Configuration of Software Agents

Configuration of Software Agents

We are working on the translation of the Pandora FMS documentation. Sorry for any inconvenience.

What is a Software Agent

They are small pieces of software that are installed in the operating systems and remain running on them to extract monitoring information and regularly send it to the Pandora FMS Data server, which processes and stores it in the database.

Introduction to Software Agent Configuration

The operation of the Software Agent (operation parameters and modules) is determined by its configuration file.

- On MS Windows®:

```
%ProgramFiles%\pandora_agent\pandora_agent.conf
```

- On GNU/Linux® systems:

```
/etc/pandora/pandora_agent.conf
```

General Agent Parameters

Most of the parameters are common for MS Windows® and GNU/Linux® systems. After having modified any general parameter, you must restart the Software Agent.

The agent configuration file encoding is UTF-8 on both GNU/Linux® and MS Windows® systems.

The first time data is received from the Software Agent, all the information is saved in the database. For successive information submissions (and depending on whether the learning mode is enabled) only the following fields of the XML will be updated: `version`, `timestamp` (date) and `os_version` (operating system version), as well as the following configuration file parameters `gis_exec`, `latitude`, `longitude`, `altitude`, `parent_agent_name`, `timezone_offset`, `address`, `custom_field` .

server_ip

IP address or name of the Pandora FMS server to which the data will be sent.

server_path

Route of the server where it receives the data files sent by the agents. By default `/var/spool/pandora/data_in`.

Temporary

Path where the Software Agent stores the data files before they are sent to the server and deleted locally.

description

Send the description of the Software Agent in the XML, and Pandora FMS imports this description when it creates the Logical Agent.

group

If a group with the name indicated in this parameter exists, the Agent will be created within this group unless the server forces the creation of all agents in a given group.

temporal_min_size

If the free space (in megabytes) of the partition where the temporary directory is located is less than this value (default one megabyte), data packet generation stops.

temporal_max_size

Maximum size (in megabytes) allowed for the XML buffer, default value `1024`.

temporary_max_files

Maximum number of files allowed for the XML buffer, default value `1024`.

logfile

Route of the log of the Pandora FMS Agent.

interval

Agent sampling time, in units of seconds. Each time this interval is complete, the Agent will collect information and send it to the Pandora FMS server.

disable_logfile

For MS Windows® only: Disables writing to `pandora_agent.log`.

debugging

If it is active (1), the Agent data files are stored and renamed in the temporary directory and are not deleted after being sent to the server, being able to open the XML files and analyze their content.

agent_name

Allows you to set a custom name. If it is not enabled, the agent name will be the hostname of the machine.

agent_name_cmd

Defines the Agent name using an external command. If `agent_name_cmd` is defined, `agent_name` is ignored. The command should return the name of the agent by STDOUT. If it returns more than one line, only the first one will be used.

agent_alias_cmd

Defines the Agent alias using an external command. If `agent_alias_cmd` is defined, `agent_alias` is ignored. The command should return the name of the agent by STDOUT. If it returns more than one line, only the first one will be used.

address

IP address or domain name associated with the Software Agent. If set to `auto`, the machine's IP address will be obtained and added to the agent as the primary address.

encoding

Install the local system encoding type, such as `ISO-8859-15`, or `UTF-8`.

server_port

Port on which the Pandora FMS Tentacle server listens to receive the data files, `41121` by default.

transfer_mode

Mode of transfer of data files to the serverPandora FMS. Default value `tentacle`.

transfer_timeout

Timeout for file transfer; if the indicated number of seconds is exceeded without completing the transfer, it will be cancelled.

server_pwd

Server password for authentication: Specific for Windows® FTP and for the Tentacle transfer mode, although the password in the latter is optional.

server_ssl

Specific for the transfer mode Tentacle. Allows you to enable (`yes`) or disable (`no`) the encryption of connections using SSL.

server_opts

Used for advanced Tentacle configurations to be able to use an HTTP proxy to send the data to the server. This HTTP proxy must have the CONNECT method enabled. To be able to use the output through a proxy, use the following option (for example):

```
server_opts -y user:pass@proxy.inet:8080
```

This option forces the Tentacle client to send the data through a proxy located at `proxy.inet` and using port `8080`, using the username `user` and the password `pass` to authenticate to that proxy.

delayed_startup

Disabled by default. Waiting time (seconds or minutes) until the agent starts running after it starts. For all Software Agents except on MS Windows®.

startup_delay

Disabled by default. Waiting time, in seconds, until the agent starts running after it starts. For MS Windows® only.

pandora_nice

It is only available for Unix/Linux agents. This parameter allows you to specify the priority that the Pandora FMS Agent process will have in the system.

autotime

If enabled (`1`) it sends a timestamp of special execution (AUTO) which causes the server to use the server's local date and time to set the time of the data, ignoring the time sent by the server. Agent. This is necessary for those agents that, for whatever reason, have an incorrect or very different time from the server.

cron_mode

With this parameter it is possible to make the Agent use the Linux® crontab to run at a certain interval instead of using the Agent's own internal system to run from time to time. Off by default (`0`).

remote_config

Enables (`1`) or disables (`0`) the remote configuration of agents. Its operation is only allowed with

the Tentacle transfer mode.

