Cisco UCSM Plugin and Addon

For Nagios Core

User Guide

October 8, 2015

Table of Content

List of Tables

List of Figures

1. Overview

Data center administrators have been using Nagios for more than a decade now and it has emerged as one of the favorite open source tool for the Data Center monitoring.

Nagios is an open source computer system monitoring, network monitoring and infrastructure monitoring software application. Nagios offers monitoring and alerting services for servers, switches, applications, and services.

The solution provides end-user with two primary components. The first is the Nagios monitoring plugin script which will provide end-user with the capability of monitoring the UCS domains.

The second is an add-on to the Nagios, which will provide end-user with the capability to auto discover UCS domains.

1.1 Acronyms and Abbreviations

The following table describes the acronyms and abbreviations used in the document.

Abbreviation	Translation
DNS	Domain Name Server
FQDN	Fully Qualified Domain Name
UCS	Unified Computing System
FI	Fabric Interconnect
IOM	I/O Module
FEX	Fabric Extender
NAGIOS_HOME	Nagios install directory.
NAGIOS_ETC_DIR	Nagios etc directory path
NAGIOS_PLUGIN_DIR	Nagios plugin directory path
NAGIOS_LOGOS_DIR	Nagios logos image directory

Table 1 : Acronyms and Abbreviations

1.2 System Requirements

The Nagios must meet the below mentioned minimum requirements for this solution to work

- *Operating System – Nagios supported Linux server.*
  - http://www.nagios.org/about/propaganda/distros
- *Nagios Core*
  - http://www.nagios.org/download/core/thanks/?t=1398749242
- *Latest Nagios Plugins*
  - http://www.nagios.org/download/plugins/
- *Latest UCS Python SDK 0.8.3 or higher*
  - https://communities.cisco.com/docs/DOC-36899

2. Deploying the solution

- The solution is provided in the tar gzip format which can be easily extracted in any of the Nagios supported Linux distributions.
- The tar gzip file extracts to a folder named cisco_nagios and contains following three files
- `INSTALL` – This file guides end user on how to install the solution in an existing Nagios installation
- cisco-ucs-nagios-x.x.x`.tar.gz` – This tar gzip contains the Cisco UCS monitoring plugin and autodiscovery add-on tar gzip.
- `installer.py` – This is installer script which uses the cisco-ucs-nagios-x.x.x`.tar.gz` tar gzip and installs the solution as per the user environment.

2.1 Install paths

Installation requires that you know the paths to the following locations and they will depend on your Nagios installation as your environment. Please check with your Nagios administrator for more information.

Listed below are typical install locations and directories for different Linux distributions

- For Debian/SUSE
  - *The Nagios home directory*
    NAGIOS_HOME=/etc/nagios3
  - *Nagios etc directory that has nagios configuration files*
    NAGIOS_ETC_DIR=/etc/nagios3
  - *Nagios plugin directory that has all the Nagios plugin*
    NAGIOS_PLUGIN_DIR=/usr/lib/nagios/plugins
  - *Nagios logos directory*
    The logos directory is generally a part of CGI configuration file and the root path to the logos directory is denoted by 'physical_html_path'.

    Appending 'images/logos' to the value of the above variable provides us the logos directory path for Nagios.
    The cgi.cfg file can be found in NAGIOS_ETC_DIR

    NAGIOS_LOGOS_DIR=/usr/share/nagios3/htdocs/images/logos/

- In other Linux variants, typical paths can be

```
NAGIOS_HOME=/usr/local/nagios
NAGIOS_ETC_DIR=/usr/local/nagios/etc
NAGIOS_PLUGIN_DIR=/usr/local/nagios/libexec
NAGIOS_LOGOS_DIR=/usr/local/nagios/share/images/logos/
```

2.2 Install Nagios Monitoring Plugin

Following are the steps for installing the Nagios monitoring plugin.

a. *Extract the installation tar gzip file in a temporary location.*
```
# tar zxvf cisco-ucs-nagios-x.x.x.tar.gz
```

b. *Now run the installer, which should be present in the extracted folder.*
```
# ./installer.py
```

c. *Installer auto detects various install paths and prompt with default options for installing this plugin.*

d. *Installer also updates the configuration files which are required for the working of this plugin. It prompts and creates the backups of all the files which will be modified in this process*

e. *By default installer will install the monitoring plugin along with the auto discovery scripts. In case, only monitoring plugin is to be installed then use the '`--plugin`' option*

```
# ./installer.py --plugin
```

2.3 Auto Discovery Nagios Add-on

1. *In the autodiscovery directory, add/Update UCSHostInfo.csv with the UCS domain IP/FQDN and login credentials. User can also use the CLI parameters if a single domain or a range of IPs needs to be discovered.*

Below is the parameter mapping of CSV file to CLI parameters:

CSV Parameter	CLI Parameter
HostName	-H, --host
User	-u, --user
Password	-p, --password
Port	-n, --port
NoSSL(True/False)	--NoSsl
Proxy URL	--proxy

Table 2 : CSV to CLI parameter mapping

The servers that are defined in this CSV/CLI will be discovered and added to the Nagios for monitoring.

Example CSV:

```
<HostName>,<User>,<Password>,<Port>,<NoSSL(True/False)>,<Proxy URL>

10.65.183.10,admin,password,80,True
10.65.183.5,admin,password,80,True,http://proxy.ip.com:8080
10.65.183.5-10,admin,password
```

The HostName, User and Password fields are mandatory for the auto discovery to discover the UCS domain.

User can provide IP range in the hostname. Auto-Discovery script allows range definition by passing "–"in the fourth octet. For all IPs in that range, connection parameters will be same i.e. the username, password, port, SSL and proxy data if applicable.

As given in the above CSV entries, there is an entry "10.65.183.5-10".
This range will be expanded by Auto-discovery script as shown below:

```
10.65.183.5,admin,password
10.65.183.6,admin,password
...
...
10.65.183.10, admin, password
```

Note:

1. *In case user password contains any special character then it has to be provided in double quotes.*

   Example

   ```
   10.65.183.16,admin,"pass,word"
   ```

```
10.65.183.16,admin,"My_password"
```

2. *In case user password contains* "(double quotes) then it has to be escaped by another "(double quotes).
   Example
   If password is my"password then we will write the same in csv file as

   ```
   10.65.183.16,admin,"my""password"
   ```

3. In case user is a domain user then the user field should be defined as "ucs-<Domain>\<username>"
   Example

   ```
   xxx.xxx.xxx.xxx,"ucs-somedomain\user",Test12345
   ```

4. Giving IP range is allowed only for IPv4 addresses.

2. Now run the auto discovery script

- *If using CSV file:*

  #./NagiosAutoDiscoveryUCS.py

- *If using CLI parameters*:

  #./NagiosAutoDiscoveryUCS.py –H <Host Name> -u <user name> -p <password>

Note:

