of this is an NRPE agent.

By using service dependencies, if the master service goes down, you prevent notifications be sent OR prevent checks from being executed. Either option simply pushes the next check or notification to the next interval.

This is very similar to the previous section on intervals, make sure your master service goes down HARD before the dependent services, use different check intervals or retries.

Master service e.g. - Ping or NRPE Version

Disable Service Checks

A new feature of Nagios Core 4.1.0 (included in XI 5 onwards) is a configuration directive called host_down_disable_service_checks.

- It's best described as automatic service dependencies on their own hosts.
- System wide setting, this applies across the board, it is not granular.
- Define in `/usr/local/nagios/etc/`**`nagios.cfg`**
  - `host_down_disable_service_checks=1`
  - Can also be defined via CCM > Advanced > Nagios Core Main Config
  - Restart nagios monitoring engine or Apply Configuration for setting to take effect
- Can reduce the load on the XI host as plugins will not be executed if this host is down.
- Keep in mind that if the host is down then any defined service dependencies will be ignored.

Check Intervals - Be Realistic

It can be very easy to setup your monitoring with the same intervals across the board. This can lead to peaks and troughs in load on the XI server as a lot of checks can occur in the same time windows.

Have a think about what you are monitoring and how often do you really need to check it. Something like disk usage rarely runs out quickly, you can monitor this every hour and be confident you'll be notified about the free disk space running low in a reasonable time.

- Does it need to be checked every 5 minutes?
- Disk Free Space – every 60 minutes perhaps?
- Too long = no performance data
  - An interval that is more than four hours apart

However if you are going to make it every hour, why not every 58 minutes or 61 minutes? Try to spread the load out a bit.

- Different intervals to spread the load
- 3, 5, 7 minute intervals
- 58, 60, 62 minute intervals

Notification & Check Intervals

Sometimes larger check intervals can have an adverse affect on notification intervals.

The monitoring engine determines if it should send a notification every time a check result is received.

Due to how the internal scheduling works, you might fall short of the notification window by a small time period like 20 seconds. This means it might be another 15 minutes until the next check is run, that's when the notification will be sent.

- e.g. 15 minute check and 60 minute notification
- Internal scheduling may cause 14min 55sec to pass, 4 x 14:55 = 59min 40sec … it's < 60min!
- Notification not sent until 75min!
- Scheduling is geared +/- to reduce load!

Use Hostgroups!

Using hostgroups in your service definitions is one of the most powerful features of Nagios.

Common services generally have the same threshold for all hosts. Instead creating individual services for each host you monitor, a service can be assigned to multiple hosts using a hostgroup. What this means is that you only need to have the service defined once, and when you want to tweak the thresholds, you only need to change it in one location and all hosts will receive the updated thresholds.

So if you have a host group called windows_servers, whenever you add a new windows server it's just a matter of adding that server to the hostgroup and that host gets that bunch of common checks. This is great for consistent monitoring, it ensures standards get applied and reduces management overhead

Use Contactgroups!

One of the most common support questions we get asked is how to add or remove a contact to or from a bunch of objects? If you don't have the enterprise edition license then you don't get access to the bulk modification tool that allows you to do this.

However just changing just the contacts can lead to human error. It's very easy to make mistakes and before you know it a notification was not sent to the correct people.

Using a contact group is a much better method. It's so much easier to go in and add or remove a contact from a contactgroup and instantly all the objects that use this group will be updated. Even if there is only one member to a contact group it still makes administration so much easier.

If you've not activated the trial of enterprise edition, this is a great way make use of the bulk modification tool to implement contactgroups and remove the individual contacts. Once your standard is in place, administration will be so much easier.

Configuration Wizards

Configuration wizards are a great ice breaker for people who are new to nagios, you didn't need to learn how a plugin worked or how to create a command definition followed by service definitions, you just pointed and clicked.

The downside to wizards is that they create a lot of services. In a large scale monitoring environment, you might use services that are applied to hostgroups which reduces administrative overhead. Using wizards doesn't really apply to these environments however they are a great primer for setting up initial services, from there you can modify them in CCM.

- Pros
  - Great for getting up and running quickly
  - No need to learn how a plugin works
- Cons
  - Creates individual services
  - More work later when enforcing "standards"

Templates

Templates are very powerful when used for the right purposes.

A really good example is how the XI Configuration Wizards use templates for the host objects. The host object template has a standard ICMP up/down check. This means if you ever wanted to change the thresholds, you could change the template and then all hosts using that template will get the updated check.

You can use multiple templates in a layering fashion. As Nagios core reads the object definition, it looks at the first template and obtains the settings. It then looks at the next template and layers those settings over the top of the previous settings. This continues and builds the final object. In the XI GUI, with multiple templates, the template at the bottom of the list will be overwritten by a template above it. Object directives can be set to inherit that setting from a template, or ignore it. Other settings can be additive, like hosts, hostgroups, contacts, contactgroups.

For example you might have a master template that defines the base settings all services should use. However you have a bunch of service checks that require a specific time period. Create a separate template that uses this time period and put that template at the top of the chain. The final service object that is created will use the specific time period.

You can even create an empty template that uses a combination of other templates, this way you can use the master templates across all your objects and easily add / remove other templates to the master template, in turn reducing your administrative overhead. Be careful not to add more administrative overhead though.

User Macros – resources.cfg

User macros are a way of storing and referencing common items such as usernames and passwords. Because you are referencing the objects as a macro, the actual value is not visible in the object definitions.

It also allows special characters to be used like an exclamation mark. Normally when an exclamation mark is used in a command_name directive, it's purpose is to split up the different arguments, so by storing it in a user macro it works around the problem.

- `$USER1$ = /usr/local/nagios/libexec`

Custom Object Variables

Custom object variables are one of the lesser known features of Nagios.

It allows you to define you own variables to use in your object definitions, this makes Nagios very flexible and very powerful.

A good example is if each windows host had it's own custom check_nt password. What you can do is store that password in the host object and then from your service objects you can reference the password. It also means that you can still have just one command that can be used my many hosts, reducing administrative overhead.

- E.G. hosts have their own check_nt password
- Define `_CHECK_NT_PASSWORD` in host object
  - Misc Settings tab > **Manage Variable Definitions** button
- In command definitions reference it as:
  - `$_HOSTCHECK_NT_PASSWORD$`

MRTG Clean Configs

Your MRTG configs may be collecting more than what you think.

In Nagios XI, the Network Switch / Router wizard uses MRTG to collect the monitoring data from the network device.

The configuration files for MRTG are created with the program cfgmaker. While you may have selected to only monitor a handful of ports from your network device, MRTG will collect data from all the interfaces. This creates extra network traffic and I/O.

- `/etc/mrtg/conf.d/*.cfg` files

You can edit these MRTG config files and comment out the ports for which you do not need data to be collected for. Each port consists of about 37 lines in total.

Also you'll find non-interfaces like VLANs, these can also be commented out, unless of course you want to monitor them.

Conclusion

This document should have helped you understand how a Nagios XI server works and different methods of configuring it for your environment so you can get the most out of out.

Final Thoughts

For any support related questions please visit the Nagios Support Forums at:

http://support.nagios.com/forum/