xml_buffer

If enabled (1), the Software Agent will save in its temporary directory the XML files that it has not been able to send to the server in case of a connectivity problem. They will be sent when communications are restored.

timezone_offset

The Software Agent may well install its timezone offset with the server. This allows the server to offset the time collected by the Agent so that it matches the local time of the server.

```
# Timezone offset: Difference with the server timezone
timezone_offset 3
```

It is calculated by subtracting the agent's time zone from the server's time zone.

parent_agent_name

Indicates the father of the Software Agent. It must be the name of an existing Agent in Pandora FMS.

agent_threads

Only available for Unix/Linux agents: Number of threads that the Agent will launch to execute modules in parallel. By default, Modules are executed one after the other without launching any additional threads. Example:

```
  # Number of threads to execute modules in parallel
  agent_threads 4
```

include

```
include <file>
```

Allows you to include an additional configuration file (`< file >`). This file can include additional Modules and collections to those of the main file. The file can be uploaded by users who have write permissions on Agents (AW).

broker_agent

```
broker_agent <broker_name>
```

Enables the Broker Agent functionality. To activate it, it is only necessary to remove the parameter from the comments and indicate the name (`< broker_name >`) that will be assigned to the Broker Agent.

pandora_user

```
pandora_user <user>
```

This parameter is optional and will allow you to run the Agent as the system user (`<user>`) that you specify. This user must have the permissions to be able to executethe Agent and its associated resources.

custom_id

Agent custom identifier for external applications.

url_address

Custom URL to open it from the Agent in the Console.

custom_fieldX_name

Name of an Agents custom field that already exists in the system. If it does not exist, it will be ignored. Example:

```
custom_field1_name Model
```

custom_fieldX_value

Value for the custom field custom_fieldX_name defined in the previous parameter. Example:

```
custom_field1_value C1700
```

module_macro

Software Agent for Unix/Linux.

```
module_macro<_macro_name_> <value>
```

Defines a local execution macro that can be used in the definition of a Module. These macros are used in the Metaconsole system and in the local Module component system to “abstract” the difficulty of using a Module by directly editing the code, presenting a less advanced user with a local interface that allows “filling” values. . These values are used underneath, using a macro system, relatively similar to the local plugins macro system.

Local run macros begin with `_fieldx_`.

group_password

```
group_password <password>
```

Password for the Agent group. If the group is not password protected you should leave this line as a comment.

ehorus_conf

```
ehorus_conf <path>
```

Absolute path (path) to a valid configuration file of a eHorus agent. The Agent will create a custom field called `eHorusID` that contains the identification key of the eHorus agent.

transfer_mode_user

```
transfer_mode_user <user>
```

User (user) of the files copied in local transfer mode. In the Console folders this user must have read and write permissions for the remote configuration to work correctly. Default is `apache`.

secondary_groups

```
secondary_groups <group name1>, <group name2>, ... <group nameN>
```

Name of the secondary groups (group name) assigned to the Agent. Multiple child groups can be

specified separated by commas. If any of the groups do not exist on the server to which the information is sent, that group will not be assigned, but the Agent creation will not be affected.

standby

```
standby <1|0>
```

If an Agent has standby mode enabled (`standby 1`), the Agent will not perform any checks or send or generate any XML. This configuration directive makes sense in Enterprise installations where there is remote configuration. Thanks to this, an Agent can be silenced at will simply by disabling it.

Debug mode overrides this functionality and the Agent runs normally.

module_absoluteinterval

```
module_absoluteinterval <interval>[s,m,h,d]
```

```
module_absoluteinterval once
```

Specifies the execution interval for the module, but unlike module_interval:

1. Remember the last run date when the agent is restarted. The module will not run until the specified interval has passed.
2. Allows you to specify the interval in seconds, minutes, hours or days (e.g., `30s`, `5m`, `1h`, `7d`).
3. It is possible to configure modules to be executed only once by specifying `once` as the interval value.

Secondary Server

A secondary server can be defined to which the data will be sent in two possible situations depending on the configuration:

- on_error: Send data to the secondary server only if it cannot send it to the primary.
- always: Always sends data to the secondary server, regardless of whether or not it can contact the primary server.

UDP Server

Keep in mind that UDP is by nature insecure (but efficient at sending messages without compromising a certain response).

The Pandora FMS Software Agent can be configured to listen to remote commands. This server listens on a UDP port specified by the user, and allows receiving orders from a remote system, usually from the Pandora FMS Console, by executing alerts on the server.

To configure the remote UDP server, there are the following options in your fileor configuration `pandora_agent.conf>`

- udp_server: To activate the UDP server set the value to 1. By default it is disabled.
- udp_server_port: Listening port number.
- udp_server_auth_address: IP addresses authorized to send orders. To specify multiple addresses, separate them by commas. If set to 0.0.0.0, it accepts commands from any address.

Although 0.0.0.0 can be set to accept from all sources, such a practice is not recommended. If you have several PFMS Servers and/or use IPv6 you can put different IP addresses separated by commas. For example if you have in IPv6:`2001:0db8:0000:130F:0000:0000:087C:140B` and its abbreviation is `2001:0db8:0:130F::87C:140B` use both separated by commas.