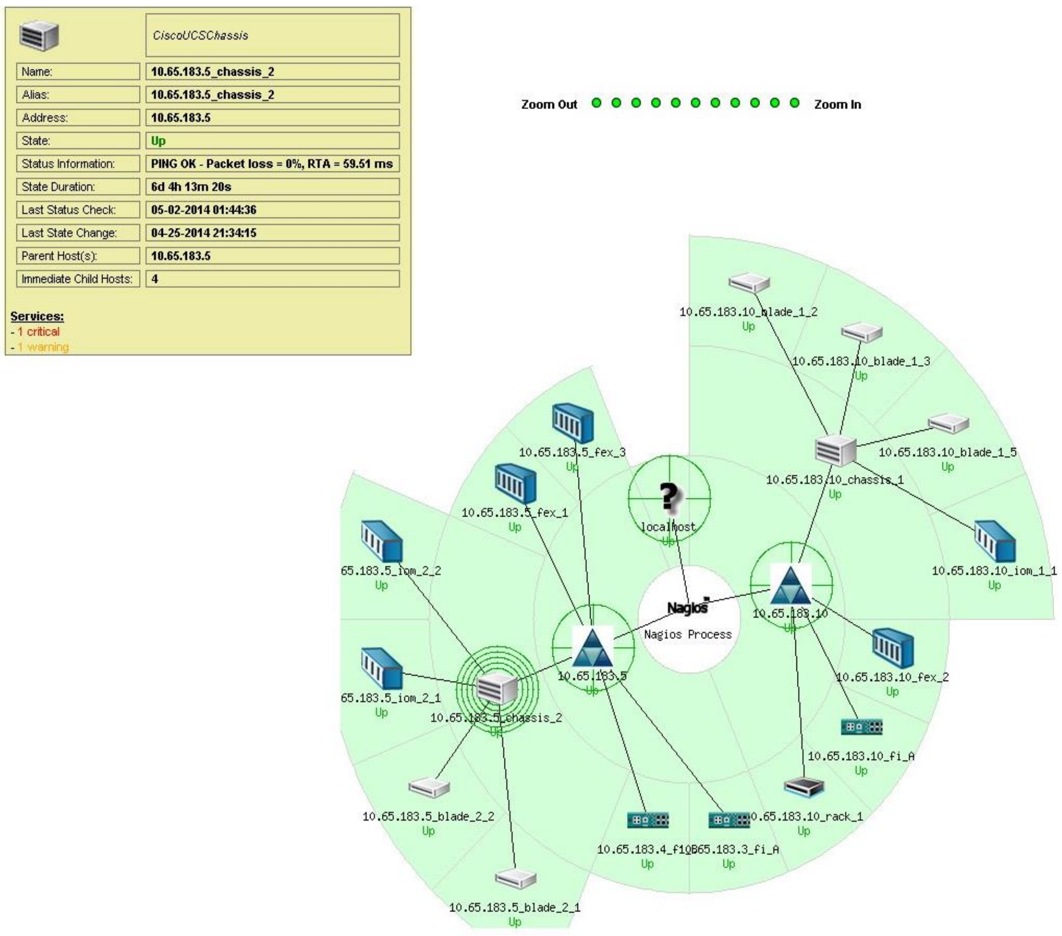
CLI parameters will be given preference over CSV file, i.e. if the Host Parameter via CLI is given then script will skip reading the CSV file.

3. Features

Once the installation is complete and auto discovery is executed, user can now see the components of the Cisco UCS domains which are discovered.

3.1 Maps View

The discovered UCS domain will be displayed in the Map section as shown below.



CiscoUCSChassis

Name:	10.65.183.5_chassis_2
Alias:	10.65.183.5_chassis_2
Address:	10.65.183.5
State:	Up
Status Information:	PING OK - Packet loss = 0%, RTA = 59.51 ms
State Duration:	6d 4h 13m 20s
Last Status Check:	05-02-2014 01:44:36
Last State Change:	04-25-2014 21:34:15
Parent Host(s):	10.65.183.5
Immediate Child Hosts:	4

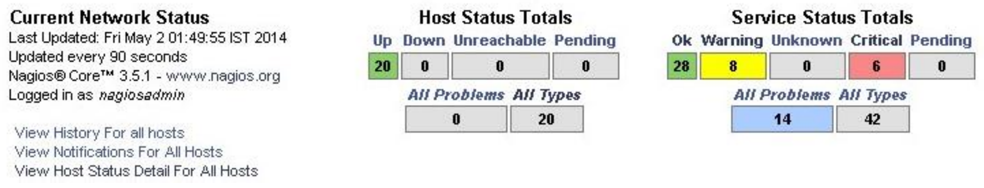
Services:
- 1 critical
- 1 warning

Figure 1 : UCS Domain Map in Nagios

3.2 Service View

The monitoring plugin for UCSM not only monitors the health status of the components in the UCS domain but also provides the relevant inventory information in case no fault has occurred on the given component.

The default monitoring service for the rack server checks for all the faults that may have occurred for the given server.



Current Network Status
Last Updated: Fri May 2 01:49:55 IST 2014
Updated every 90 seconds
Nagios® Core™ 3.5.1 - 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
20	0	0	0

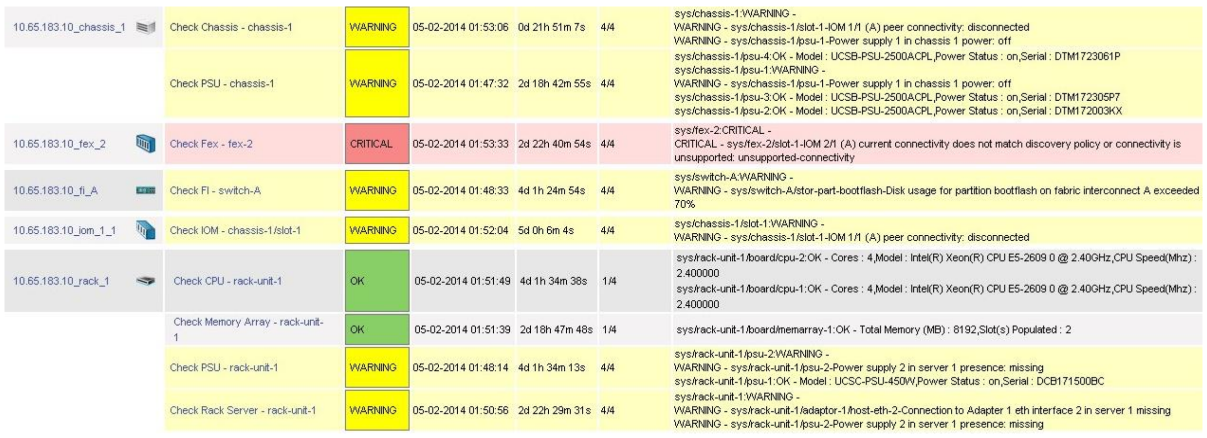
All Problems	*All Types*
0	20

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
28	8	0	6	0

All Problems	*All Types*
14	42

Figure 2 : UCS Service Overview in Nagios



10.65.183.10_chassis_1	Check Chassis - chassis-1	WARNING	05-02-2014 01:53:06	0d 21h 51m 7s	4/4	sys/chassis-1:WARNING - WARNING - sys/chassis-1/slot-1-IOM 1/1 (A) peer connectivity: disconnected WARNING - sys/chassis-1/psu-1-Power supply 1 in chassis 1 power: off
	Check PSU - chassis-1	WARNING	05-02-2014 01:47:32	2d 18h 42m 55s	4/4	sys/chassis-1/psu-4:OK - Model : UCSB-PSU-2500ACPL,Power Status : on,Serial : DTM1723061P sys/chassis-1/psu-1:WARNING - WARNING - sys/chassis-1/psu-1-Power supply 1 in chassis 1 power: off sys/chassis-1/psu-3:OK - Model : UCSB-PSU-2500ACPL,Power Status : on,Serial : DTM172305P7 sys/chassis-1/psu-2:OK - Model : UCSB-PSU-2500ACPL,Power Status : on,Serial : DTM172003KX
10.65.183.10_fex_2	Check Fex - fex-2	CRITICAL	05-02-2014 01:53:33	2d 22h 40m 54s	4/4	sys/fex-2:CRITICAL - CRITICAL - sys/fex-2/slot-1-IOM 2/1 (A) current connectivity does not match discovery policy or connectivity is unsupported: unsupported-connectivity
10.65.183.10_fi_A	Check FI - switch-A	WARNING	05-02-2014 01:48:33	4d 1h 24m 54s	4/4	sys/switch-A:WARNING - WARNING - sys/switch-A/stor-part-bootflash-Disk usage for partition bootflash on fabric interconnect A exceeded 70%
10.65.183.10_iom_1_1	Check IOM - chassis-1/slot-1	WARNING	05-02-2014 01:52:04	5d 0h 6m 4s	4/4	sys/chassis-1/slot-1:WARNING - WARNING - sys/chassis-1/slot-1-IOM 1/1 (A) peer connectivity: disconnected
10.65.183.10_rack_1	Check CPU - rack-unit-1	OK	05-02-2014 01:51:49	4d 1h 34m 38s	1/4	sys/rack-unit-1/board/cpu-2:OK - Cores : 4,Model : Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz,CPU Speed(Mhz) : 2.400000 sys/rack-unit-1/board/cpu-1:OK - Cores : 4,Model : Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz,CPU Speed(Mhz) : 2.400000
	Check Memory Array - rack-unit-1	OK	05-02-2014 01:51:39	2d 18h 47m 48s	1/4	sys/rack-unit-1/board/memarray-1:OK - Total Memory (MB) : 8192,Slot(s) Populated : 2
	Check PSU - rack-unit-1	WARNING	05-02-2014 01:48:14	4d 1h 34m 13s	4/4	sys/rack-unit-1/psu-2:WARNING - WARNING - sys/rack-unit-1/psu-2-Power supply 2 in server 1 presence: missing sys/rack-unit-1/psu-1:OK - Model : UCSC-PSU-450W,Power Status : on,Serial : DCB171500BC
	Check Rack Server - rack-unit-1	WARNING	05-02-2014 01:50:56	2d 22h 29m 31s	4/4	sys/rack-unit-1:WARNING - WARNING - sys/rack-unit-1/adaptor-1/host-eth-2-Connection to Adapter 1 eth interface 2 in server 1 missing WARNING - sys/rack-unit-1/psu-2-Power supply 2 in server 1 presence: missing

