

The Industry Standard in IT Infrastructure Monitoring

Purpose

This document is designed to assist Nagios administrators in understanding and using the Negate plugin in Nagios® XI™. The Negate plugin allows for any standard plugin output to be reversed and is very useful with hosts or services that are expected to be in a Critical or Warning state but you wish to show them as OK. This function can be used for the opposite effect (i.e. showing a CRITICAL state when the actual state is OK).

Target Audience

The intended audience for this document is Nagios administrators with hosts or services that are expected to be in a critical or warning state but wish to show them as OK and vice versa. An understanding of how plugins work and return exit codes will help understand the negate plugin.

What Is The Negate Plugin

Negate is used to execute other plugins, the state returned by the other plugin can be changed by the negate plugin. For example when a check is normally considered to be in a Critical or Warning state, but the system administrator would instead prefer to see an OK when in such a state. Understand that this will not turn any check consistently to an OK state, but will reverse a critical to OK or an OK to critical, depending on the actual check being run.

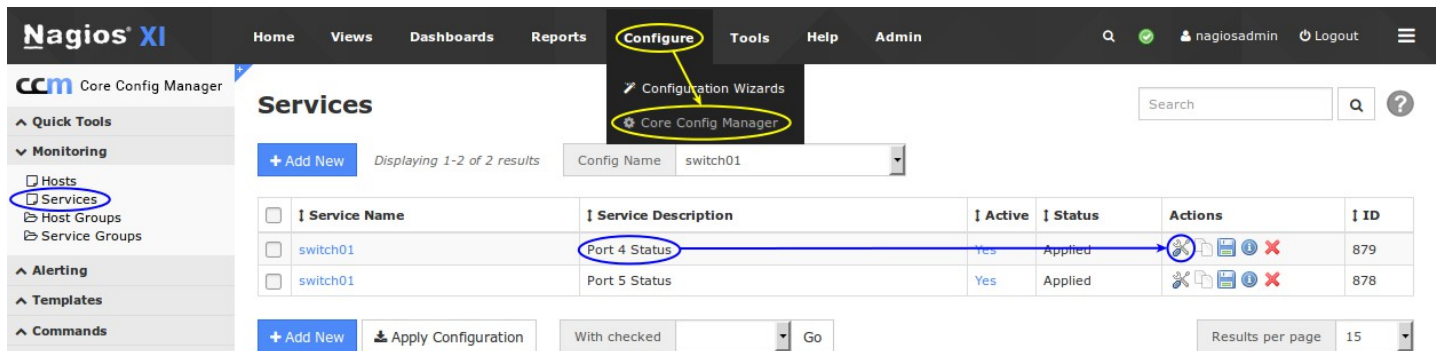
Negate Plugin Example

For our example we will use a service check for Port 4 Status on a network switch. Below is the service check for Port 4 Status which is currently not being used and is in a Critical state:

Host	Service	Status	Duration	Attempt	Last Check	Status Information
switch01	Port 4 Status	Critical	5m 43s	5/5	2017-01-27 13:54:36	CRITICAL: Interface tengigabitethernet1/0/4 (index 4) is down.
	Port 5 Status	Ok	4m 53s	1/5	2017-01-27 13:51:28	OK: Interface tengigabitethernet1/0/5 (index 5) is up.

Before showing you how to use the negate plugin with this service, lets understand how the Port 4 Status service works at the command line. To do that we need to view the service definition in Core Configuration Manager (CCM).

Navigate to **Configure > Core Configuration Manager > Monitoring > Services** and locate the **Port 4 Status** check.



Click the **modify** icon to view the service configuration.

Notice the **Check command** drop down has **xi_service_check_ifoperstatus** selected.

Check command

check_xi_service_ifoperstatus

The **Command View** field shows what arguments are used for this command.

Command view

```
$USER1$/check_ifoperstatus -H $HOSTADDRESS$ -C $ARG1$ -k $ARG2$ $ARG3$
```

The **\$ARGx\$** fields are the values being used for the command. When Nagios XI executes this command it replaces the **\$variables\$** with actual values, which results in something like:

\$ARG1\$ box293
 \$ARG2\$ 4
 \$ARG3\$ -v 2 -p 161

```
/usr/local/nagios/libexec/check_ifoperstatus -H 10.25.4.3 -C box293 -k 4 -v 2 -p 161
```

When this is executed at the command line, the output is:

```
[root@xitest ~]# /usr/local/nagios/libexec/check_ifoperstatus -H 10.25.4.3 -C box293 -k 4 -v 2 -p 161
CRITICAL: Interface tengigabitethernet1/0/4 (index 4) is down.
[root@xitest ~]# echo $?
2
```

You'll notice the first line of output is the "CRITICAL: Interface tengigabitethernet1/0/4 (index 4) is down.". This is actually only for us humans to understand what the result of the plugin was.

The second line **echo \$?** is telling us what the exit code of the plugin was, which is the value 2. The exit code is what tells Nagios XI that the service is in a critical state.

Now lets execute that same command again, but this time use the negate command to turn that 2 state into a 0 state.

```
/usr/local/nagios/libexec/negate /usr/local/nagios/libexec/check_ifoperstatus -H 10.25.4.3 -C box293 -k 4 -v 2 -p 161
```

```
[root@xitest ~]# /usr/local/nagios/libexec/negate /usr/local/nagios/libexec/check_ifoperstatus -H 10.25.4.3 -C box293 -k 4 -v 2 -p 161
CRITICAL: Interface tengigabitethernet1/0/4 (index 4) is down.
[root@xitest ~]# echo $?
0
```

The exit state returned by the plugin is what tells Nagios XI that it's in an OK state, because it's a 0.