- process_<name>_start <command>: Command that will start a process defined by the user.
- process_<name>_stop <command>: Command that will stop the process.
- service_<name> 1: Allows service <name> to be stopped or started remotely from the UDP server.

There is a script on the server, in `/util/udp_client.pl` which is used by Pandora FMS Server as an alert command, to start processes or services. It has this syntax:

```
./udp_client.pl <address> <port> <command>
```

Definition of Modules

The local execution modules are defined in the configuration file `pandora_agent.conf`. The general syntax is as follows:

```
module_begin
module_name <module_name>
module_type generic_data
module_exec <local_command>
module_end
```

Elements common to all Modules

The Module fields (except the Module data, the description and the extended information) are only updated when the

Module is created, they will never be updated once the Module already exists.

module_begin

Start tag of a Module. Mandatory.

module_name

```
module_name <name>
```

Name (name) of the Module. Said name must be unique and singular in the Agent. Mandatory.

module_type

```
module_type <type>
```

Type (type) of data that the Module will return. Mandatory. The types available are:

- Numeric (`generic_data`): Single, floating point, or integer numeric data.
- Incremental (`generic_data_inc`): Numeric data equal to the difference between the current value and the previous value divided by the number of seconds elapsed. When this difference is negative, the value is reset, this means that when the difference becomes positive again, the previous value will be taken as long as the increment returns to a positive value.
- Incremental absolute (`generic_data_inc_abs`): Numeric data equal to the difference between the current value and the previous value, without dividing by the number of seconds elapsed, to measure total increment instead of increment per second. When this difference is negative, the value is reset, this means that when the difference becomes positive again, the last value from which the current increment obtained is positive will be used.
- Alphanumeric (`generic_data_string`): Returns alphanumeric text strings.
- Booleans (`generic_proc`): For values that can only be correct or affirmative (1) or incorrect or negative (0). Useful to check if a computer is alive, or a process or service is running. A negative value (0) is pre-assigned to the critical state, while any higher value will be considered correct.
- Asynchronous alphanumeric (`async_string`): For asynchronous text strings. Asynchronous monitoring depends on events or changes that may or may not occur, so this type of Module is never in an unknown state.
- Async Boolean (`async_proc`): For boolean values of async type.
- Asynchronous Numeric (`async_data`): For numeric values of asynchronous type.

module_min

```
module_min <value>
```

Minimum value (value) that the Module must return to be accepted. Otherwise it will be discarded by the server.

module_max

```
module_max <value>
```

Maximum value (value) that the Module must return to be accepted. Otherwise it will be discarded by the server.

module_min_warning

```
module_min_warning <value>
```

Minimum value (value) of the warning threshold `warning`.

module_max_warning

```
module_max_warning <value>
```

Maximum value (value) of the warning threshold `warning`.

module_min_critical

```
module_min_critical <vamon>
```

Minimum value of the critical threshold `critical`.

module_max_critical

```
module_max_critical <value>
```

Maximum value of the critical threshold `critical`.

module_disabled

```
module_disabled <0|1>
```

Indicates if the module is enabled (`0`) or disabled (`1`).

module_min_ff_event

```
module_min_ff_event <value>
```

Value of the flip-flop protection for false positives. The number of state changes indicated in this value will need to occur for the module to visually change its state in the Web Console.

module_each_ff

```
module_each_ff <0|1>
```

If enabled (1), instead of using `module_min_ff_event` the state-based flip flop thresholds will be used:

- `module_min_ff_event_normal`.
- `module_min_ff_event_warning`.
- `module_min_ff_event_critical`.

module_min_ff_event_normal

```
module_min_ff_event_normal <value>
```

Value of the flip flop protection to go to “normal” state.

module_min_ff_event_warning

```
module_min_ff_event_warning <value>
```

Value of the flip flop protection to go to the `warning` state.

module_min_ff_event_critical

```
module_min_ff_event_critical <value>
```

Value of the flip flop protection to go to the `critical` state.

module_ff_timeout

```
module_ff_timeout <seconds>
```

Resets the flip flop threshold counter after the given number of seconds. This implies that the number of state changes determined in `module_min_ff_event` must occur in an interval of `module_ff_timeout` seconds before the state changes on the console at a visual level.

module_ff_type

Version NG 734 or higher.

```
module_ff_type <value>
```

It is an advanced option of the Flip Flop to control the status of a Module. Through Keep `counters` it establishes counter values to pass from one state to another depending, instead of the value, on the state of the module with the received value.

Indicates whether Keep `counters` is enabled (`1`) or disabled (`0`).

module_ff_event

```
module_ff_event X
```

This directive is the module execution flip flop threshold (in seconds)

module_description

```
module_description <text>
```

Free text with information about the Module.

module_interval

```
module_interval <factor>
```

Individual Modulo interval. This value is a multiplying factor of the agent interval, not a free time.

For the `module_interval` to work on Broker Agents, it must have the same interval as the Agent from which it comes. Otherwise, it may fail to function. As of version 776 the broker agent interval field in the Web Console has been removed.

module_timeout

```
module_timeout <secs>
```

In seconds, maximum time allowed for the execution of the Module. If this time is exceeded before having completed its execution, it will be interrupted.

module_postprocess

```
module_postprocess <factor>
```

Numerical value by which the data returned by the Module will be multiplied. Useful for unit conversions.

module_save

```
module_save <varname>
```

Stores the value returned by the Module in a variable with the name indicated in this parameter (`<var name>`). This value can be used later in other Modules.