Figure 3 : UCS Service View in Nagios

Based on the UCSM fault information the plugin decides on the Nagios service state.

The following table shows the mapping of the severity levels of the Cisco UCSM faults to Nagios States.

UCSM Fault Severity	Nagios States
Critical and Major	CRITICAL
Minor and Warning	WARNING
Info and Cleared	OK

Table 3 : Cisco UCSM Fault Severity to Nagios State Mapping

Note:

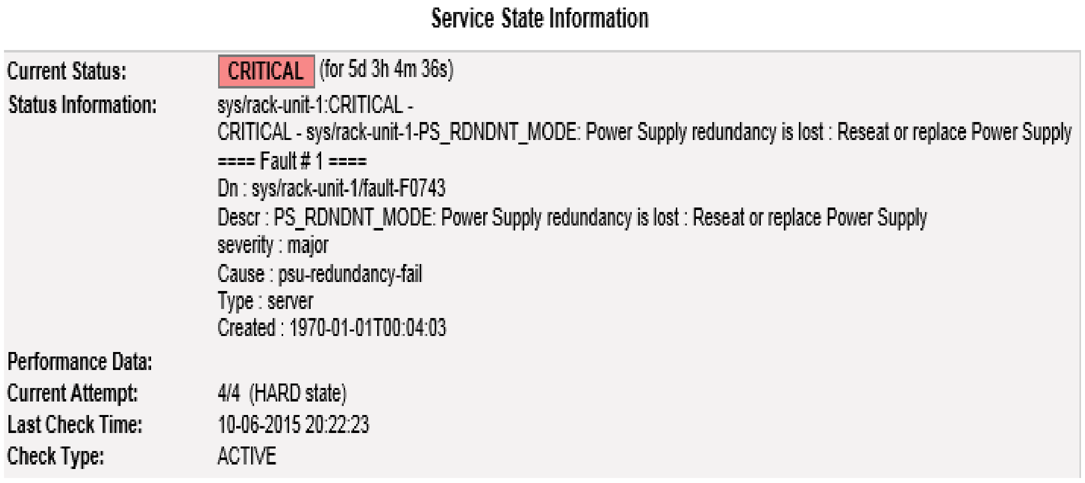
User can modify this mapping by updating the below lists present in the plugin configuration file. Any fault severity which is not mentioned in this list will be considered in Nagios “OK” state.

```
# Define User Mapping for Nagios Critical and Warning
# with UCS fault states. It is a regex based mapping

NAGIOS_CRITICAL=critical|major
NAGIOS_WARNING=minor|warning
```

3.3 Detail Fault View

The monitoring plugin for Cisco UCSM will fetch the relevant faults details for a given dn or class.

For example a fault as major on the chassis will be depicted as critical and the fault details will be shown on the service state information page of the Nagios service.



Service State Information

Current Status:	CRITICAL (for 5d 3h 4m 36s)
Status Information:	sys/rack-unit-1:CRITICAL - CRITICAL - sys/rack-unit-1-PS_RDNDNT_MODE: Power Supply redundancy is lost : Reseat or replace Power Supply ==== Fault # 1 ==== Dn : sys/rack-unit-1/fault-F0743 Descr : PS_RDNDNT_MODE: Power Supply redundancy is lost : Reseat or replace Power Supply severity : major Cause : psu-redundancy-fail Type : server Created : 1970-01-01T00:04:03
Performance Data:	
Current Attempt:	4/4 (HARD state)
Last Check Time:	10-06-2015 20:22:23
Check Type:	ACTIVE

Figure 4 : UCS Fault Details View in Nagios

3.4 Fault for Operational Power State

The plugin has the capability to detect and show faults for the UCS blades and Rack servers depending on their Operating Power State.

When a service check is run on the blade or rack unit, the plugin internally validates the power state of that blade and returns a fault message if it's in an unhealthy condition.

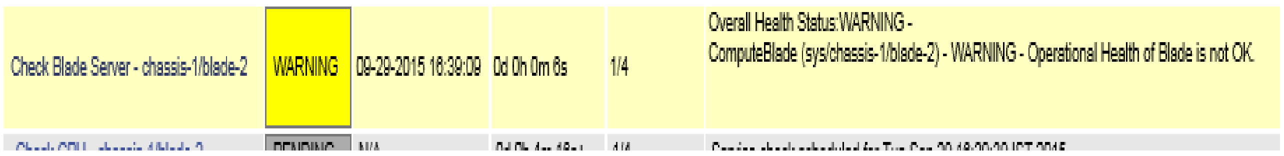
This validation of power state is only carried if the blade/rack server is in associated condition i.e. the blade has some service profile attached to it. If the blade is not associated the plugin will skip the power state validation process and treat it as healthy.

The fault state of a blade is controlled by below entries in the "cisco_ucs_nagios.cfg" file:

```
POW_STATE_CRITICAL= failed|degraded|error|unknown|not-supported
POW_STATE_WARNING= off|offline|offduty|power-save|test
POW_STATE_HEALTHY= on|online|ok
```

User can configure different states as CRITICAL, WARNING and HEALTHY by modifying this list.

The plugin will show a fault if the operating state is from "CRITICAL" or "WARNING" list otherwise will not display anything if it's in healthy state.



Check Blade Server - chassis-1/blade-2	WARNING	09-29-2015 18:39:09	0d 0h 0m 6s	1/4	Overall Health Status:WARNING - ComputeBlade (sys/chassis-1/blade-2) - WARNING - Operational Health of Blade is not OK.

