How NMIS interfaces with WMI-based devices

Overview

In NMIS version 8.6.0 we've added support for collecting data from Windows systems using the Windows Management Instrumentation infrastructure (or short WMI).
This this page describes how to approach modelling devices for WMI, and where WMI modelling differs from modelling for SNMP.

Prerequisites

Tools

To collect WMI data NMIS has to use a WMI access tool. As of NMIS 8.6.0 we are using `wmic`, a commandline tool belonging to the Samba software suite. NMIS 8.6.0 ships with a precompiled `wmic` program, and installs it as `/usr/local/nmis8/bin/wmic`. If the precompiled version should not work on your platform, the installer will notify you of that problem and you'll have to perform a manual build of `wmic`. The sources for `wmic` can be downloaded here: http://dl-nmis.opmantek.com/wmic-omk.tgz and you shouldn't have to do more than unpack that, and run `make`. When the build is complete, you should copy the resulting `wmic` file to `/usr/local/nmis8/bin/`.

WMI Access

On the target systems, the WMI service must be running and the network (and any firewalls) must be configured to let WMI accesses pass. WMI accesses are generally negotiated to use dynamic ports (following up on an initial conversation on TCP port 135), but Microsoft provides instructions on how to setup fixed ports for WMI.

Node Configuration

NMIS does not attempt any WMI accesses unless the node in question is configured with both a `wmiusername` and a `wmipassword` property. This can be done in the GUI, under "Edit Node'; the WMI options are shown prominently near the top of the page.

If the node in question requires a windows domain for the WMI access, then prepend that to the `wmiusername` followed by a "/", e.g. "`somedomain /theuser`".
In NMIS 8.6.7 and newer you can also provide the domain in the form "`theuser@somedomain`".

Automatic model selection does include WMI as a source of information, if SNMP is not available and if `wmiusername` and `wmipassword` are set.

Modelling Preparation

We recommend that you verify the availability of WMI (and your credentials) with `wmic`, before performing any modelling work. This should be done using the `wmic` tool on your NMIS server, like in the following example:

```
$ /usr/local/nmis8/bin/wmic -U somewmiuser --password='somewmipassword' //testserver "select Caption,
Manufacturer,Model,Name from Win32_ComputerSystem"

CLASS: Win32_ComputerSystem
Caption|Manufacturer|Model|Name
TestServer|VMware, Inc.|VMware Virtual Platform|TestServer
```

If WMI is properly configured and the access details match you'll see output similar to the three lines shown.

Besides using the standard Widows models that NMIS ships with as examples, you will likely also need to consult the online WMI reference documentation for determining what is available in the WMI universe where and how to tell NMIS about it.

WMI Modelling in a nutshell

Let's examine an example model:

```
'system' => {
  ...lots of stuff...
  'sys' => {
    'standard' => {
      'snmp' => {
      ....lots of stuff...
      },
      'wmi' => {
        'bios' => {
          title  => "Bios Name",
          query => 'select name from win32_bios',
          field => "Name",
          calculate => '$r =~ s/\s*$//; return $r;',
        },
        # if we want to type less, we can set a shared query - not required, though!
        "-common-" => {
          query => 'select * from win32_pagefileusage'
        },
        'totalswap' => {
          title => "total swap in bytes",
          query => 'select allocatedbasesize from Win32_pagefileusage',
          field => 'AllocatedBaseSize',
          calculate => 'return $r*(1<<20);',
        },
```

Here are the crucial aspects:

1. Wherever an `snmp` section is allowed in a model, you may add a `wmi section`.
2. A model may have either or both `snmp` and `wmi` sections, but the collected variables must be uniquely named.
3. Just like SNMP sections, a WMI section consists of any number of variable collection definitions.
4. A WMI section may also contain a section called `-common-`, which specifies a shared `query` property for variables without explicit `query`.
5. A WMI variable definition must have a `query` (or inherit one from `-common-`) and a `field` declaration.
    a. The `query` is issued to the host in question using the `wmic` tool, and must *at least* select the field/column you're interested in.
    For efficiency you should use the same combined query for as many variables in your section as possible; i.e. `select * from someclass`.
    If you have multiple variables in your section and set the same query argument for all of them, then NMIS will issue the query just once and reuse the results.
    The same goes if you use the `-common-` mechanism as shown in the example above, in which case you don't have to give your variable section an explicit `query` property.
    b. The `field` is used to select the column or property from the query result.
    The field is case-sensitive! (The `select` attributes are generally not.)
6. All other NMIS modelling mechanisms work the same, i.e. `control`, `replace`, `nosave` etc.

Here is another example, this time of an indexed `systemHealth` section:

```
'systemHealth' => {
   'sys' => {
     'WindowsPagefile' => {
       'headers' => 'Name,pageTotal,pageUsage',
       'indexed' => 'Name',
       'wmi' => {
         "-common-" => {
           'query' => 'select * from win32_pagefileusage',
         },
         'Name' => {
           'field' => 'Name',
           'title' => 'Name'
         },
   ...lots of other stuff
   'rrd' => {
     'WindowsPagefile' => {
       'graphtype' => 'WindowsPaging',
       'threshold' => 'WindowsPaging',
       'indexed' => 'Name',
       'wmi' => {
         'pageUsage' => {
           'query' => 'select * from win32_pagefileusage',
           'field' => 'CurrentUsage',
           'calculate' => 'return $r*(1<<20);',
         },
         'pageTotal' => {
           'query' => 'select * from win32_pagefileusage',
           'field' => 'AllocatedBaseSize',
           'calculate' => 'return $r*(1<<20);',
         },
```

This example illustrates one more crucial aspect:

1. If your WMI-sourced variables are indexed (i.e. belong to a table with multiple instances), then you ***must*** set the `indexed` model property to the name of the variable.
   And, of course, there must be a variable section for the given variable to index by (in the example above, it's called `Name` and indicates the name of the page/swap file).
2. If you are collecting such variables both for current display (`sys` section) and long-term collection (`rrd` section), then ***both*** sections must contain the same indexed property, not just 'true'.
   In an SNMP section under `rrd`, 'true' is sufficient because the property to index by can be deduced in that case. For WMI this doesn't hold.

WMI Modelling Limitations

As of version 8.6.0 there are a few modelling limitations that we plan to remedy incrementally.

- It is not possible for a `systemHealth` section to have both `snmp` and `wmi` sections.
  This is because only one index per `systemHealth` section is supported, but `wmi` and `snmp` can not share that single index.
- At this time, collection of the following types of statistics from WMI is not supported:
  Network Interfaces
  Environment Data
  CBQoS Data
  Calls
  Server-type processor and load information
- NMIS does not yet support service tests for WMI-sourced process information.
- Collection of indexed WMI sections is not optimised for maximum efficiency yet.
  query results are reused to some extent but not universally, and further optimisations are planned.
- The GUI model editor does not support editing of WMI sections yet.