Example on Unix/Linux:

```
module_begin
module_name echo_1
module_type generic_data
module_exec echo 41121
module_save ECHO_1
module_end
```

It will store the value “41121” in the variable “ECHO_1”.

```
module_begin
module_name echo_2
module_type generic_data
module_exec echo $ECHO_1
module_end
```

This second Module will show the content of the variable “$ECHO_1”, being “41121”.

In Software Agents on Windows® the Module syntax must be formed by enclosing the variable in percentage symbols `%var%` instead of `$var`. Following the given example:

```
module_begin
module_name echo_2
module_type generic_data
module_exec echo %ECHO_1%
module_end
```

module_crontab

Modules can be scheduled to run on certain dates according to the following format:
`module_crontab <minute> <hour> <day> <month> <day of the week>`

Being:

- Minute 0-59.
- Hour 0-23.
- Day of the month 1-31
- Month 1-12.
- Weekday 0-6 (0 is Sunday) .

module_condition

module_condition <operation> <command>

Allows you to define actions that will be executed by the Agent based on the value returned by the Module. Only available for numeric values. The syntax is as follows:

- > [value]: Execute the command when the value of the Moduleis greater than the given value.
- < [value]: Execute the command when the Modulo value is less than the given value.
- = [value]: Execute the command when the value of the Module is equal to the given value.
- != [value]: Executes the command when the value of the Module is different from the given value.
- =~ [regular|expression]: Executes the command when the modulo value matches the given regular expression.
- (value, value): Executes the command when the value of the module is between the given values.

Multiple conditions can be specified for the same module.

In the MS Windows® operating system it is recommended
to prepend cmd.exe /c to the command to ensure that it
runs properly.

module_precondition

It works the same way as `module_condition`.

module_unit

```
module_unit <string>
```

Units expressed in a text string to display next to the value obtained by the Module. Example:

`module_unit %`.

module_group

```
module_group <value>
```

Allows you to indicate the group of Modules to which the Module will be assigned. Example: `module_group Networking`.

module_custom_id

```
module_custom_id <value>
```

This directive is a custom identifier for the Module. Example: `module_custom_id host101`.

module_str_warning

```
module_str_warning <value>
```

It allows to indicate a regular expression to define the warning threshold `warning` in Alphanumeric type Modules.

module_str_critical

```
module_str_critical <value>
```

Allows you to indicate a regular expression to define the critical threshold `critical` in Alphanumeric type Modules.

module_warning_instructions

```
module_warning_instructions <value>
```

Informs instructions that will be displayed in the event generated by the Module when it goes into the warning state `warning`.

module_critical_instructions

```
module_critical_instructions <value>
```

Reports instructions that will be displayed in the event generated by the module when it goes to `critical` state.

module_unknown_instructions

```
module_unknown_instructions <value>
```

Informs instructions that will be displayed in the event generated by the module when going to unknown state `unknown`.

module_tags

```
module_tags <value>
```

Labels that you want to assign to the Module, separated by commas.

module_warning_inverse

```
module_warning_inverse <value>
```

It allows to activate (`1`) the inverse interval for the warning threshold `warning`.

module_critical_inverse

```
module_critical_inverse <value>
```

Allows you to activate (`1`) the inverse interval for the critical threshold `critical`.

module_native_encoding

For Win32 only.

```
module_native_encoding <value>
```

This configuration token only affects Modules that are executed via a command directive, ie there is a `module_exec` present.

MS Windows® supports three encodings for its processes: the command line (OEM) encoding, the system (ANSI) encoding, and UTF-16. These encodings coincide in the basic characters, but differ in those less common, such as accents. With this token, the Pandora FMS agent converts the

output of the command to the encoding specified in the encoding of the configuration file.

`module_native_encoding` has four valid values:

- `module_native_encoding OEM`: For command line encoding.
- `module_native_encoding ANSI`: For system encoding.
- `module_native_encoding UTFLE`: For UTF-16 little-endian.
- `module_native_encoding UTFBE`: For UTF-16 big-endian.

If `module_native_encoding` does not appear, no re-encoding will be performed.

module_quiet

```
module_quiet <value>
```

If it is enabled (1) the Module will be in silent mode: it will not generate events or trigger alerts.

module_ff_interval

```
module_ff_interval <value>
```

It allows to indicate a Flip Flop threshold in the Module.

module_macro

```
module_macro<macro> <value>
```

Only applicable to local components from the Console. It has no use in the configuration file.

module_alert_template

```
module_alert_template <template_name>
```

This macro assigns to the created Module the alert template corresponding to the name entered as a parameter (see Alert Templates).

intensive_interval

Interval of intensive monitoring. Modules using `module_intensive_monitorig` will be able to notifyar if its status is bad in this interval.

module_intensive_condition

Condition for intensive monitoring. When an intensive monitoring module reaches the value configured in this parameter, it will notify at the intensive interval defined.

module_end

Module end label. It is mandatory.

Specific directives to obtain information

In each Module only one of these types can be used.

module_exec

```
module_exec <command>
```

The desired execution must be specified to obtain the information in a single line.