Figure 5 : UCS Fault View for Operation Power State in Nagios

4. Monitoring Plugin

4.1 Plugin Script

As per the Nagios standards, the Cisco UCS Nagios monitoring plugin takes multiple standard inputs like the host information, connection information and service status criteria. The plugin is named as "cisco_ucs_nagios" *and can take the following cli inputs*

Argument Name	***Description***	***Remarks***
-H/ --host	*IP/FQDN of a UCS domain*	*Required*
-n/--port	*UCS domain http/https port*	*Optional*
--NoSsl	*If defined this flag will turn off the secure communication (SSL) with the UCS domain*	*Optional*
-u /--user	*UCS domain login user name.* ***Note:*** *In case user is a domain user then the user field should be defined as* *"ucs-<Domain>\<User>"* *User needs to prefix the domain with "ucs-"* *Example* *"ucs-QALAB\admin"*	*Required*
-p/--password	*UCS domain login password*	*Required, if --passEnc is not provided.Either of the one should be provided.*
--passEnc	*Base64 encoded password for internal framework communication. Services added by auto discovery uses this variable.*	*Required, if -p or --password is not provided. Either of the one should be provided.*
-t/-type	*Query type, i.e class or dn.* *Example* `-t class` *or* `--type dn.`	*Required*

-q/--query	*Value of query class or dn.*
	Example
	if -t was dn then -q should be a dn like
	`–q sys/chassis-1,sys/chassis-1/blade-1`
	If -t was class then -q should be
	`--query EquipmentBlade or`
	`–q EquipmentFan`
		Required

-a/ --attribute	*Attribute that user want to fetch value for. User can either provide just the attribute name or user can provide the attribute name and the user given name which will be displayed on the Nagios web UI.* *So, user can either provide just the* `-a <attr name>` *or* `--attribute <attr name>:<user given name>` *If <user given name> is not provided then the <attr name> will be displayed in the Nagios web UI along with its value.* *If <user given name> is provided the web UI will display the <user given name> along with the attribute value.*	*In case user provide -r,-w or –c options then it is required else optional.*
-w/ --warn	*User defined Warning level*	*In case user provides a –c option then this is required else optional*
-c/ --critical	*User defined Critical level*	*In case user provides a –w option then this is required else optional.*
-r/ --regex	*Regular Expression for matching the pass condition which will result in service status in Nagios as 'OK' else it will be 'CRITICAL'*	*Optional*
--proxy	*Proxy URL for connecting to the UCS domains.*	*Optional*

	Example
	`--proxy http://<Proxy IP>:<Port>`
	`--proxy http://user:pass@<Proxy IP>:<Port>`	

-R / --inHierarchical	*If specified this will provide a hierarchical overall health status for all the elements under the given class or dn. The information that user may want from this option can be controlled via CLASS_FILTER_LIST parameter defined in the plugin configuration file.*	*Optional*
--verbose	*If specified it will work with inHierarchical flag and will provide a detailed status information for all the subcomponents which may be there in the provided dn or class.*	*Optional*
-F / --faultDetails	*If specified it will work with inHierarchical flag and will look for fault details under the given class or dn. It is quick way for checking the overall status of the given dn or class*	*Optional*

-f / --filter	*Provide a filter string in the format* `attribute:value`*. This filter is only valid for type class and will apply on the class provided in the CLI. User can provide wildcard filter which uses standard regular expression syntax.*
	Example
	`--filter dn:sys/chassis-1/blade-1`
	`--filter dn:sys/chassis-1/blade-1/*`
	`--filter dn:^sys/chassis-1$`
		Optional

--debug	*If defined it will print the xml communication between the plugin and UCS domain. It is also helpful for detailed debugging in case of any error that may have occurred while using this CLI. Use this for debug purpose only.*	*Optional*
-S/--useSharedSession	*If specified it will try and reuse an existing UCSM connection for a specified user.* *If the connection does not exist, then a session will be created and left for other processes to reuse this connection again.* *Else, if not defined, plugin will create a new user session every time and will destroy this after each run.*	*Optional*
--getPerfStats	*If this flag is specified this will provide performance data for the Nagios to use to plot graphs. This flag can only be used when -a option is used.*	*Optional*
--version	*If specified, it will print the current Cisco UCSM Nagios Monitoring plugin version.* ***NOTE:*** *Any other options will be ignored.*	*Optional*
-h/--help	*Prints the help content and the plugin input arguments supported*	*Optional*

Table 4 : Plugin Argument Parameters

There are multiple ways in which this script can work. For example in a conventional way, user can provide a range for warning or critical values and based on the given values the plugin script can decide the service state.

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN> -t
class -q equipmentFanStats -a speed -w 3000 -c 4000
```

Else user can also provide a regular expression as OK or CRITICAL criteria.

```
# cisco_ucs_nagios -u <username> -p <password> -H <UCSM IP/FQDN> -t dn
-q sys/chassis-1/blade-2/health-led -a color -r green
```

By default the script uses the Cisco UCSM faults as the basis for returning the service state. Here user can just pass a dn or class as query and the plugin script will return CRITICAL, WARNING or OK as per the faults found on that dn or class.

```
# cisco_ucs_nagios -u <username> -p <password> -H <UCSM IP/FQDN> -t dn
-q sys/chassis-1/blade-2
```

So based on the query it will fetch all the related faults and if this query has a critical fault then the plugin script will return the service as CRITICAL.

In case there is no fault in the query passed then Nagios plugin script will fetch the relevant inventory information and will display the same on the Nagios web UI or CLI.

***Note**: In one of the CLI combination where end user passes the `'--inHierarchical'` flag with `'--verbose'` flag, user may get lot of information as per the query passed.*

To help end user with limiting the required information we have provided a filter variable named `CLASS_FILTER_LIST` where end user can provide name of those sub classes that user want the information for.

So for example, for `ComputeBlade` class there are number of subclasses like `BiosVfConsoleRedirection`, `ComputeBoard`, `MemoryArray`, `MemoryUnit`, `BiosUnit`, `MgmtController`, `AdaptorHostEthIfFsm` etc to name some.

User may only be interested in say `ComputeBoard`, `MemoryArray` and `MemoryUnit` then these classes can be defined in this filter list and the plugin will then only display the status information only related to these three classes.

4.2 Plugin CLI Example

Below are some examples of different CLI options that can be used for fetching different type of information and status for a given query (dn or class).

***CLI (DN as input)** – This will provide status for only the given DN*

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN> -t "dn" -q "sys/chassis-1"
```

Output

```
sys/chassis-1:OK - partNumber : 68-4777-02,serial : FOX1721GVH5
```

***CLI (Class as input)** – This will provide the status for all chassis objects in given UCS domain.*

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN>  -t "class" -q "EquipmentChassis"
```

Output

```
sys/chassis-1:CRITICAL -
CRITICAL - sys/chassis-1-Power state on chassis 1 is redundancy-failed
sys/chassis-2:CRITICAL -
CRITICAL - sys/chassis-2-Power state on chassis 2 is redundancy-failed
==== Fault # 1 ====
Dn : sys/chassis-1/fault-F0408
Descr : Power state on chassis 1 is redundancy-failed
severity : major
Cause : power-problem
Type : environmental
Created : 2013-09-16T23:42:17.258
```

```
==== Fault # 2 ====
Dn : sys/chassis-2/fault-F0408
Descr : Power state on chassis 2 is redundancy-failed
severity : major
Cause : power-problem
Type : environmental
Created : 2013-08-19T18:32:16.878
```

***CLI** (with **–a**, **-w and –c**) – Here the end user can provide a warning and a critical value for a given attribute. Based on these inputs the plugin will return the service status as per the attribute value.*

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN>  -t "class" -q "EquipmentFanStats" -a SpeedAvg -w 3600 -c 3700
```

Output

```
Overall Status : CRITICAL -
WARNING - sys/chassis-1/fan-module-1-1/fan-1/stats - SpeedAvg : 3696
CRITICAL - sys/chassis-1/fan-module-1-1/fan-2/stats - SpeedAvg : 3828
OK - sys/chassis-1/fan-module-1-2/fan-1/stats - SpeedAvg : 3520
CRITICAL - sys/chassis-1/fan-module-1-2/fan-2/stats - SpeedAvg : 3740
WARNING - sys/chassis-1/fan-module-1-3/fan-1/stats - SpeedAvg : 3608
WARNING - sys/chassis-1/fan-module-1-3/fan-2/stats - SpeedAvg : 3696
OK - sys/chassis-1/fan-module-1-4/fan-1/stats - SpeedAvg : 3564
CRITICAL - sys/chassis-1/fan-module-1-4/fan-2/stats - SpeedAvg : 3740
WARNING - sys/chassis-1/fan-module-1-5/fan-1/stats - SpeedAvg : 3652
CRITICAL - sys/chassis-1/fan-module-1-5/fan-2/stats - SpeedAvg : 3828
OK - sys/chassis-1/fan-module-1-6/fan-1/stats - SpeedAvg : 3476
WARNING - sys/chassis-1/fan-module-1-6/fan-2/stats - SpeedAvg : 3696
WARNING - sys/chassis-1/fan-module-1-7/fan-1/stats - SpeedAvg : 3608
CRITICAL - sys/chassis-1/fan-module-1-7/fan-2/stats - SpeedAvg : 3784
OK - sys/chassis-1/fan-module-1-8/fan-1/stats - SpeedAvg : 3520
OK - sys/chassis-1/fan-module-1-8/fan-2/stats - SpeedAvg : 3564
```

