



Purpose

This document describes how to automate and monitor Windows actions, tasks, and tests using Autolt and Nagios. Autolt is an extremely powerful scripting recorder and editor and this is an example of how it can be used.

Target Audience

This document is intended for use by Nagios XI Administrators interested in monitoring automated actions on Windows systems.

Prerequisites

You must have NSClient++ installed on the Windows machine. NSClient++ must be configured to allow NRPE checks from the Nagios XI server. Information on installing and configuring NSClient++ can be found in the following documents:

- [Installing The XI Windows Agent](#)
- [Configuring The XI Windows Agent](#)
- [Enabling The NRPE Listener In NSClient++ 0.4.x](#)

This guide is specifically aimed at NSClient++ v 0.4.x or newer, the previous 0.3.x version of NSClient++ is no longer supported by the developer of the application.

In addition to this you will need to install the **Firefox** web browser, which can be downloaded from:

<http://www.mozilla.org>

Installing Autolt

Both of the following packages must be installed on the Windows system you intend to automate tasks on.

- Autolt – <http://www.autoitscript.com/site/autoit/downloads/>
- SciTE Script Editor – <http://www.autoitscript.com/site/autoit-script-editor/downloads/>

Once you have downloaded and installed both applications on your Windows system, you are ready to either record an automation script or write one by hand. More information, along with guides and tutorials on writing and recording scripts can be found at the main Autolt site (www.autoitscript.com).

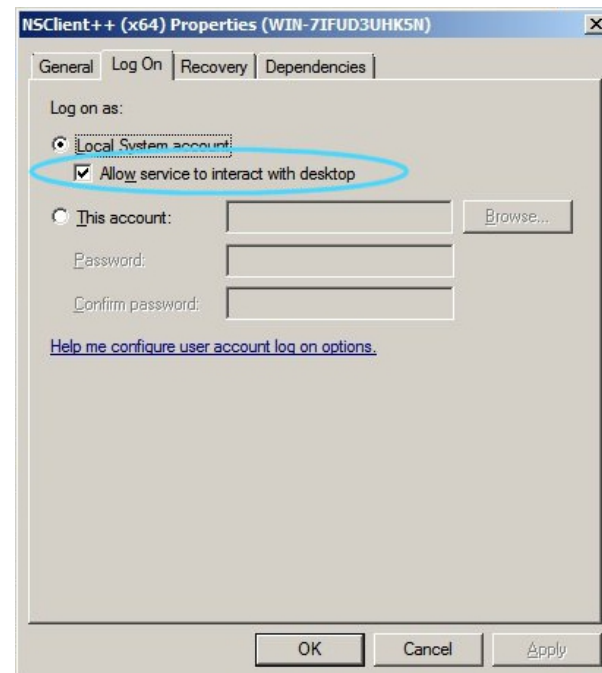
Configure NSClient++ Service

Configure NSClient++ Service

AutoIT by its nature requires control of the Windows workspace, thus it is a good idea to turn on **Allow service to interact with desktop** for the NSClient++ service.

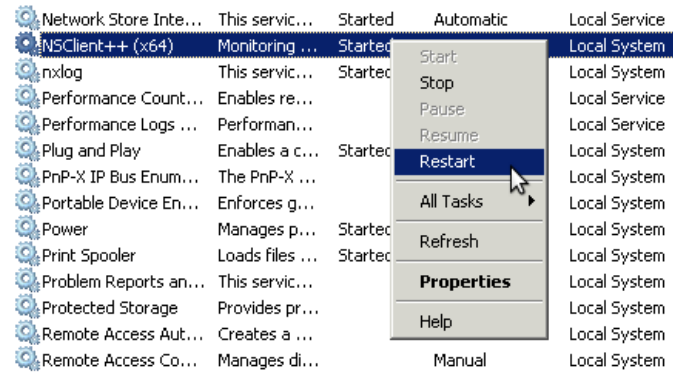
In the Windows open the **Services** console under **Administrative Tools**. If you cannot locate this, use `services.msc` to open the Services console.

- Find the NSClient++ service in the list
- Right click on NSClient++ and select Properties
- Click the **Log On** tab
- Tick the box **Allow service to interact with desktop**
- Click **OK**



Restart NSClient++ Service

Now right click the service and select **Restart**. You can leave the Services console open as you'll need to use it again later to restart the service again.



Create Script

For this example we will be creating a script that opens `firefox.exe`, selects the URL box and enters a URL. After the URL is entered a timer is started which runs for as long as it takes the site to completely load. After it is loaded, the timer ends and the time required to load the page is output to the console.