On GNU/Linux, the command will be executed through the default shell. The default interpreter is determined by the symlink of `/bin/sh`. Normally the link points to bash, but on systems like Ubuntu it doesn't. A solution that will work in most cases:

```
module_exec bash -c "<command>"
```

If the execution of the command returns an error code (return code) different from 0, it will be interpreted that the command fails and the data obtained will be discarded.

For a Software Agent on MS Windows® there are more directives to obtain data, these are described below.

module_exec_powershell

For MS Windows® only.

```
module_exec_powershell < commands >
```

Allows you to perform native checks with PowerShell.

module_service

```
module_service <service>
```

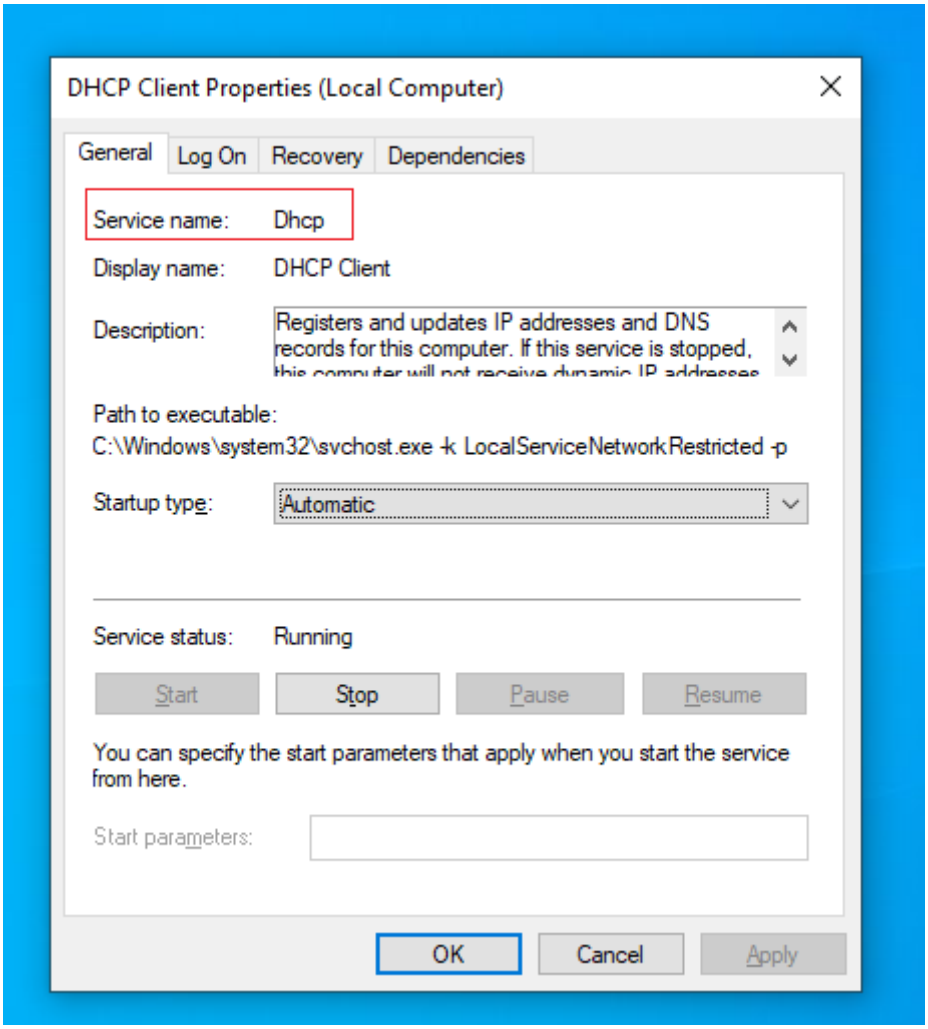
Check if a certain service is running on the machine.

On MS Windows

If the service name contains blank spaces, you must use quotes “ ”.

```
module_begin
module_nameService_Dhcp
module_type generic_proc
module_service Dhcp
module_description Service DHCP Client
module_end
```

The service is identified by the service short name (`Service name`), as it appears in the Windows service manager.



Asynchronous mode

To do this, simply add the directive:

```
module_async yes
```

This functionality is not supported in Broker Agents.

In Windows Home Edition® versions, this asynchronous functionality is not supported and, only in these versions, the Pandora FMS Agent performs a periodic query to find out if the service is running or not. This can be quite resource intensive so it is recommended to use the synchronous version if you are monitoring a large number of services.

Service watchdog

There is a surveillance mode or watchdog for the services, in such a way that the agent can start them again if they stop, it is specified as:

```
module_watchdog yes
```

On Unix

On Unix it works the same as on MS Windows®, only for Unix process and service it's the same concept.

watchdog mode and asynchronous detection are not possible on Unix Agent.

For `module_service` you must put the full path as the service appears with the `ps aux` command. For example, to find the SSH service:

```
  ps aux | grep ssh
```

It must be configured:

```
module_service /usr/sbin/sshd -D
```

module_proc

```
module_proc <process>
```

Check if a certain process name is running on this machine.

On MS Windows®

Quotation around the process name is unnecessary. Note that the process name must have the extension `.exe`. The Module will return the number of processes that are running with this name.

