Using Scriptomatic to create custom monitors

Contents

- Download Scriptomatic and locate the desired metric(s)
- Allow Remote WMI Requests
- Output Format
- Create and import the monitor UI
- Additional Step for Windows Monitoring Stations

Download Scriptomatic and locate the desired metric(s)

Scriptomatic is a Microsoft utility that allows you to create scripts that access the WMI counters built into Windows. You can get Scriptomatic from the Microsoft Download Center.

The following screenshot shows the main Scriptomatic interface:

blocked URL

After selecting a WMI Namespace and Class, Scriptomatic will automatically generate a script to retrieve the associated values. Review the list of namespaces and classes to locate the metric(s) that you are interested in. Note that you may need to contact Microsoft or your systems vendor to determine where the counters are located (Google also helps). An example of system memory information is provided below:

blocked URL

You can also test the script by clicking the Run button, which will bring up a command line window with the following output:

blocked URL

Format the script output

Once you have a script that successfully retrieves the required values, some changes will be needed to enable it to work with Uptime Infrastructure Monitor.

Allow Remote WMI Requests

The initial script should work for the local system but it may not work when retrieving values for your remote servers. The script can be placed on all the remote systems but this complicates the installation and setup process and also presents scaling issues. Making the small modifications described below will allow the script to retrieve metrics from your remote servers with a properly authenticated method.

More detailed information can be found in Connecting to WMI on a Remote Computer on the MSDN website.

Example script from Scriptomatic:

```
    On Error Resume Next

Const wbemFlagReturnImmediately = &h10
Const wbemFlagForwardOnly = &h20

arrComputers = Array("PSO-JPEREIRA2")
For Each strComputer In arrComputers
    WScript.Echo
    WScript.Echo "==========================================="
    WScript.Echo "Computer: " & strComputer
    WScript.Echo "==========================================="

    Set objWMIService = GetObject("winmgmts:" & strComputer & "rootCIMV2")
    Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_PerfRawData_PerfOS_Memory", "WQL", _
        wbemFlagReturnImmediately + wbemFlagForwardOnly)

    For Each objItem In colItems
        WScript.Echo "AvailableBytes: " & objItem.AvailableBytes
        WScript.Echo "AvailableKBytes: " & objItem.AvailableKBytes
        WScript.Echo "AvailableMBytes: " & objItem.AvailableMBytes
...
```

The green code in the above example is the WMI Namespace and Class. Replace the namespace and class in the next section with the original values from above. The red text in the above example should be replaced with the following section to enable remote WMI requests (remember to use the original WMI namespace and class from the original code):

```
    strComputer = WshSysEnv("UPTIME_HOSTNAME")
strUser = WshSysEnv("UPTIME_USERNAME")
strPassword = WshSysEnv("UPTIME_PASSWORD")

Set objSWbemLocator = CreateObject("WbemScripting.SWbemLocator")
Set objSWbemServices = objSWbemLocator.ConnectServer(strComputer, _
    "rootCIMV2", _
    strUser, _
    strPassword, _
    "MS_409")
objSWbemServices.Security_.ImpersonationLevel = 3
Set colItems = objSWbemServices.ExecQuery("SELECT * FROM Win32_PerfRawData_PerfOS_Memory ", _
    "WQL", wbemFlagReturnImmediately + wbemFlagForwardOnly)
```

Because one of the “for loops” was removed, the outer “Next” will also need to be removed at the bottom of the script. Confirm that the script works properly before continuing to the next step.

For example:

```
    ...
WScript.Echo "UserName: " & objItem.UserName
        WScript.Echo
    Next
Next

Function WMIDateStringToDate(dtmDate)
WScript.Echo dtm:
    WMIDateStringToDate = CDate(Mid(dtmDate, 5, 2) & "/" & _
    Mid(dtmDate, 7, 2) & "/" & Left(dtmDate, 4) _
    & " " & Mid (dtmDate, 9, 2) & ":" & Mid(dtmDate, 11, 2) & ":"
& Mid(dtmDate,13, 2))
End Function
```

Output Format

The Uptime Infrastructure Monitor monitoring station expects to see data in the following format:

variable1 value1
variable2 value2

Clean up the script by removing the colon (:) from each of the variables that you want to report. For the list of metrics that aren't required, simply delete the lines in the script.

For example, change the script from:

```
    WScript.Echo "AvailableBytes: " & objItem.AvailableBytes
WScript.Echo "AvailableKBytes: " & objItem.AvailableKBytes
WScript.Echo "AvailableMBytes: " & objItem.AvailableMBytes
```

To:

```
    WScript.Echo "AvailableBytes " & objItem.AvailableBytes
WScript.Echo "AvailableKBytes " & objItem.AvailableKBytes
WScript.Echo "AvailableMBytes " & objItem.AvailableMBytes
```

Create and import the monitor UI

The monitor interface can be generated with the uptime Plug-In Monitor XML Generator. Please refer to Creating the XML definition for your plug-in service monitor for further assistance.

The final monitor can be imported by placing the XML file in the Uptime Infrastructure Monitor directory and running the following commands on the command line:

> cd <uptime_dir>
> scripts\erdcloader -x <xml_filename>

Additional Step for Windows Monitoring Stations

If the Uptime Infrastructure Monitor monitoring station is running on a Windows platform, the following steps will need to be implemented to allow the VB script to run properly.

Create a Windows batch file that contains the following command:

```
    cscript "C:Program Files uptime softwareuptimescripts.vbs"
%UPTIME_HOSTNAME% %UPTIME_USERNAME% %UPTIME_PASSWORD% //nologo
Edit the VBS script file created at the end of step 2 and change the following lines from:
strComputer = WshSysEnv("UPTIME_HOSTNAME")
strUser = WshSysEnv("UPTIME_USERNAME")
strPassword = WshSysEnv("UPTIME_PASSWORD")
```

To:

```
    strComputer = Wscript.Arguments.Item(0)
strUser = Wscript.Arguments.Item(0)
strPassword = Wscript.Arguments.Item(0)
```

Next, edit the 'Script Name' field of the plugin monitor to point to this batch script. For example: "C:\Program Files\uptime software\uptime\scripts\<plugin batch script name>.bat"

After this step is completed, the plugin monitor should run the Wwindows batch script, which then runs cscript to execute the VB script.