***CLI** (with **–a and –r**) – The end user can provide a regular expression for a given attribute and based on these inputs the plugin will return if the service status is OK or in a CRITICAL state.*

Output

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN> -t "dn" -q sys/switch-A/slot-1/switch-ether/port-7  -a operState -r up
CRITICAL - sys/switch-A/slot-1/switch-ether/port-7 - operState : sfp-not-present
```

***CLI** (with **-a** and **--getPerfStats**) – The end user can provide the getPerfStats flag with attribute option. When this flag is set then the CLI will return the performance data appended to the other output via a pipeline "|".*

Output

```
# cisco_ucs_nagios -u "admin" -p "Nbv12345" -H "10.65.183.5" -t "dn" -q "sys/switch-A/fan-module-1-2/fan-2/stats" -a "speed" –getPerfStats
```

```
OK - sys/switch-A/fan-module-1-2/fan-2/stats - speed : 9215|speed=9215
```

***CLI** (with **–a, -w, -c** and **--getPerfStats**) – Here the end user can provide a warning and a critical value for a given attribute. Based on these inputs the plugin will return the service status as per the attribute value. With getPerfStats flag the attribute value and the warning and critical values are used to return the performance data.*

Output

```
# cisco_ucs_nagios -u "admin" -p "Nbv12345" -H "10.65.183.5" -t "dn" -q "sys/switch-A/fan-module-1-2/fan-2/stats" -a "speed" -w 10000 -c 12000 --useSharedSession –getPerfStats

OK - sys/switch-A/fan-module-1-2/fan-2/stats - speed : 9215|speed=9215;10000;12000;
```

***CLI** (with **--inHierarchical**) – This will provide an overall hierarchical overview of health status for the given query*

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN>  -t "dn" -q "sys/chassis-1"  --inHierarchical
```

Output

```
Overall Health Status:CRITICAL -

EquipmentChassis (sys/chassis-1) - CRITICAL - Power state on chassis 1 is redundancy-failed

*** Hierarchical Fault Filtering ON ***

 Please Check CLASS_FILTER_LIST property.
```

***CLI** (**with –inHierarchical** and **--verbose**) – This will provide a detailed status for a given query and the output details can be controlled via `CLASS_FILTER_LIST` property as detailed in Section 5 note.*

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN> -t "dn" -q "sys/chassis-1/blade-2"  –inHierarchical  --verbose
```

Output

```
Overall Health Status:OK -
ComputeBlade (sys/chassis-1/blade-2)- OK
  LsbootDef (sys/chassis-1/blade-2/boot-policy)- OK
  ComputeBoard (sys/chassis-1/blade-2/board)- OK
    MemoryArray (sys/chassis-1/blade-2/board/memarray-1)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-12)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-11)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-10)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-9)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-8)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-7)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-6)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-5)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-4)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-3)- OK
      MemoryUnit (sys/chassis-1/blade-2/board/memarray-1/mem-2)- OK
… (text truncated)
*** Hierarchical Fault Filtering ON ***
```

Please Check CLASS_FILTER_LIST property

***CLI** (**with –inHierarchical** and **--faultDetails**) – This will fetch all the components which are faulty (WARNING or CRITICAL) with their fault details. And will display state OK with inventory information for the parent class if no fault has been found in its hierarchy.*

```
# cisco_ucs_nagios  -u <username> -p <password> -H <UCSM IP/FQDN> -t "class" -q
"computeBlade"  –inHierarchical  --faultDetails
```

Output

```
Overall Health Status:WARNING -

  ComputeBoard (sys/chassis-1/blade-1/board) - WARNING - TCA: computeMbPowerStats
consumedPower, current-value = 131.745239, raised above esc value 75.000000

sys/chassis-1/blade-2 -OK - TotalMemory : 32768,AssignedToDn : org-root/ls-
Chassis_1_Blade_2,PartNumber : 74-7334-02,NumOfCpus : 2,NumOfCores : 12


==== Fault Detail ====
Dn : sys/chassis-1/blade-1/board/fault-F35962
Descr : TCA: computeMbPowerStats consumedPower, current-value = 131.745239, raised above
esc value 75.000000
severity : warning
Cause : threshold-crossed
Type : server
Created : 2015-09-18T14:59:25.678


*** Hierarchical Fault Filtering ON ***
 Please Check CLASS_FILTER_LIST property.
```

***CLI** (**with --filter**) – With this option, user can provide a class attribute as a filter string to the plugin CLI. This option helps in reducing the monitoring scope of the plugin.*

Like for example, if user wants to monitor 'processorUnit' health for all the blades in chassis 1, then user can define the plugin cli as

```
# cisco_ucs_nagios -u <username> -p <password> -H <UCSM IP/FQDN> -t class -q
processorUnit --filter dn:sys/chassis-1/*
```

Output

```
sys/chassis-1/blade-1/board/cpu-1:OK - Cores : 4,Model : Intel(R) Xeon(R) CPU           E5520  @
2.27GHz,CPU Speed(Mhz) : 2.266000

sys/chassis-1/blade-6/board/cpu-2:OK - Cores : 4,Model : Intel(R) Xeon(R) CPU           E5520  @
2.27GHz,CPU Speed(Mhz) : 2.266000

sys/chassis-1/blade-6/board/cpu-1:OK - Cores : 4,Model : Intel(R) Xeon(R) CPU           E5520  @
2.27GHz,CPU Speed(Mhz) : 2.266000
```

The filter uses wildcard filtering hence user can provide standard regular expression syntax which can be used to fetch the desired results.

Another example can be to fetch health status for chassis 1.

As there can be more than one chassis in a UCS domain hence a simple filter like "`dn:sys/chassis-1`" may end up matching all chassis-10,chassis-11…chassis-19.

In any such cases it is recommended that user should anchor the filter in between ^ and $ expression, like "dn:^sys/chassis-1$". This filter will only match chassis-1 now.

```
# cisco_ucs_nagios -u <username> -p <password> -H <UCSM IP/FQDN> -t class -q
equipmentChassis --filter dn:^sys/chassis-1$
```

Output

```
sys/chassis-1:CRITICAL -

CRITICAL - sys/chassis-1-Current connectivity for chassis 1 does not match discovery policy:
unsupported-connectivity
```

CRITICAL - sys/chassis-1-Power state on chassis 1 is redundancy-failed

(text truncated)

Although the above result can easily be achieved by using query type as ‘`dn`’ and query as ‘`sys/chassis-1`’.