You'll notice that the text is still saying CRITICAL, this doesn't affect Nagios but it can be confusing for us humans. There is an additional argument `-s` that will substitute the output text as well:

```
/usr/local/nagios/libexec/negate -s /usr/local/nagios/libexec/check_ifoperstatus -H 10.25.4.3 -C box293 -k 4 -v 2 -p 161
```

```
[root@xitest ~]# /usr/local/nagios/libexec/negate -s /usr/local/nagios/libexec/check_ifoperstatus -H 10.25.4.3 -C box293 -k 4 -v 2 -p 161
OK: Interface tengigabitethernet1/0/4 (index 4) is down.
[root@xitest ~]# echo $?
0
```

Now you can see the text output says OK.

Update Service To Use Negate

Now that you have tested negate from the command line and know how it works you can now implement it in your service definition.

Looking at the original service definition, the command used is `xi_service_check_ifoperstatus`, and the command definition for this is:

```
$USER1$/check_ifoperstatus -H $HOSTADDRESS$ -C $ARG1$ -k $ARG2$ $ARG3$
```

All that is required is to put the negate command in front of this like so:

```
$USER1$/negate -s $USER1$/check_ifoperstatus -H $HOSTADDRESS$ -C $ARG1$ -k $ARG2$ $ARG3$
```

However you shouldn't change the original `xi_service_check_ifoperstatus` command definition as it'll affect all services, instead you can copy the existing command to create a new command.

Navigate to **Configure > Core Configuration Manager > Commands > >_Commands**.

The screenshot shows the Nagios XI interface with the 'Commands' page. The left sidebar has 'Commands' selected. The main table lists two commands. The first row, 'check_xi_service_ifoperstatus', has its name circled in blue. A blue arrow points to the 'copy' icon in the 'Actions' column of this row.

<input type="checkbox"/>	Command Name	Command Line	Active	Actions	ID
<input type="checkbox"/>	check_xi_service_ifoperstatus	\$USER1\$/check_ifoperstatus -H \$HOSTADDRESS\$ -C \$ARG1\$ -k \$ARG2\$ \$ARG3\$	Yes		86
<input type="checkbox"/>	check_xi_service_ifoperstatusnag	\$USER1\$/check_ifoperstatnag \$ARG1\$ \$ARG2\$ \$HOSTADDRESS\$	Yes		85

Locate the **xi_service_check_ifoperstatus** command and click the **copy** icon.

The screenshot shows the Nagios XI interface after a copy action. A message at the top says 'Data successfully inserted to the database! Object check_xi_service_ifoperstatus_copy_1 created.' The table now has three rows. The new row, 'check_xi_service_ifoperstatus_copy_1', has its name circled in blue. A blue arrow points to the 'modify' icon in the 'Actions' column of this row.

<input type="checkbox"/>	Command Name	Command Line	Active	Actions	ID
<input type="checkbox"/>	check_xi_service_ifoperstatus	\$USER1\$/check_ifoperstatus -H \$HOSTADDRESS\$ -C \$ARG1\$ -k \$ARG2\$ \$ARG3\$	Yes		86
<input type="checkbox"/>	check_xi_service_ifoperstatusnag	\$USER1\$/check_ifoperstatnag \$ARG1\$ \$ARG2\$ \$HOSTADDRESS\$	Yes		85
<input type="checkbox"/>	check_xi_service_ifoperstatus_copy_1	\$USER1\$/check_ifoperstatus -H \$HOSTADDRESS\$ -C \$ARG1\$ -k \$ARG2\$ \$ARG3\$	No		182

When the screen refreshes you'll have a duplicate command appended with **_copy_1**.

Click the **modify** icon to edit this command.

You will need to give the command a new name. In this example it seems logical to append the name with **_negate** so the command name is **xi_service_check_ifoperstatus_negate**.

Then you need to add the negate command (**\$USER1\$/negate -s**) to the beginning of the command line.

Lastly you need to click the **Active** checkbox.

Click the **Save** button.

Command Management

This object is currently set as **Inactive** and will not be written to the configuration files.

This screenshot shows the changes that were just made.

Command Name *

Example: check_example

Command Line *

Example: \$USER1\$/check_example -H \$HOSTADDRESS\$ -P \$ARG1\$ \$ARG2\$

Command Type:

Active

Available Plugins

The last step is to update the the Port 4 Status service with the new check command. Navigate to **Configure > Core Configuration Manager > Monitoring > Services** and edit the **Port 4 Status** check.

Use the **Check command** drop down list to select the new command `xi_service_check_ifoperstatus_negate`.

Once selected you'll see the Command view update, it shows the negate command being used.

Click **Save** button and then **Apply Configuration**.

Check command


Command view

\$ARG1\$

\$ARG2\$

\$ARG3\$

After the configuration is applied and the Port 4 Status service is checked, the service will now be in an OK state:

Host	Service	Status	Duration	Attempt	Last Check	Status Information
switch01 	Port 4 Status	Ok	9s	1/5	2017-01-27 15:19:34	OK: Interface tengigabitethernet1/0/4 (index 4) is down.
	Port 5 Status	Ok	1h 28m 15s	1/5	2017-01-27 15:15:52	OK: Interface tengigabitethernet1/0/5 (index 5) is up.

You can see that the Port 4 Status service check is in an OK state and the status information shows that the port is down.

This completes using negate to reverse a plugin's output.

Finishing Up

If you have any issues with configuring the negate or its use, or for any other support related questions, please visit the Nagios Support Forums at:

<https://support.nagios.com/forum/>