We have provided an example script below, open the **SciTE Editor** and paste the script below into the editor.

```
Func _WinWaitActivate($title,$text,$timeout=0)
    WinWait($title,$text,$timeout)
    If Not WinActive($title,$text) Then WinActivate($title,$text)
    WinWaitActive($title,$text,$timeout)
EndFunc

$title_string = "Yahoo"
Local $begin = TimerInit()
Run("C:\Program Files (x86)\Mozilla Firefox\firefox.exe www.yahoo.com")
_WinWaitActivate($title_string,"")
Local $dif = TimerDiff($begin)
$time_string = $dif
WinClose($title_string,"")
ConsoleWrite($time_string)
```

The script is opening Firefox to the `www.yahoo.com` address (line 9).

Once the page has loaded, the title of the web browser becomes **Yahoo** and it's this particular string we are looking for. You can see on line 7 we have created a variable that has the word **Yahoo**.

NOTE: You may need to open the URL to make sure this is actually what the title of the web browser becomes, as it may change if the Yahoo site may have changed (*also different countries show different versions of the web page*).

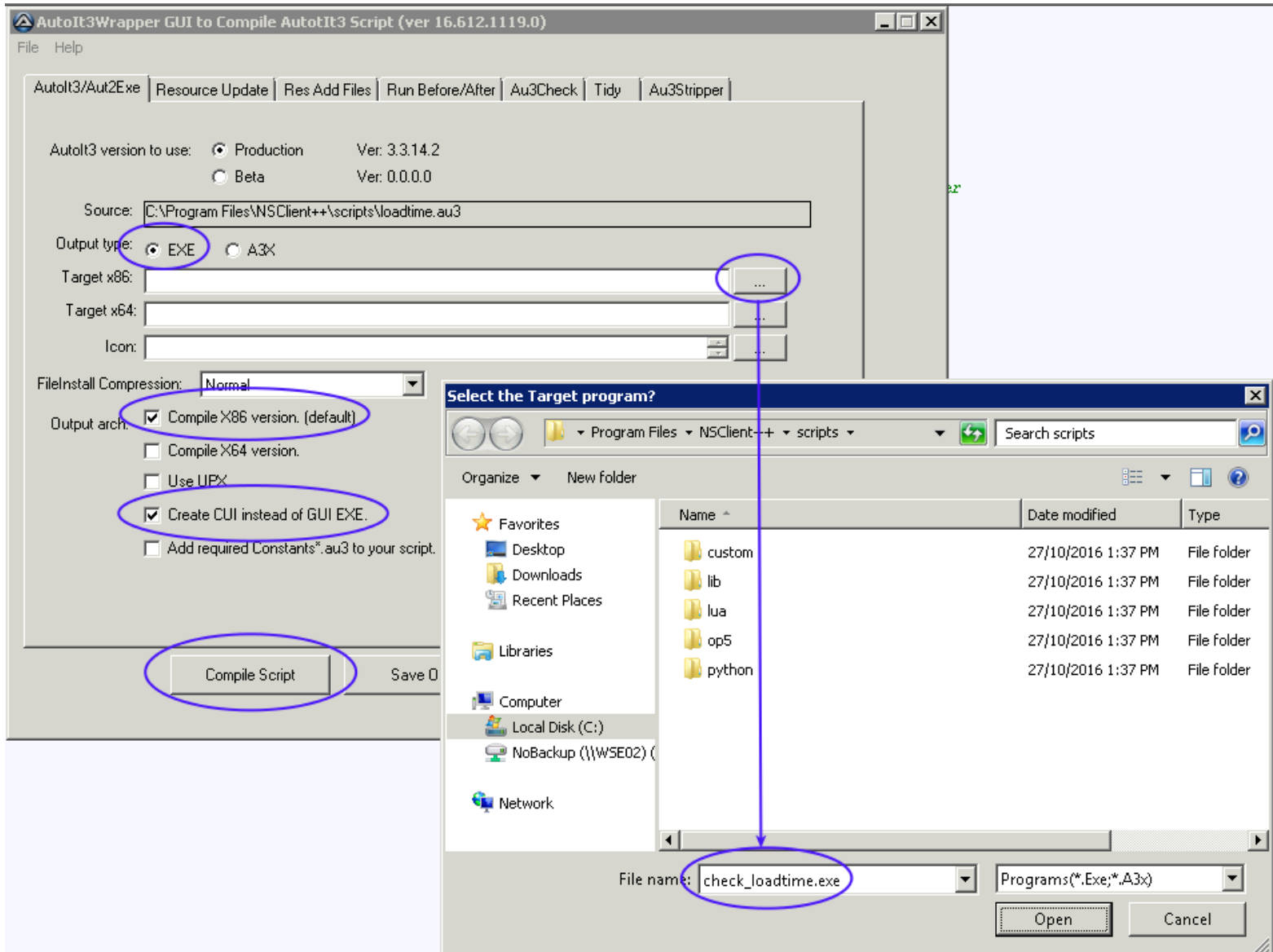
Once it detects the string, the time it took is calculated (line 11) and the time difference is stored in the variable `$time_string` (line 12). Finally the script outputs the variable `$time_string` to the console.

