PANDORAFMS
ENTERPRISE

Pandora FMS
Administrator's Manual
PerfCounter Monitoring

Administrator's Manual PerfCounter Monitoring

© Artica Soluciones Tecnológicas 2005-2012

Index

1 Changelog

Date	Author	Change	Version
02/08/12	Tomas	First Version	v1r1

2 Introduction

This document has as main objective the description of the massive monitoring of performance counters in Windows environments with an integrated SQL service.

To extract the information, it uses:

- Powershell 2.0 Console (installed by default in Windows Server 2008 R2, and Windows 7. Available from Windows XP SP2 forward)
- An “open” interface (Pandora's as extension of the administration section) to specify free SQL queries.
- The system, that is integrated with the Windows agent.

It is important to say that the plugin of Performance Counters monitoring could be used to collect information of numerical kind (to do performance management)

3 Requirements

The requirements in order this monitoring could work properly are the following:

- To install the Pandora FMS agent in the 3.2.1 version or higher.
- A Powershell 2.0 console to execute the plugin. By default is comes installed in Windows Server 2008 R2 and Windows 7 systems, but it should be downloaded for Windows previous versions. Powershell is not compatible with Windows XP SP1 or lower system versions.
- It is necessary that the user with which the Pandora FMS agent is executed, that is the user that will execute the plugin, has available the following system permissions:
  - Local administrator.
- The Powershell scripts execution policy should be fixed as RemoteSigned or lower.

```
Set-ExecutionPolicy RemoteSigned
```

- The plugin will automatically get the information about all the counters that we have specified to it in a list in the *counters.txt file* and it will generate one module for each one in Pandora (Plugin PerfCounter).

4 Compatibility Matrix

The agent compatibility matrix is the following:

Systems where it has been tested	• Windows XP SP2 • Windows Server 2003 • Windows Server 2008 • Windows Vista • Windows 7

Systems where it should work	• Same system of higher

Depending on the language of the system, the format of the counters to monitor could change, so it would be necessary to adapt the counters.txt file according to those circumstances.

5 Software Agent Modules generated

The verification of performance counters is done through the Pandora_Plugin_PerfCounter.ps1 plugin and could be applied in different policies for different technologies, each of them with different counter lists depending on which we want to monitor in each of those technologies.

Monitor technologies should be included in the file counters.txt or if we don't use it would load the default Get-Service counters.

6 Installing

Copy the plugin to the agent plugin directory, distribute it through file collections. Do the same with the counter list.The call from the agent will be similar to this, but using the paths where the plugin and the list would be installed. For example:

```
module_plugin "<ruta-powershell>\powershell.exe" -command C:\'<ruta-
plugin>\Pandora_Plugin_PerfCounter_vx.y.ps1' -list C:\'<ruta-listado>\counters.txt'
2> counter_plugin.error
```

If the call that we use is:

```
module_plugin "<ruta-powershell>\powershell.exe" -command C:\'<ruta-
plugin>\Pandora_Plugin_PerfCounter_vx.y.ps1' -list none 2> counter_plugin.error
```

Then the plugin will return in XML format the information extracted from the counters that the the Powershell cmdlet Get-Counter provides by default.

7 Monitoring

Considering that we have already installed and configured both Pandora and the Powershell, we are going to explain how to get information about the status of the hard disk installed in the machines to monitor

In this case we are going to install both one Panodra software agent and aslos the PerfCounter plugin in this machine.

Is very important that the Powershell scripts execution policy should be fixed as RemoteSigned or lower in the following way:

```
Set-ExecutionPolicy RemoteSigned
```

To do that the Pandora software agent that we have installed in our server to monitor execute that script, we should edit the agent configuration file and do the call to the plugin through the **module_plugin** configuration token

```
module_plugin "<ruta-powershell>\powershell.exe" -command C:\'<ruta-
plugin>\Pandora_Plugin_PerfCounter_vx.y.ps1' -list C:\'<ruta-listado>\counters.txt'
2> counter_plugin.error
```

Save the file and restart the Pandora agent.

To avoid incrementing the log without control and keep receiving all the errors that had place when executing the plugin in the last interval (just in case there is a real error), do the readressing using the symbol "2>" as it comes specified in the line that you should introduce in the configuration file.

One of the most powerful characteristics of the plugin in Powershell, is the possibility of specifying instead of create one by one modules for each performance counter, select all the counters that are specified in one list, as the plugin will do a single check and generate automatically one module for all these counters, optimizing at maximum the time that is needed to extract the information. This list should be located in the same folder where the plugin is, with the name **counters.txt.** Lets see an example of its content:

```
\Web Service(*)\Total Bytes Sent
\Web Service(*)\Bytes Sent/sec
\Web Service(*)\Total Bytes Received
```

In base to this cmdlet, we could add it after any other cmdlet that give us statistics, preceded by (|), and it will give us information about **all** the properties that this cmdlet has got, though when executing the first cmdlet in a general way without using parameters, it will only return a default info list.

This way, our monitoring possibilities using Powershell are considerably extended.

In case that we want to add new modules to our plugin, before doing anything, try to execute the cmdlet from which we want to get information together with the previously mentioned to could get as much information as possible.

An example of the use of this command would be this:

```
Get-Service | Select-Object -Property *
```

Usually, the result of the Get-Service cmdlet would be a list in table format of all the services with their description and status. Though, when applying this second cmdlet, we get, for each service, information of all the properties that this service has:

```
Name                  : service
RequiredServices      : {service1, service2}
CanPauseAndContinue   : False
CanShutdown           : True
CanStop               : True
DisplayName           : This is a Windows Service
DependentServices     : {service3}
MachineName           : .
ServiceName           : service
ServicesDependedOn    : {service1, service2}
ServiceHandle         : SafeServiceHandle
Status                : Stopped
ServiceType           : Win32ShareProcess
Site                  :
Container             :
```

NOTE: *Is very important to consider that the files though for the plugin in WINDOWS must be edited and stored with carriage returns type "WINDOWS" and that if we use carriage returns type "UNIX" the plugin will not work correctly.*