WMI

Overview

This page reviews how to use WMI agentless plugins and is valid for use with any version of GroundWork Monitor. Further, those customers using GroundWork Distributed Monitor Agent (GDMA) may find the description of plugins and WMI troubleshooting sections useful.

CONTENTS

RELATED RESOURCES

- WMI troubleshooting
- Microsoft Developer Network (MSDN)
- Nagios Plugin Developer's Guidelines

WAS THIS PAGE HELPFUL?

- Leave Feedback

1.0 About WMI

WMI Agentless Plugins Project - 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. **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.

There are large numbers of WMI Classes on Windows hosts. This plugin package with a small change to each plugin can be used to retrieve almost any parameter of interest in the default WMI namespace. There is no need for agents on the remote hosts.

The plugin package comes with an NRPE configuration file that can be included from the base `nrpe.cfg` file. Some of the plugins retrieve specific properties such as CPU load percentage, disk utilization, disk and network I/O, etc., while others are for retrieving arbitrary properties from WMI. The provided NRPE configuration file defines NRPE commands using the plugins to get both specific and arbitrary properties and there are documented syntax examples for each command.

Plugins are configured to return performance data where it makes sense which facilitates easy graphing of results on a Nagios server.

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.

Description of plugins

The following table lists and describes plugins in the package.

Plugin	Description
check_100nsec_timer.vbs	checks WMI counters of type 'PERF_100NSEC_TIMER'
check_counter_bulk_count.vbs	checks WMI counters of type 'PERF_COUNTER_BULK_COUNT'
check_counter_counter.vbs	checks WMI counters of type 'PERF_COUNTER_COUNTER'
check_counter_large_rawcount.vbs	checks WMI counters of type 'PERF_COUNTER_LARGE_RAWCOUNT'
check_counter_rawcount.vbs	checks WMI counters of type 'PERF_COUNTER_RAWCOUNT'
check_cpu_load_percentage.vbs	checks load percentage of one or more CPUs
check_disks_io.vbs	checks disk I/O of one or more logical disks
check_disks_percentage_space_used.vbs	checks disk usage of one or more logical disks
check_memory_percentage_space_used.vbs	checks RAM, page file, or total memory usage
check_network_io.vbs	checks network I/O of one or more TCP/IP network interfaces
check_proc_num.vbs	checks number of running processes matching a search expression
check_raw_fraction.vbs	checks WMI counters of type 'PERF_RAW_FRACTION'
check_services_states.vbs	checks the state of an installed service
get_computer_info.vbs	enumerates information about processors, installed services, running processes, network interfaces, and logical disks for use in configuring other plugins
get_counter_type.vbs	enumerates information about WMI classes and properties to determine which counter plugin to use
get_system_uptime.vbs	checks system uptime and enumerations information about the hardware and OS
verify_wmi_status.vbs	checks OS version to verify WMI is working

2.0 Installation

Prerequisites

1. NRPE_NT version 0.8b. The remainder of this document assumes that `NRPE_NT` has been installed under `C:\NRPE_NT` with the executable and configuration file under `C:\NRPE_NT\bin`. It also assumes that you have tested correct operation of `NRPE_NT` by calling it with the `check_nrpe` Nagios plugin from your Nagios server. Version 0.8b is necessary in order to fully support specification of some special characters when calling `NRPE_NT` from `check_nrpe`. `NRPE_NT` can be downloaded from Sourceforge.
2. Nagios 2.x or later with `check_nrpe` plugin compatible with `NRPE_NT` version 0.8b. Nagios can be found at www.nagios.org.

Installing

1. Once you have installed `NRPE_NT` on a host (designated as `<host>`) and tested that it can be called from your Nagios server with a `check_nrpe -H <host>` create the following directories:

```
C:\NRPE_NT\Plugins
C:\NRPE_NT\Plugins\V2
```

2. Copy all of the VBS programs from this directory into:

```
C:\NRPE_NT\Plugins\V2
```

3. Copy the file `V2_nrpe_commands.cfg` into:

```
C:\NRPE_NT\bin
```

4. Edit `C:\NRPE_NT\bin\nrpe.cfg` and append the following line to the bottom:

```
include=C:\NRPE_NT\bin\V2_nrpe_commands.cfg
```

5. Restart the NRPE_NT service.

Testing

After completing the installation procedure you should be able to call each of the installed scripts with the commands defined in `V2_nrpe_commands.cfg` either from the Windows server itself or from the Nagios server using `check_nrpe`.