```
# cisco_ucs_nagios -u <username> -p <password> -H <UCSM IP/FQDN> -t dn -q sys/chassis-1
```

Output

```
sys/chassis-1:CRITICAL -

CRITICAL - sys/chassis-1-Current connectivity for chassis 1 does not match discovery policy:
unsupported-connectivity

CRITICAL - sys/chassis-1-Power state on chassis 1 is redundancy-failed

(text truncated)
```

***NOTE**: Filter option only work with query type class..*

5. Auto Discovery Addon

5.1 Working With Auto Discovery

Currently auto discovery addon creates host and services in the Nagios system as per the details provide in the table below.

Host	***Services***	***Service Details***
UCS Domain	*Ping UCS Domain virtual IP*	*This service will check the ping to the given UCS domain.*
Chassis	*Check Chassis*	*This service checks overall faults status of the UCS chassis and will return Nagios state accordingly. If there are no faults on the UCS chassis then it will display the inventory information for the same.*
	Check PSU	*This service checks just the faults that may have occurred on the chassis PSU's and will return Nagios state accordingly. If there are no faults on the chassis PSU(s) then it will display the inventory information for the same.*
Fex	*Check Fex*	*This service checks overall faults status of the UCS FEX and its sub components. Based on this service will return Nagios state accordingly. If there are no faults on the UCS FEX then it will display the inventory information for the same.*
FI	*Check FI*	*This service checks overall faults status of the UCS FI switch and its sub components. Based on this service will return Nagios state accordingly. If there are no faults on the UCS FI switch then it will display the inventory information for the same.*
IOM	*Check IOM*	*This service checks overall faults status of the UCS Chassis IOM module and its sub components. Based on this service will return Nagios state accordingly. If there are no faults on the UCS Chassis IOM module then it will display the inventory information for the same.*
Blade Server	*Check Blade Server*	*This service checks overall faults status of the UCS blade server and its sub components. Based on this service will return Nagios state accordingly. If there are no faults on the UCS blade server then it will display the inventory information for the same.*
	Check CPU	*This service checks all the faults that may have occurred on the blade CPU(s) and will return Nagios state accordingly. If there are no faults on the blade CPU(s) then it will display the inventory information for the same.*
	Check Memory	*This service checks overall faults status of the memory array units and its sub components. Based on this service will return Nagios state accordingly. If there are no faults on the memory array unit then it will display the inventory information for the same.*
Rack Server	*Check Rack Server*	*This service checks overall faults status of the UCSM managed C-series rack server and its sub components. Based on this service will return Nagios state accordingly. If there are no faults on the UCSM managed rack server then it will display the inventory information for the same.*
	Check CPU	*This service checks all the faults that may have occurred on the rack CPU(s) and will return Nagios state accordingly. If there are no faults on the rack CPU(s) then it will display the inventory information for the same.*
	Check Memory	*This service checks overall faults status of the memory array units and its sub components. Based on this service will return Nagios state accordingly. If there are no faults on the UCS rack memory array then it will display the inventory information for the same.*
	Check PSU	*This service checks just the faults that may have occurred on*

		the rack PSU(s) and will return Nagios state accordingly. If there are no faults on the PSU(s) then it will display the inventory information for the same.

Table 5 : Host and Service Mapping

The auto-discovery add-on script can either be manually invoked or can be added to a cron job for periodic inventory checks. This script can take the following optional inputs from the end user.

Argument Name	***Description***
-f / --csvFile	*User can provide an absolute path with the UCS host csv file name. If this option is not provided then by default the path is taken as the current working directory and filename is UCSHostInfo.csv*
-r /--restartService	*This option provides end user with the flexibility of restarting the Nagios service after the auto discovery is finished. The default is not to restart the Nagios service. If user wants to restart the service he can just pass this option in the CLI.*
-H / --Host	*IP/FQDN of a UCS domain.* ***Note****: If this option is given the script will skip the CSV file and only discover the Host provided by this option.*
-u / --user	*UCS domain login user name.* ***Note:*** *In case user is a domain user then the user field should be defined as "ucs-<Domain>\<User>"* *User needs to prefix the domain with "ucs-"* *Example* *"ucs-QALAB\admin"*
-p / --password	*Specifies UCS users password to login to the server.*
-n / --port	*This specifies the UCS Manager http/https port.*
--NoSsl	*If defined this flag will turn off the secure communication (SSL) with the UCS domain.*

--proxy	*User can specifies a proxy url that contains proxy connection and optionally authentication information.*
	Example
	`--proxy http://<Proxy IP>:<Port>`
	`--proxy http://user:pass@<Proxy IP>:<Port>`

-D/--disable-hostgroup-creation	*This option provides user a way to disable default host groups creation via auto-discovery.*
-S, --singleHost	*If defined this flag will disable multiple host creation and will just create one host i.e UCS Domain and put all discovered services under it.*
--debug	*If defined it will print the xml communication between the plugin and UCS domain. It is also helpful for detailed debugging in case of any error that may have occurred while using this CLI. Use this for debug purpose only.*
--version	*If specified, it will print the current Cisco UCSM Nagios Auto Discovery version.* ***NOTE:*** *Any other options will be ignored.*
-h/--help	*Prints the help content and the cli input arguments supported.*

Table 6 : Auto discovery CLI options

Argument Name	***Description***
ASSOCIATED_BLADES_ONLY	*This flag toggles the Auto-Discovery add-on behavior to discover the blades which are associated with some service-profile.* *If set to "FALSE" then all blades will be discovered.* *If set to "TRUE" then blades with service profile will be discovered and others will be skipped.*
REMOVE_OLD_DISCOVERY	*This flag in the configuration file sets the add-on*

	script behavior to either remove old discovered device or keep old discoveries and update only the new UCS domains. *If this flag is set to False, then add-on will not remove the previously discovered objects.* *If this flag is set to True, then the old discoveries will be removed. This is the default behavior.*

Table 7 : Auto discovery CFG file options

If the script is invoked without using the "-r /--restartService" cli options then, at the end of the discovery process it will prompt end user for input on restarting the Nagios service.

5.2 Add Service

As an advance feature in the auto discovery add-on, user can define his own services around the UCS components.

This can be done by editing the `'NagiosAutoDiscoveryUCS.cfg'` *and updating the different service variables which are defined as 'Service_' suffixed by the class name of the components.*
For example:
- *Service_EquipmentChassis*
- *Service_ComputeBlade*
- *Service_EquipmentFex*
- *Service_EquipmentNetworkElementFanStats*

Here user can provide his own service name and service class or dn which is restricted to sub classes or dn of the above said classes. Optionally, user can also provide various cli options that user want to pass to the monitoring plugin script.

This value should be in the following format
```
<service name>:<class or dn> ,<service name>:<class or dn>:<optional cli
options>
```

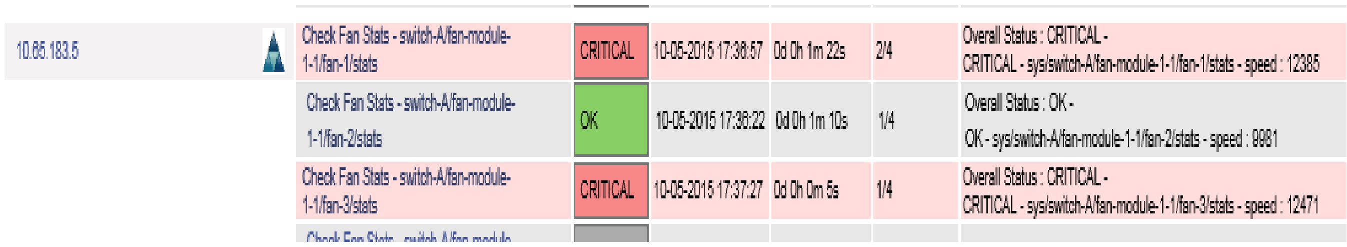
For example user can update the `NagiosAutoDiscoveryUCS.cfg` *with the following custom service list like*

```
Service_ComputeRackUnit = Fault Status:ComputeRackUnit:"--inHierarchical --
faultDetails", Processors:processorUnit, Memory:memoryArray,
Adaptor:adaptorUnit

Service_EquipmentNetworkElementFanStats=Check Fan
Stats:EquipmentNetworkElementFanStats:"--useSharedSession -a speed -w 10000 -c
12000 --getPerfStats"
```

*If user wants to get these services to be discovered by Auto-Discovery AddOn, then the Class Name should be added in the "**DISCOVERY_CLASS_LIST**" present in the same configuration file. This entry needs to be made when user is adding a new entry of "Service_<class_name>" and wants these services to be discovered.*

```
#This List defines the Class list for which Services need to be
discovered.
#DISCOVERY_CLASS_LIST=EquipmentChassis,ComputeBlade,EquipmentFex,…,<New
_Class>

