Automated Host Management In Nagios XI

Nagios®

The Industry Standard in IT Infrastructure Monitoring

Purpose

This document describes how to automate adding and removing hosts and services in Nagios XI from the command line.

Target Audience

This document is intended for use by Nagios XI Administrators and Developers and is recommended for those who have prior knowledge of scripting and automating solutions.

Summary

This document will cover the following topics.

- Automation Overview Including REST API (new in XI 5.x)
- Key Files and Directory Locations
- Importing a Configuration File
- Import Verification and Exit Codes
- Removing Hosts and Services

Automation Overview

Some administrators may have need to automate the process of adding and removing hosts and services in Nagios XI for use with cloud computing or large environments where solutions like *Puppet* or *Chef* may already be implemented. The procedures below outline how administrators can create their own automation solutions to safely add and remove hosts and services in Nagios XI while still maintaining the integrity of the monitoring environment.

Before Nagios XI 5 was released, the method for automatically adding hosts and services was performed by adding Nagios object definition files into the import directory and then running a script. This requires direct access to the Nagios XI host, perhaps via a ssh session. Security is provided by requiring access to the XI server console, permission via the Linux Operating system local users.

Nagios XI 5 introduced a REST API that allows you to create host and objects using a HTTP post (commonly done with a CURL command). This method can be issued from an external machine, you don't need to be on the XI server to create objects. Security is provided by an API key which is linked to a specific Nagios XI user.

Both methods are supported automation methods and will be covered in this documentation.

Key Files and Directory Locations

The directory locations and key files in Nagios XI are as follows.

`/usr/local/nagios/etc/import/`

Any configuration files in this directory that end with **`.cfg`** will automatically be imported into Nagios XI every time **Apply Configuration** is run from within Nagios XI, or the **`reconfigure_nagios.sh`** script is run.

`/usr/local/nagiosxi/scripts/reconfigure_nagios.sh`

This script is the command that is executed when **Apply Configuration** is initiated from the web interface. This script will import files from the **import** directory, verify configuration, and restart Nagios if verification succeeds. If verification fails, Nagios XI will roll the configuration back to the last working checkpoint.

`/usr/local/nagiosxi/scripts/nagiosql_delete_host.php`

Command-line script for host deletion. This script must be run from the `/usr/local/nagiosxi/scripts/` directory.

`/usr/local/nagiosxi/scripts/nagiosql_delete_service.php`

Command-line script for service deletion. This script must be run from the `/usr/local/nagiosxi/scripts/` directory.

REST API

The Nagios XI REST API (herein refereed to as "API") is completely documented in the Nagios XI web interface. Navigate to **Help** > **REST API Docs**. With that in mind, a full explanation will not be covered in this documentation, with the exception of the API Key.

The API Key is what provides authentication to Nagios XI users to access the API. A user can find their API Key by clicking their name in the top right of the navigation menu. This will take them to their Account Information page which show them their API Key in a read-only field. **NOTE**: The API Key may be longer than the field, simply click in the field and press **CTRL** + **A** on your keyboard to select it all.

An example of a CURL command used to access the API is as follows:

```
curl -XGET "http://10.25.5.2/nagiosxi/api/v1/system/status?apikey=5goacg8s&pretty=1"
```

It will produce output like this:

```
{
    "instance_id": "1",
    "instance_name": "localhost",
    "status_update_time": "2017-01-31 16:37:46",
    "program_start_time": "2017-01-31 14:20:42",
    "program_run_time": "8225",
```

```
    "program_end_time": "0000-00-00 00:00:00",
    "is_currently_running": "1",
    "process_id": "2819",
    "daemon_mode": "1",
    "last_command_check": "1970-01-01 10:00:00",
    "last_log_rotation": "1970-01-01 10:00:00",
    "notifications_enabled": "1",
    "active_service_checks_enabled": "1",
    "passive_service_checks_enabled": "1",
    "active_host_checks_enabled": "1",
    "passive_host_checks_enabled": "1",
    "event_handlers_enabled": "1",
    "flap_detection_enabled": "1",
    "process_performance_data": "1",
    "obsess_over_hosts": "0",
    "obsess_over_services": "0",
    "modified_host_attributes": "0",
    "modified_service_attributes": "0",
    "global_host_event_handler": "xi_host_event_handler",
    "global_service_event_handler": "xi_service_event_handler"
}
```