Asynchronous mode

In a similar way to services, monitoring processes can be critical in some cases. The Software Agent for Windows® now supports asynchronous checks for the `module_proc` parameter. In this case the Agent notifies immediately when the process changes state, without waiting for the Agent execution interval to expire. In this way, you can find out about the failure of critical processes almost immediately after they occur. This functionality is not supported in Broker Agents.

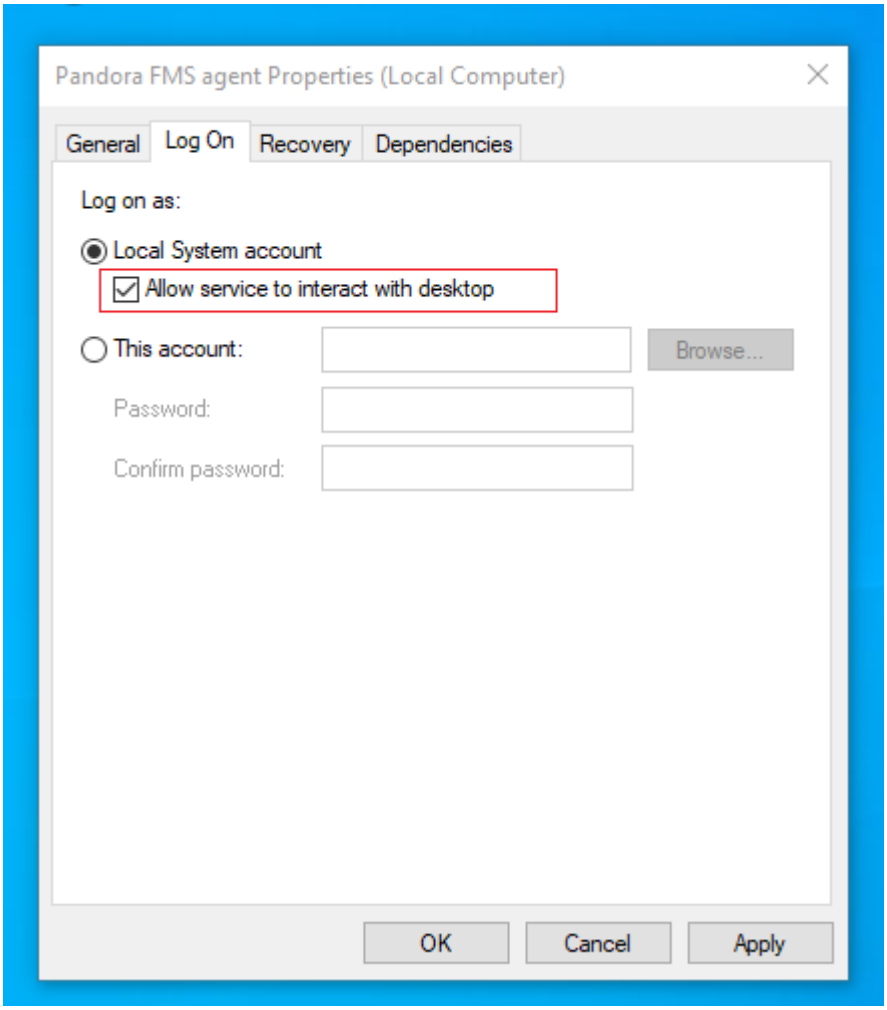
Process Watchdog

It is important to note that Watchdog mode only works when the module type is asynchronous.

Since running a process may require some parameters, there are some additional configuration options for these types of modules. Example of configuring a `module_proc` with watchdog:

```
module_begin
module_name Notepad
module_type generic_proc
module_proc notepad.exe
module_description Notepad
module_async yes
module_watchdog yes
module_start_command c:\windows\notepad.exe
module_startdelay 3000
module_retrydelay 2000
module_retries 5
module_end
```

For versions prior to Windows Vista®, the token `module_user_session` can be configured in a general way by enabling the “Interactive access with desktop” box in the Pandora FMS service properties (Allow service to interact with desktop):



Pandora FMS, as a service, runs under the SYSTEM account and that the process executed will do so under that user and with that environment, so that if you need to run a specific process that needs to be used with a specific user, you must encapsulate in a script (`.bat` or similar) the previous processes to initialize the environment, environment variables, etc), and execute that script as action of the Watchdog .

On Unix

On Unix it works exactly the same as in module_service. It also does not support asynchronous mode or watchdog.

module_cpuproc

For Unix only.

```
module_cpuproc <process>
```

Returns the specific CPU usage of a process.

module_memproc

For Unix only.

```
module_memproc <process>
```

Returns the specific memory consumption of a process.

module_freedisk

```
module_freedisk <disk_letter:>|<vol>
```

Check the free space on the drive.

module_freepercentdisk

```
module_freepercentdisk <disk_letter:>|<vol>
```

This Module returns the percentage of free disk on a logical drive.

module_occupiedpercentdisk

For Unix only.

```
module_occupiedpercentdisk <vol>
```

This Module returns the percentage of busy disk.

module_cpuusage

```
module_cpuusage [<cpu id>|all]
```

Returns the CPU usage in a number of CPUs. If there is only one CPU, do not set any value or use

the value `all`. For Windows® and Unix.