DISCOVERY_CLASS_LIST=EquipmentChassis,ComputeBlade,...,EquipmentNetwork
ElementFanStats
```

By default all the services created via new "Service_" will be kept under the Domain host. So, now when the auto discovery is executed again the following list of services will appear in the Nagios web UI.



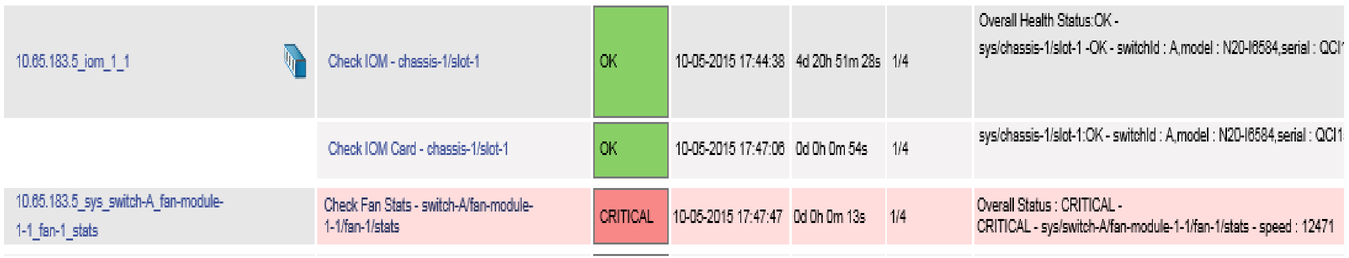
10.65.183.5	Check Fan Stats - switch-A/fan-module-1-1/fan-1/stats	CRITICAL	10-05-2015 17:38:57	0d 0h 1m 22s	2/4	Overall Status : CRITICAL - CRITICAL - sys/switch-A/fan-module-1-1/fan-1/stats - speed : 12385
	Check Fan Stats - switch-A/fan-module-1-1/fan-2/stats	OK	10-05-2015 17:38:22	0d 0h 1m 10s	1/4	Overall Status : OK - OK - sys/switch-A/fan-module-1-1/fan-2/stats - speed : 8981
	Check Fan Stats - switch-A/fan-module-1-1/fan-3/stats	CRITICAL	10-05-2015 17:37:27	0d 0h 0m 5s	1/4	Overall Status : CRITICAL - CRITICAL - sys/switch-A/fan-module-1-1/fan-3/stats - speed : 12471
	Check Fan Stats - switch-A/fan-module					

Figure 6 : Custom Service under domain host

*If user wants that these new services should have their own host and should not be placed under domain. Then an entry for the class should be made under **"HOST_CLASS_LIST"** parameter present in the same configuration file.*

```
#This List defines the Class Names which can have Nagios Host Created.
HOST_CLASS_LIST=EquipmentChassis,ComputeBlade, EquipmentFex
,...,<New_Class>
```

Now on re-running the auto-discovery process these new services will be placed under a new host of that class.



10.65.183.5_iom_1_1	Check IOM - chassis-1/slot-1	OK	10-05-2015 17:44:38	4d 20h 51m 28s	1/4	Overall Health Status:OK - sys/chassis-1/slot-1 -OK - switchId : A,model : N20-I6584,serial : QCI
	Check IOM Card - chassis-1/slot-1	OK	10-05-2015 17:47:06	0d 0h 0m 54s	1/4	sys/chassis-1/slot-1:OK - switchId : A,model : N20-I6584,serial : QCI1
10.65.183.5_sys_switch-A_fan-module-1-1_fan-1_stats	Check Fan Stats - switch-A/fan-module-1-1/fan-1/stats	CRITICAL	10-05-2015 17:47:47	0d 0h 0m 13s	1/4	Overall Status : CRITICAL - CRITICAL - sys/switch-A/fan-module-1-1/fan-1/stats - speed : 12471

Figure 7 : Custom Service with its own host

6. Customizing Monitoring Plugin

6.1 Customize Inventory information

Another feature in the plugin is a provision wherein user can select the required valid attribute(s) for a given class and provide a user friendly name to these attribute that user wants to see on the Nagios Web UI. This can be configured via the configuration file "`cisco_ucs_nagios.cfg`" which is found in the same location as that of the monitoring plugin.

For fetching Inventory attributes from the class user needs to provide "Inv_" as prefix followed by the class name as the variable name and the list of attributes as its value.

So the configuration property string will be of the following type

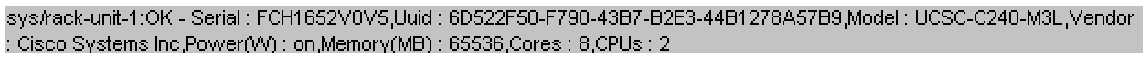
```
Inv_<class id >=<AttributeName>,<AttributeName>:<UserGivenName>,<AttributeName>
```

Here user can also customize the attribute name that user wants to see on the Nagios Web UI. So for example, if the attribute name is say '`OperPower`' and user wants that to be seen as say '`Power(W)`', then user can define the attribute first followed by a colon ":" and the name that user wants to see on the Nagios Web UI.

Like `OperPower:Power(W)`

So a complete example for a class '`ComputeRackUnit`', *the entry may look like*

```
Inv_ComputeRackUnit=Serial,Uuid,Model,Vendor,OperPower:Power(W),TotalMemory:Memo
ry(MB),NumOfCores:Cores,NumOfCpus:CPUs
```



```
sys/rack-unit-1:OK - Serial : FCH1652V0V5,Uuid : 6D522F50-F790-43B7-B2E3-44B1278A57B9,Model : UCSC-C240-M3L,Vendor
: Cisco Systems Inc,Power(W) : on,Memory(MB) : 65536,Cores : 8,CPUs : 2
```

Figure 8 : Custom Inventory Information

6.2 Customize Statistics Information

The plugin provides user with flexibility to select the required valid attribute(s) for a given class to be used as performance data. This can be configured via the configuration file "cisco_ucs_nagios.cfg" which is found in the same location as that of the monitoring plugin.

One could add entries in this configuration file for getting performance stats for specific "Class" by adding at least one of its attribute.

The basic format of the entry is as shown below:

Stats_<Class Name>= <Attribute_Name>;<UOM>;<warn>;<crit>;<min>;<max>

<Class Name>: *It's the name of the class for which the stats needs to be generated. The plugin will look for this entry and read the given parameters.*

<Attribute Name>: *This is a MANDATORY parameter. This will be one of the valid attribute from the selected class. This attribute should return a numeric value as graphs are plotted against the numeric values only.*
One could also give an optional name to this attribute by writing this name after ":". If this optional name is given then this will be shown as the label instead of the "attribute name". Below is an example for it.

Stats_MemoryArray=CurrCapacity:"Current Capacity(MBs)"
<class name> = <attribute name> : <attribute optional name>

<UOM>: *Unit of measurement. It's the unit associated to the value of the attribute. This field is OPTIONAL and can be left blank. It can have one of the following values.*

a. *no unit specified - assume a number (int or float) of things (eg, users, processes, load averages)*
b. *s - seconds (also us, ms)*
c. *% - percentage*
d. *B - bytes (also KB, MB, TB)*
e. *c - a continuous counter (such as bytes transmitted on an interface)*

Note:
Allowed Unit of measurement is controlled by "STATS_UOM_LIST" parameter present in configuration file. User can update the list according to the use.

#User can append more "Unit Of Measurements" which they want to allow in getting performance statistics.
STATS_UOM_LIST =%,s,us,ms,c,B,KB,MB,TB

<Warn>,<Crit>,<Min>,<Max> : *These are OPTIONAL parameters. These parameters can either be a numeric value or a parameter of the class returning some numeric number. All these parameters should be in same "Unit of Measurement" as that of label. If any of these value is not present or user does not want to set any parameter for them then he can leave these fields blank.*

Warn – *It sets the warning threshold in graphing the stats for that attribute.*
Crit – *it sets the Critical level threshold in graphing the stats for that attribute*.
Min – *This field sets the minimum possible value for the selected attribute*.
Max – *This field sets the maximum possible value for the selected attribute.*

Below are few possible ways to write Stats Class definition in the configuration file.