1. On the Windows server you installed `NRPE_NT` and the plugins on, open a command window and type:

   ```
   cscript //nologo C:\NRPE_NT\Plugins\V2\verify_wmi_status.vbs -h 127.0.0.1
   ```

   The output you get should look something like:

   ```
   OK - Microsoft Windows XP Professional, SP 2.0
   ```

   This indicates that the plugin was able to successfully talk with the WMI service on 127.0.0.1 and retrieve the OS version.

2. On the Nagios server, type the following command in a shell once you have changed directory to the location of the `check_nrpe` plugin:

   ```
   ./check_nrpe -H <host> -c show_os -a 127.0.0.1
   ```

   The output you get should look something like:

   ```
   OK - Microsoft Windows XP Professional, SP 2.0
   ```

   You have just executed the same `verify_wmi_status.vbs` plugin but through the `NRPE_NT` service from your Nagios server.

Examples

The following example Nagios command and service definitions assume you have Nagios 2.x or later installed with a version of `check_nrpe` compatible with `NRPE_NT` v0.8b. They also assume that `$USER1$` points to the directory containing `check_nrpe`, and that `$USER21$` is the address of the host on which `NRPE_NT` is installed! Download information about both Nagios 2.x (or later) and NRPE_NT v0.8b is available in the install document referenced below.

```
define command {
  command_name check_wmi_mssql_transactions
  command_line $USER1$/check_nrpe -t 60 -H $USER21$ -c \\
    get_mssql_transactions -a "$HOSTADDRESS$" "$ARG1$" \\
    "$ARG2$" "$ARG3$"
}

define service {
  name wmi_mssql_transactions
  check_command check_wmi_mssql_transactions!Name=_Total!10!20
  #.
  #. other parameters as desired for service definition
  #
}
```

```
define command {
  command_name check_wmi_cpu
  command_line $USER1$/check_nrpe -t 60 -H "$USER21$" -c \\
    get_cpu -a "$HOSTADDRESS$" "$ARG1$" "$ARG2$"
}

define service {
  name wmi_cpu
  check_command check_wmi_cpu!_Total!80,90
  #.
  #. other parameters as desired for service definition
  #.
}
```

```
define command {
  command_name check_wmi_mem
  command_line $USER1$/check_nrpe -t 60 -H $USER21$ -c \\
    get_mem -a "$HOSTADDRESS$" "$ARG1$" "$ARG2$"
}

define service {
  name wmi_mem_ram
  check_command check_wmi_mem!RAM!80,90
  #.
  #. other parameters as desired for service definition
  #.
}
```

```
define command {
  command_name check_wmi_net_io
  command_line $USER1$/check_nrpe -t 60 -H $USER21$ -c \\
    get_netio -a "$HOSTADDRESS$" "$ARG1$" "$ARG2$" "$ARG3$"
}

define service {
  name wmi_net_io_loopback
  check_command check_wmi_net_io!"MS TCP Loopback Interface" \\
    !BytesReceivedPerSec,BytesSentPerSec!300,1000
  #.
  #. other parameters as desired for service definition
  #.
}
```

For a list of further examples refer to the NRPE_NT configuration file available for download below. One key point is that you can reverse warning and critical thresholds and have match in the reverse sense, (e.g., If you set -w 80 -c 50 and the returned value is 90 the return code will be 0 (OK)).

Downloads

Name	Size	Creator	Creation Date	Comment
wmi-1.4.zip	129 kB	NotSupportContact-Mark Carey	Jul 15, 2011 07:29	
gwnrpewmi-1.4.zip	1.46 MB	Groundwork Support	Aug 30, 2011 13:31	