module_freememory

It works on both Unix and Windows®. Returns the free memory in the entire system.

```
module_begin
module_name FreeMemory
module_type generic_data
module_freememory
module_description Non-used memory on system
module_end
```

module_freepercentmemory

It works on both Unix and MS Windows®. This Module returns the percentage of free memory on a system:

```
module_begin
module_name freepercentmemory
module_type generic_data
module_freepercentmemory
module_end
```

module_tcpcheck

Only MS Windows®. This module tries to connect with the specified IP and port. It returns `1` if successful and `0` if not. It is also recommended to specify an expiration time with `module_timeout`. Example:

```
module_begin
module_name tcpcheck
module_type generic_proc
module_tcpcheck www.pandorafms.com
module_port 80
module_timeout 5
module_end
```

module_regexp

For MS Windows® only

This Module monitors a log for matches using regular expressions, discarding already existing lines when starting monitoring. The data returned by the Module depends on the type of Module:

- generic_data_string, async_string: Returns all lines matching the regular expression.
- generic_data: Returns the number of lines matching the regular expression.
- generic_proc: Returns 1 if there is a match, 0 otherwise.
- module_noseekeof: Default inactive `0`. With this configuration token `1` active, at each execution, regardless of the modifications in the log file, the module restarts its check without seeking the end of the file (flag EOF). In this way, it will always output in the XML all those lines that match the search pattern.

module_wmiquery

Windows® only. WMI Modules allow you to run any WMI query or query locally without using an external tool. It is configured through two parameters:

- module_wmiquery: WQL query used. Multiple lines can be obtained as a result, which will be inserted as multiple data.
- module_wmicolumn: Name of the column to be used as data source.

module_perfcounter

For MS Windows® only.

Gets performance counter (performance counter) data through the PDH interface. The `pdh.dll` file must be installed on the system. PDH.DLL belongs to a MS Windows® library, if it is not available you must install the MS Windows® performance analysis tool that usually comes by default.

module_inventory

Currently this functionality has been replaced by inventory from Agent plugins both on Windows® and Linux/Unix® systems.

module_logevent

For MS Windows® only.

It allows obtaining information from the log of MS Windows® events based on the indicated patterns, allowing filtering based on the source and type of event.

The general format of this Module is as follows:

```
module_begin
module_name MyEvent
module_type async_string
module_logevent
module_source <logName>
module_eventtype <event_type/level>
module_eventcode <event_id>
module_application <source>
module_pattern <text substring to match>
module_description
module_end
```

To avoid displaying repeated information, only those events that have occurred since the last time the Agent was run are taken into account.

`module_logevent` accepts the following parameters, all of which require correct case:

- module_source: Event source (System, Application, Security). Obligatory field.
- module_eventtype: Type of event (error, information...). Optional field.
- module_pattern: Pattern to search (substring). Optional field.
- module_eventcode: numerical ID of the event. Optional field.
- module_application: Application that originates the event registered in the log. distinguish well from `module_source` which indicates the name of the source or log file from which the events are searched.

module_logchannel

For MS Windows® only.

Type of Module that allows obtaining information from the logs channels of MS Windows®. It works in the same way as `module_logevent`

To obtain the name of the event channel it will be necessary to right click on it, select Properties and copy the parameter Full name, required for `module_source`.

module_plugin

For the execution of Agent plugins. It is a special case since it does not require any other tags such as `module_begin` or `module_end` and neither does it indicate the type of Module.

Syntax with their respective parameters:

```
module_plugin plugin_filename parameter_1 parameter_2 (...) parameter_X
```

However, it is possible to also use it between the usual Module tags to add additional options such as conditions or interval:

```
  module_begin
  module_plugin plugin_filename parameter_1 parameter_2 (...) parameter_X
  module_interval 2
  module_condition(0, 1) script.sh
  module_end
```

The parameters to be used will be different for each plugin, so it will be necessary to refer to your particular documentation. To describe the operation of one of the plugins that comes by default with the Agent, the plugin `grep_log` serves as an example to search for matches in a file:

```
module_plugin grep_log /var/log/syslog Syslog ssh
```

In this example, the plugin name is called, `grep_log` and it will look for the regular expression `ssh` in the file `/var/log/syslog` and save it in a module called `Syslog`.

module_ping

For MS Windows® only.

```
module_ping <host>
```

This module pings the specified host or host and returns 1 if it is online.

Settings:

- module_ping_count x: Number of ECHO_REQUEST packets to send (1 by default).
- module_ping_timeout x: Expiration time in milliseconds to wait for each response (1000 by default).
- module_advanced_options: Advanced options for `ping.exe`.

module_snmpget

For MS Windows® only.

```
module_snmpget
```

This Module executes a SNMP `get` query and returns the requested value. Configuration parameters must be specified on subsequent lines like this:

- module_snmpversion [1_2c_3]: SNMP version (1 by default).

- module_snmp_community <community>: SNMP community (public by default).
- module_snmp_agent <host>: Target SNMP agent.
- module_snmp_oid <oid>: Target OID.
- module_advanced_options: Advanced options for `snmpget.exe`.

module_wait_timeout

Only for MS Windows®.

Expiration time to be used when checking the output of modules module_exec and module_plugin.

```
module_wait_timeout X
```

Default value 500 milliseconds.