Creating Host And Services By Importing A Configuration File

Typically automated configuration management will make heavy use of templates. Simpler configuration imports will be easier to maintain and have less potential issues upon import and deletion. When automating a configuration import, we recommend the following:

- Use one configuration file for a single host and each of it's services
- Service definitions should only be applied to a single host, not a host list or a hostgroup
- The `.cfg` file should be named according to the host name, example: **`LOC_MASShost_1.cfg`**

Example file called **`LOC_MASShost_1.cfg`**:

```
define host{
        host_name LOC_MASShost_1
        address 127.0.0.1
        use xiwizard_generic_host
}
```

```
define service{
        host_name LOC_MASShost_1
        service_description LOC_MASSservice_ping
        use xiwizard_website_ping_service
}

define service{
        host_name LOC_MASShost_1
        service_description LOC_MASSservice_dnsip
        use xiwizard_website_dnsip_service
        check_command  check_xi_service_dns!'-a 127.0.0.1'
}
```

Place the configuration file into the **/usr/local/nagios/etc/import** directory.

Run the **/usr/local/nagiosxi/scripts/reconfigure_nagios.sh** script.

When this script is run, Nagios XI looks in the import directory and pulls all of the configuration files to either update old versions, or populate new ones.

Creating Host And Services By Using API

Typically automated configuration management will make heavy use of templates. Simpler API commands will be easier to maintain and have less potential issues upon import and deletion. When use the API, we recommend the following:

- Service definitions should only be applied to a single host, not a host list or a hostgroup
- Special characters like `$` and `!` will need to be escaped to prevent the Linux shell from interpreting them, for example:
  - `\!`

API commands use the same directive names as a Nagios config file has, you provide them in the format `directive_name=value` and separate them using the ampersand (&). For example:

```
address=127.0.0.1&use=xiwizard_generic_host
```

Example commands to create host and services, just like the Config File definitions above:

Host Object:

```
curl -XPOST "http://10.25.5.2/nagiosxi/api/v1/config/host?apikey=5goacg8s&pretty=1" -d
"host_name=LOC_MASShost_1&address=127.0.0.1&use=xiwizard_generic_host&force=1&applyconfig=1
```

Service Ping:

```
curl -XPOST "http://10.25.5.2/nagiosxi/api/v1/config/service?apikey=5goacg8s&pretty=1" -d
"host_name=LOC_MASShost_1&service_description=LOC_MASSservice_ping&use=xiwizard_website_ping_
service&force=1&applyconfig=1"
```

Service DNS:

```
curl -XPOST "http://10.25.5.2/nagiosxi/api/v1/config/service?apikey=5goacg8s&pretty=1" -d
"host_name=LOC_MASShost_1&service_description=LOC_MASSservice_dnsip&use=xiwizard_website_dnsi
p_service&check_command=check_xi_service_dns\!'-a 127.0.0.1'&force=1&applyconfig=1"
```

Import Verification and Exit Codes

It is important to monitor the return status of the **`reconfigue_nagios.sh`** script. Nagios XI will not apply a broken configuration, so if the configuration verification fails, Nagios XI will restore the configuration files to the last working checkpoint, but the broken configuration will still remain in the database. Detecting failures is crucial, otherwise all future automated changes will likely fail after the first. Each exit code in `reconfigure_nagios.sh` corresponds to a different stage of the script, and will reveal where the problem occurred.

The following are the error codes that `reconfigure_nagios.sh` will return.

```
0       no problems detected
1       config verification failed
2       nagiosql login failed
3       nagiosql import failed
4       reset_config_perms failed
5       nagiosql_exportall.php failed (write configs failed)
6       /etc/init.d/nagios restart failed
7       db_connect failed
```

If you get a return code of `0`, your new configuration file has been successfully verified as a working configuration and has been accepted by Nagios XI.