You may want to save this script to the location `C:\Program Files\NSClient++\scripts\` directory, this example saves the file as `loadtime.au3`. Saving the script is not a necessary part of the process, but you may want to come back to it later if you want to re-compile.

The next step is to compile this script into an EXE file, this is necessary so that the output is correctly written to the console.

Compile Script

- To compile the script, click the pull down menu **Tools** and select **Compile**.
 - For the **Target x86** field click the ... button
 - Browse to `C:\Program Files\NSClient++\scripts\`
 - Give it the name `check_loadtime.exe`
 - Click **Open**
 - Compile X86 version = **Ticked**
 - Create CUI instead of GUI.EXE = **Ticked**
 - Click the **Compile Script** button to create `check_loadtime.exe`

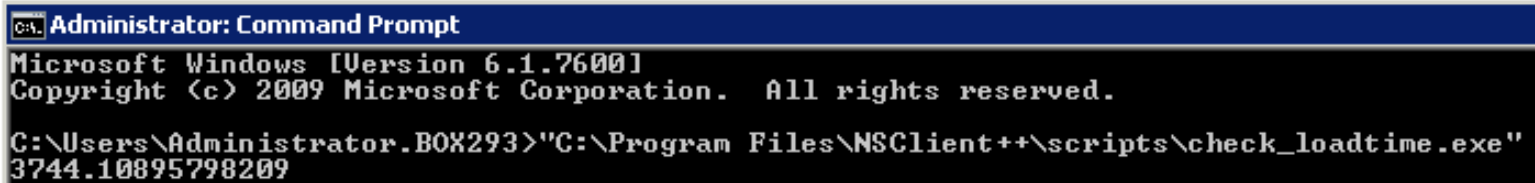


A dialogue window will appear while the script is compiled. When the compile finishes you will be returned to the SciTE Editor.

At this point we have a simple Autolt script created. You can test it by opening a command prompt and typing the following command:

```
"C:\Program Files\NSClient++\scripts\check_loadtime.exe"
```

You should not move the mouse or type on the keyboard until all the actions have been performed and you see the console output like the following:



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator.BOX293>"C:\Program Files\NSClient++\scripts\check_loadtime.exe"
3744.10895798209
```

If you get the number output as per the screenshot then it's working as expected.

The next step will be to configure NSClient++ to execute `check_loadtime.exe` so that Nagios XI server can execute it.

Configure NSClient++

You must now edit the NSClient++ configuration file, open the file `C:\Program Files\NSClient++\nsclient.ini` in Notepad.

Locate the External Scripts section. If you are already using NRPE checks there should be commands listed here, if not simply add this directly below it:

```
[/settings/external_scripts/scripts]
check_loadtime = scripts\check_loadtime.exe
```

Once this has been added save the `nsclient.ini` file and restart NSClient++ (the same way you did in earlier in the section [Restart NSClient++ Service](#)).

This is all the configuration that is required on the Windows machine. Next we will combine this check on the windows side with a plugin called `check_autoit_timer`. This plugin will take the data the Autolt script outputs and turn it into something readable as well as being able to trigger warning and critical thresholds.

Add A Plugin To Nagios

In order to automate the Autolt script and monitor its output with Nagios, you will first need to upload a plugin in Nagios XI.

A `check_autoit_timer.sh` plugin that can work with the example Autolt script in this document can be downloaded from Nagios Exchange:

<https://exchange.nagios.org/directory/Plugins/Operating-Systems/Windows/NRPE/autoIT-Timer-plugin/details>

Once you've downloaded the plugin open Nagios XI and navigate to **Admin > System Extensions > Manage Plugins**.

Upload the plugin using the **Browse** and **Upload Plugin** buttons.

Creating The Check In Nagios XI

Now the check must be configured in the Nagios XI Web Interface using Core Configuration Manager (CCM). The first step will be to create a custom command specifically for this check.

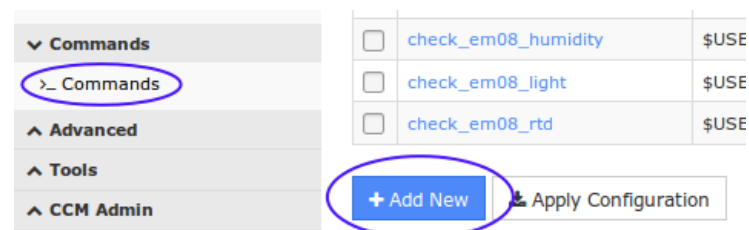
Create Check Command

Navigate to **Configure > Core Config Manager (CCM)**

In the left pane expand **Commands** and then click

>_ Commands

Click the **Add New** button



The Command Management page will open. Populate the fields with the following values:

Command Name:

```
check_autoit_timer
```

Command Line:

```
$USER1$/check_autoit_timer.sh -H $HOSTADDRESS$ -p 5666 -c "$ARG1$" -w $ARG2$ -c $ARG3$
```

Command Type:

```
check command
```

Active:

checked

Click the **Save** button to create this new command.

Command Management

Command Name *

Example: check_example

Command Line *

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

Command Type:

Active ?

Available Plugins

 ?

Create Service

The final step is to create a new service definition that is associated with the remote windows host. It is assumed that you are already monitoring the Windows host and there is a HOST object already created. If there isn't, go and run the **Windows Server Configuration Wizard** and then return to this step. This guide is going to use the host `10.25.14.52` as an example.

In the left pane expand **Monitoring** and then click **Services**.

Click the **Add New** button.

The screenshot shows the Nagios XI Core Config Manager interface. The left sidebar has 'Monitoring' expanded, and 'Services' is selected. The main area shows a table of services for configuration 10.25.14.52. The 'Add New' button is circled in blue.

<input type="checkbox"/>	Service Name	Service Description
<input type="checkbox"/>	10.25.14.52	CPU Usage
<input type="checkbox"/>	10.25.14.52	Drive C: Disk Usage
<input type="checkbox"/>	10.25.14.52	Drive D: Disk Usage
<input type="checkbox"/>	10.25.14.52	Memory Usage
<input type="checkbox"/>	10.25.14.52	Uptime

Common Settings tab

Config Name:

10.25.14.52

Description:

AutoIt Timer

Click the **Manage Hosts** button

Select 10.25.14.52 in the left pane and click the **Add Selected >** button

Click the **Close** button

For this service we will use the **generic-service** template as it has a lot of the required directives already configured

Click the **Manage Templates** button

Select **generic-service** in the left pane and click the **Add Selected >** button

Click the **Close** button

Check command (drop down list)

check_autoit_timer

\$ARG1\$:

check_loadtime

\$ARG2\$:

2000

This is the warning threshold (in milliseconds), if the number returned from the AutoIt script exceeds this number then the service will go into a WARNING state.

\$ARG3\$:

5000

This is the critical threshold (in milliseconds), if the number returned from the AutoIt script exceeds this number then the service will go into a CRITICAL state.

Active:

Checked

Service Management

Common Settings
✓ Check Settings
Alert Settings
Misc Settings

Config Name *

Description *

Display name

Check command

Command view

```
$USER1$/check_autoit_timer.sh -H $HOSTADDRESS$ -p 5666 -c "$ARG1$" -w $ARG2$ -c $ARG3$
```

Manage Hosts 1

Manage Templates 1

\$ARG1\$

\$ARG2\$

\$ARG3\$

Check Settings tab

Check interval:

5

Retry interval:

1

Max check attempts:

3

Click the **Save** button and then click the **Apply Configuration** button at the bottom of the screen.

Service Management

Common Settings
✓ Check Settings
Alert Settings

Initial state

Warning
Critical
Ok
Unreachable

Check interval

5
min

Retry interval




1
min

Max check attempts

3
attempts

End Result

Now that the service has been created, navigate to **Home > Service Detail** and search for the service. If the check has been configured correctly, you should see a result like the one below.

Host	Service	Status	Duration	Attempt	Last Check	Status Information
10.25.14.52   	Autolt Timer	Warning	23s	1/3	2016-11-01 12:36:01	WARNING: Script took 4.2690 seconds to complete: time=4269ms:2000:5000:0

From the Warning status you can see that the Autolt script is taking more than 2000 milliseconds to load the test webpage.

Finishing Up

This completes the documentation on how to integrate Autolt with Nagios XI.

If you have additional questions or other support related questions, please visit us at our Nagios Support Forums:

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

The Nagios Support Knowledgebase is also a great support resource:

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