```
# Only attribute defined all optional parameters skipped.
Stats_ProcessorUnit=Speed

#Optional name for attribute has been mentioned.
Stats_MemoryArray=CurrCapacity:Current Capacity(MB);B;;;;maxCapacity

#Few optional parameter skipped
Stats_EquipmentNetworkElementFanStats=Speed;;;;speedMin;speedMax

#All optional parameters provided.
Stats_EquipmentNetworkElementFanStats=Speed;c;10000;20000;speedMin;speedMax
```

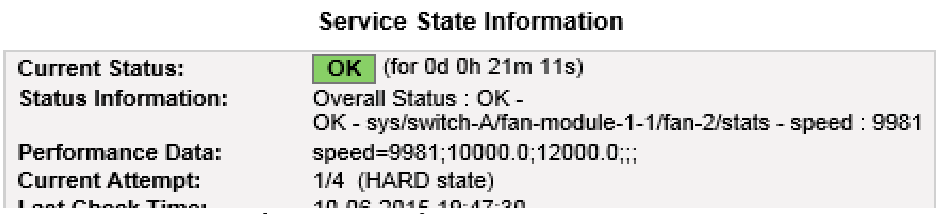
Now when Nagios service is called and uses one of these Stats "class", then with the normal inventory related data, the plugin will also return the listed attribute as performance stats.

Below is the CLI output of a service call:

```
# ./cisco_ucs_nagios - u <username> -p <password> -H <UCSM IP/FQDN> -t class -q EquipmentNetworkElementFanStats --filter dn:sys/switch-A/fan-module-1-1/fan-1/stats

sys/switch-A/fan-module-1-1/fan-1/stats:OK - Speed : 9981,airflowDirection : FrontToBack,speedMin : 10953,speedMax : 11226|Speed=9981;10000.0;12000.0;;;
```

Here the attribute "Speed" after '|' is the performance stat for this service. When such a service is run in Nagios, then this performance stat is stored in historical information database which then a third party graphing tool uses to populate graphs.

On Nagios GUI "Performance Data" field in the service gets populated when this service is run.


Service State Information

Current Status:	OK (for 0d 0h 21m 11s)
Status Information:	Overall Status : OK - OK - sys/switch-A/fan-module-1-1/fan-2/stats - speed : 9981
Performance Data:	speed=9981;10000.0;12000.0;;;
Current Attempt:	1/4 (HARD state)

Figure 9 : Performance Data

Note:

- *The plugin follows Nagios generic guidelines for generating performance data.*
- *User can install any third party graphing plugin from Nagios Communities to populate graphs by using performance data*.

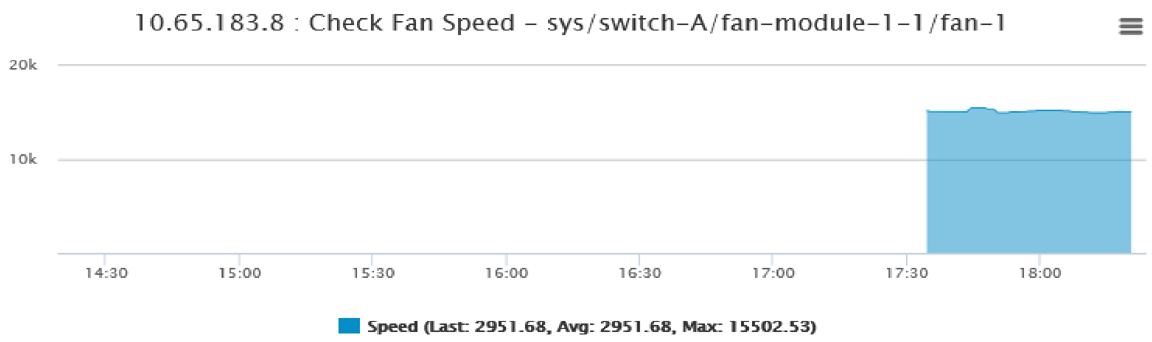


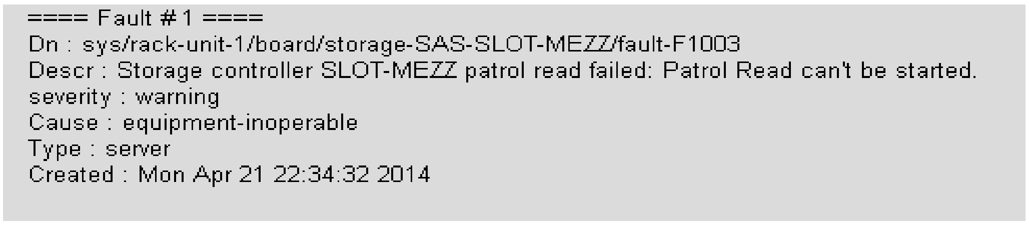
Figure 10 : Graph plotted using performance data

6.3 Customize fault information

User can also control the attributes user wants to see in the fault details by editing the monitoring plugin configuration file 'cisco_ucs_nagios.cfg' and updating the 'FaultInst' variable with the required attribute names.

So for example user can have the following list of attributes which user wants to see as a part of the fault details

```
FaultInst=Dn,Descr,severity,Cause,Type,Created
```



```
==== Fault # 1 ====
Dn : sys/rack-unit-1/board/storage-SAS-SLOT-MEZZ/fault-F1003
Descr : Storage controller SLOT-MEZZ patrol read failed: Patrol Read can't be started.
severity : warning
Cause : equipment-inoperable
Type : server
Created : Mon Apr 21 22:34:32 2014
```

Figure 11 : Custom fault details

6.4 Skipping Faults

With this feature user can define the faults user wants to skip from the Nagios monitoring service.
For example if user wants to skip faults with severity as 'info' then this can be defined as the value for the configuration variable SKIP_FAULT_LIST in the monitoring plugin configuration file 'cisco_ucs_nagios.cfg'.

The format of value to this variable is of type
<fault attribute>:<value>,<fault attribute>:<value>.

Example

```
SKIP_FAULT_LIST=Lc:suppressed,Type:fsm,Severity:info,Severity:condition
```

Note:

Skipping fault based on "Description" field is not advisable as it might contain some special characters which might not let the fault to be skipped when a comparison is done

7. Uninstall

To uninstall the Cisco UCSM Nagios integration, follow the step as mentioned below

a. Extract the installation tar gzip file in a temporary location.
```
# tar zxvf cisco-ucs-nagios-x.x.x .tar.gz
```

b. Now run the installer with '`--uninstall`' option
```
# ./installer.py --uninstall
```

NOTE: Uninstaller will prompt before trying to stop Nagios service. Uninstallation process requires Nagios services to be stopped first.

8. Known Caveats

8.1 Frequent Service timeouts

If user has a large UCSM domain with more than 600 services and is seeing frequent service timeout, then it is recommended that the user should check for network related issues. For example ping timeouts, high network latencies, etc.

If the above doesn't help, then user may try and tweak the following parameters in Nagios configuration file 'nagios.cfg' to check if this resolves the issue.

Service Check Timeout

Format: service_check_timeout=<seconds>
Example: service_check_timeout=600

This is the maximum number of seconds that Nagios will allow service checks to run. If the network is slow or UCSM is slow in responding to the xml requests then it is recommended to increase this value and check the results.

It may be a case that increasing only this value may not help.It is then recommended that user should use this value in conjunction with 'max_concurrent_check' option.

Maximum Concurrent Service Checks

Format: max_concurrent_checks=<max_checks>
Example: max_concurrent_checks=50

In case of slow Nagios host or network or UCSM responding slowly to the xml requests, user is recommended to keep this value to a minimum. This option will run minimum number of concurrent services on the Nagios host thereby stabilizing the system and in turn handling the frequent service time out issue.

More details on the above Nagios configuration options can be found at the following link http://nagios.sourceforge.net/docs/3_0/configmain.html