WMI Agentless Plugins Project

GroundWork Open Source Projects

Windows Management Instrumentation (WMI)

Windows Management Instrumentation (WMI) is a management standard technology for accessing management information and automating administrative tasks in an enterprise environment. There are two main systems management architectures; Agent-based where the proprietary software agent is loaded on a managed system and, Agent-less which depends on management functionality that is built into a managed system. See Microsoft Developer Network (MSDN) for a detailed overview of WMI.

GroundWork WMI Agentless Plugins Project

This project consists of a collection of script monitors (.vbs for starters) that use the Microsoft .Net Framework and WMI to retrieve performance data from remote Windows hosts without the need for agents on the remote hosts. Initially we have about 20 scripts, although it is a fairly minor matter to script others.

To support the extra WMI checks of system uptime you can find the plugin called get_system_uptime.vbs which has been added to this project in the GroundWork knowledge base here [DOCDEV:WMI].

Typical Monitor

A typical monitor works like this:

```
cscript //nologo check_cpu.vbs -h hostname -w -c -user username -pass password
```

This script returns a string to stand out that says something like:

```
OK - CPU Utilization 67%" or "WARNING - CPU Utilization 89%" or "CRITICAL - CPU Utilization 98%
```

The warning and critical thresholds will be passed as command line arguments (in percent).

In addition it returns an exit value like this:

```
0 = OK 1 = Warning 2 = Critical 3 = Error
```

All scripts return syntax and help if passed the --help command line option. And all scripts return performance data formatted according to the Nagios Plugin Developer's Guidelines.

Zip Archive with nrpe_nt and .vbs Monitors

These monitors implement the agentless WMI monitors for Windows.

1. Install this directory on your Windows proxy server under `root` (i.e. `C:\nrpe_nt`).
2. Change directories (`cd`) into this directory.
3. Enter `nrpe_nt -i` to install the nrpe_nt agent as a service on the proxy server.
4. You will need to edit the `nrpe.cfg` file in that directory and change the IP of the allowed hosts to match your Nagios server IP. This allows Nagios to access the `nrpe_nt` service.
5. On the `nagios` end you will need some command definitions for your `check_commands.cfg` file.
6. Once you have `nrpe_nt` up and running and the check command definitions in you can start setting up your service templates and services.
7. Each of the plugins can be executed from the command line using the `--help` option which gives you usage. Get them all working from the proxy server command prompt first, then move to the Nagios Server and get them working from the command line there - using the syntax outlined below.

Notes

Please note that these command definitions and the actual commands as defined in nrpe.cfg in the config file might not line up exactly. They came from different places.

Also note that in the nrpe.cfg file we have two versions of each monitor. One uses authentication while the other assumes that the proxy server

has the same Administrator password as the monitored servers. In that case no authentication strings are passed. The checkcommand definitions above all use authentication (therefore the `_auth` on the end of the script names called on the proxy server).

The last command line $ARGx$ argument passed in the below definitions is the username (typically supplied in the service template). The final argument $USER9$ is the password which is in the nagios resources.cfg file where $USER9$ is defined as the password. Around here we keep that file (and the passwords, community strings etc. in a secured subdirectory).

Example

There should be no carriage returns in the actual command_line (the IP address is the address of the Proxy Server):

```
# nagios-WMI-cpu check command definition
define command {
command_name check_WMI_cpu
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_cpu_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-disk check command definition
define command {
command_name check_WMI_disk
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_disk_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $ARG4$ $USER9$
}

# nagios-WMI-disks check command definition
define command {
command_name check_WMI_disks
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_disk_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-mem check command definition
define command {
command_name check_WMI_mem command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_mem_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# commands check_WMI_printque
define command {
command_name check_WMI_printque
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_printque_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-proc check command definition
define command {
command_name check_WMI_procs
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_process_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $USER9$
}

# commands check_WMI_swap
define command {
command_name check_WMI_swap
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_swap_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-exchange-mailbox-receiveq check command definition
define command {
command_name check_WMI_exchange_mailbox_receiveq
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_exc_mbx_rcvq_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-exchange-mailbox-sendq check command definition
define command {
command_name check_WMI_exchange_mailbox_sendq
```

```
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_exc_mbx_sendq_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# commands check_WMI_exchange_mta_workq
define command {
command_name check_WMI_exchange_mta_workq
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_exc_mta_workq_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# commands check_WMI_exchange_public_recieveq
define command {
command_name check_WMI_exchange_public_receiveq
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_exc_pub_rcvq_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# commands check_WMI_exchange_public_sendq
define command {
command_name check_WMI_exchange_public_sendq
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_exc_pub_sendq_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-mssql-buffer-cache-hit check command definition
define command {
command_name check_WMI_mssql_buff_cache_hit
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_mssql_buf_cache_hit_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-mssql-latch-waits check command definition
define command {
command_name check_WMI_mssql_latch_waits
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_mssql_latch_waits_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-mssql-lock-wait-time check command definition
define command {
command_name check_WMI_mssql_lock_wait_time
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_mssql_lock_wait_time_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $USER9$
}

# nagios-WMI-mssql-log-growth check command definition
define command {
command_name check_WMI_mssql_log_growth
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_mssql_log_growth_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $ARG4$ $USER9$
}

# nagios-WMI-mssql-log-used check command definition
define command {
command_name check_WMI_mssql_log_used
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_mssql_log_used_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $ARG4$ $USER9$
}

# nagios-WMI-mssql-transactions check command definition
define command { command_name check_WMI_mssql_transactions
```

```
command_line $USER1$/check_nrpe -H 192.168.1.243 -c check_mssql_transactions_auth -a
$HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $ARG4$ $USER9$
}
```