API Verification and Exit Codes

The API itself will produce different output, usually an error message saying a value was missing. For example, the following command is executed:

```
curl -XPOST "http://10.25.5.2/nagiosxi/api/v1/config/service?apikey=5goacg8s&pretty=1" -d
"host_name=LOC_MASShost_1&service_description=LOC_MASSservice_ping&use=xiwizard_website_ping_
service&applyconfig=1"
```

The following output is produced:

```
{
    "error": "Missing required variables",
    "missing": [
        "max_check_attempts",
        "check_interval",
        "retry_interval",
        "check_period",
        "notification_interval",
        "notification_period",
        "contacts OR contact_groups"
    ]
}
```

The output above explains that certain directives are missing. You might notice the command used was very similar to the command executed earlier, except for one difference. The command did not have the `&force=1` argument, the command originally issued is using a template and hence those "missing" directives are actually in the template. Using the `&force=1` argument allows you to force the API to create the configuration file you submitted to it.

Additionally, the curl command does not receive an exit code, because the curl command itself received a valid response from the Nagios XI server. It's up to you to process the data received to ensure it's correct. By default the data received is a JSON object, so you can easily test to see if the "success" or "error" keys exist to ensure the object was created, as if it's executed by an automation system like *Puppet* or *Chef* the exit code will usually be `0`.

Removing Hosts and Services

To remove a host from Nagios XI, the host must be removed from the Core Config Manager's database, which will handle the deletion of the physical configuration file as well. **A host or service can only be removed after all dependent relationships have been removed**. This applies for either automation method used. For the example above, the services for this host must first be deleted.

Using Nagios XI Scripts

Service can be deleted either by **database ID** or **Config Name**, which corresponds to the physical configuration file that it is stored in. This is why it is necessary to name the import file the same as the host name.

First change into the directory:

```
cd /usr/local/nagiosxi/scripts
```

This command will delete all the services for the host LOC_MASShost_1 (created earlier):

```
./nagiosql_delete_service.php --config=LOC_MASShost_1
```

This will produce a lot of output, the last line of the command should be the most important:

```
2 services deleted successfully!
```

After the services have successfully been deleted, the host can be removed as well:

```
./nagiosql_delete_host.php --host=LOC_MASShost_1
```

Once the host is removed, the new configuration can be applied and verified by running the reconfigure_nagios.sh

```
./reconfigure_nagios.sh
```

Using REST API

The services need to be individually removed and are deleted by using the **Host Name** and the **Service Description**. These commands don't apply the config as there is no need to do this multiple times, when the host is deleted is fine.

This command will delete the LOC_MASSservce_ping services for the host LOC_MASShost_1 (created earlier):

```
curl -XDELETE "http://10.25.5.2/nagiosxi/api/v1/config/service?
apikey=5goacg8s&pretty=1&host_name=LOC_MASShost_1&service_description=LOC_MASSservice_ping"
```

This will produce the following output:

```
{
    "success": "Removed LOC_MASShost_1 :: LOC_MASSservice_ping from the system. Config
imported but not yet applied."
}
```

This command will delete the LOC_MASSservce_dnsip services for the host LOC_MASShost_1 (created earlier):

```
curl -XDELETE "http://10.25.5.2/nagiosxi/api/v1/config/service?
apikey=5goacg8s&pretty=1&host_name=LOC_MASShost_1&service_description=LOC_MASSservice_dnsip"
```

After the services have successfully been deleted, the host can be removed as well:

```
curl -XDELETE "http://10.25.5.2/nagiosxi/api/v1/config/host?
apikey=5goacg8s&pretty=1&host_name=LOC_MASShost_1&applyconfig=1"
```

The output will be as follows:

```
{
    "success": "Removed LOC_MASShost_1 from the system. Config applied, Nagios Core was
restarted."
}
```

Conclusion

This concludes the instructions for automated host and service management in Nagios XI. For any problems or further details in developing an automated procedure, contact our support team at:

https://support.nagios.com/forum

Other useful automation resources:

- Nagios Object Definitions:
  - https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/objectdefinitions.html
- Puppet:
  - http://puppetlabs.com/
- Chef:
  - https://www.chef.io/chef/