When collecting a lot of data (more than two million bytes) this value can be decreased to 10 milliseconds so that buffer fills and processing (blocks of sixteen thousand bytes) are handled quickly, thus increasing the performance of the PFMS server. It should be used at discretion, for other cases avoid modifying this default value.

module_advanced_options

For MS Windows® only.

```
module_advanced_options <parameter>
```

For module_ping and module_snmpget allow you to use additional parameters.

Automatic agent configuration

Introduction

In the Agent auto-configuration process, you can establish a series of rules so that they are configured automatically and it works as follows:

1. Prepare the automatic configurations in your Pandora FMS Console or Pandora FMS Metaconsole.
2. Install the Agents reporting to your Pandora FMS (if you have a Metaconsole with the auto-provisioning system configured, establish the Metaconsole itself as the server).
3. Pandora FMS Server will receive an XML (`.data`) with the Agent data for the first time.
4. The rules will be evaluated to determine the automatic configuration to apply.
5. The Agent will collect the new configuration and will report in the next cycle with the updated

configuration.

Creating/editing autoconfiguration

Console

Access the management of automatic configurations through Configuration → Manage agent autoconfiguration.

Metaconsole

Go to Centralized management → Agent management → automatic agent configuration icon.

By accessing the administration page you can create new automatic configurations by pressing the Add new configuration definition button. You will need to choose a name and description for your automatic configuration.

Once the new auto configuration is created, you can display the configuration forms by clicking on the section you need: Rules, Agent autoconfiguration or Extra actions.

Rules

To define the Agents to which the automatic configuration will be applied, you can first add rules to identify them.

Expand the rules section within your automatic configuration, and select Add new rule. You can choose a series of options in the rules selector to identify the agents that are going to be configured.

- Server name: Match on server name.
- Group name: Match on group name.
- OS: Match operating system name using regular expressions.
- Custom field: Match by key/value based on a custom field reported by the Agent. Indicate the name of the custom field and the value it should have.
- IP range: Match by range of IP (network) addresses, use IP/mask notation.
- Script output (> 0): Intended to execute a script whose execution result is evaluated as valid when the standard output is greater than 0.
- Rules script call: Supports the following macros in the 'arguments' field (you can choose between AND and OR operators to modify the logic of the rules):
    - `_agent_` : Will be replaced by the name of the Agent.
    - `_agentalias_` : Will be replaced by the Agent's alias.
    - `_address_` : Will be replaced by the primary IP address reported by the Agent.
    - `_agentgroup_` : Will be replaced by the name of the group reported by the Agent.
    - `_agents_` : Will be replaced by the Agent operating system.

If you don't add any rules, the automatic settings will not

be applied. If you need a single configuration for all agents, you can use the following regular expression to match any alias: `.*`

Settings

- Agent Group: You can keep it unchanged or force it to be a specific one.
- Secondary Groups: The groups selected here will be added as secondary groups to the Agent.
- Policies: You can select policies to be applied automatically when the Agent reaches the server.
- Configuration Block: Adds the extra raw configuration to the Agent configuration file.

If you try to access the automatic configuration administration from a node that belongs to a Metaconsole, with centralized administration active, the view will be read-only.

Extra Actions

From this section you can associate other actions to the autoconfiguration, such as:

1. Launch a custom event (Launch custom event).
2. Execute an alert action (Launch alert action).
3. Execute a script (Launch script).

The system supports the following macros:

- `_agent_` Will be replaced by the name of the Agent.
- `_agentalias_` Will be replaced by the Agent's alias.
- `_address_` Will be replaced by the primary IP address reported by the agent.
- `_agentgroup_` Will be replaced by the name of the group reported by the Agent.
- `_togentos_` Will be replaced by the Agent operating system.
- `_agentid_` Replaced by the Agent ID.

Unix/Linux Agents

Configuration of Pandora FMS Unix Agents

The fundamental paths and directories to take into account are:

- `/usr/share/pandora_agent` : Where the Pandora FMS Agent is installed. On systems where this is not allowed by policy, it is recommended to create a link to this path from the actual installation path, for example `/opt/pandora` → `/usr/share/pandora_agent`.
- `/etc/pandora/pandora_agent.conf` : Main configuration file of the Agent. Local Runtime Modules and Agent plugins are configured here.

- `/usr/local/bin/pandora_agent`: Agent executable binary. It usually has a link to `/usr/bin/pandora_agent`.
- `/usr/local/bin/tentacle_client`: Tentacle executable binary, for file transfer to the server. It usually has a link to `/usr/bin/tentacle_client`.
- `/etc/init.d/pandora_agent_daemon`: Script start/stop/restart.
    - On AIX systems the daemon is`/etc/rc.pandora_agent_daemon`.
- `/var/log/pandora/pandora_agent.log`: Text file where the activity of the Pandora FMS Agent is saved, when the Agent runs in debugging mode.
- `/etc/pandora/plugins`: Directory that contains the agent plugins. It is linked to the `/usr/share/pandora_agent/plugins` directory.
- `/etc/pandora/collections`: Directory containing the collections deployed to the Agent. It is linked to the `/usr/share/pandora_agent/collections` directory.

Initial execution of the Unix agent

To start the Agent it is only necessary to execute:

```
/etc/init.d/pandora_agent_daemon start
```

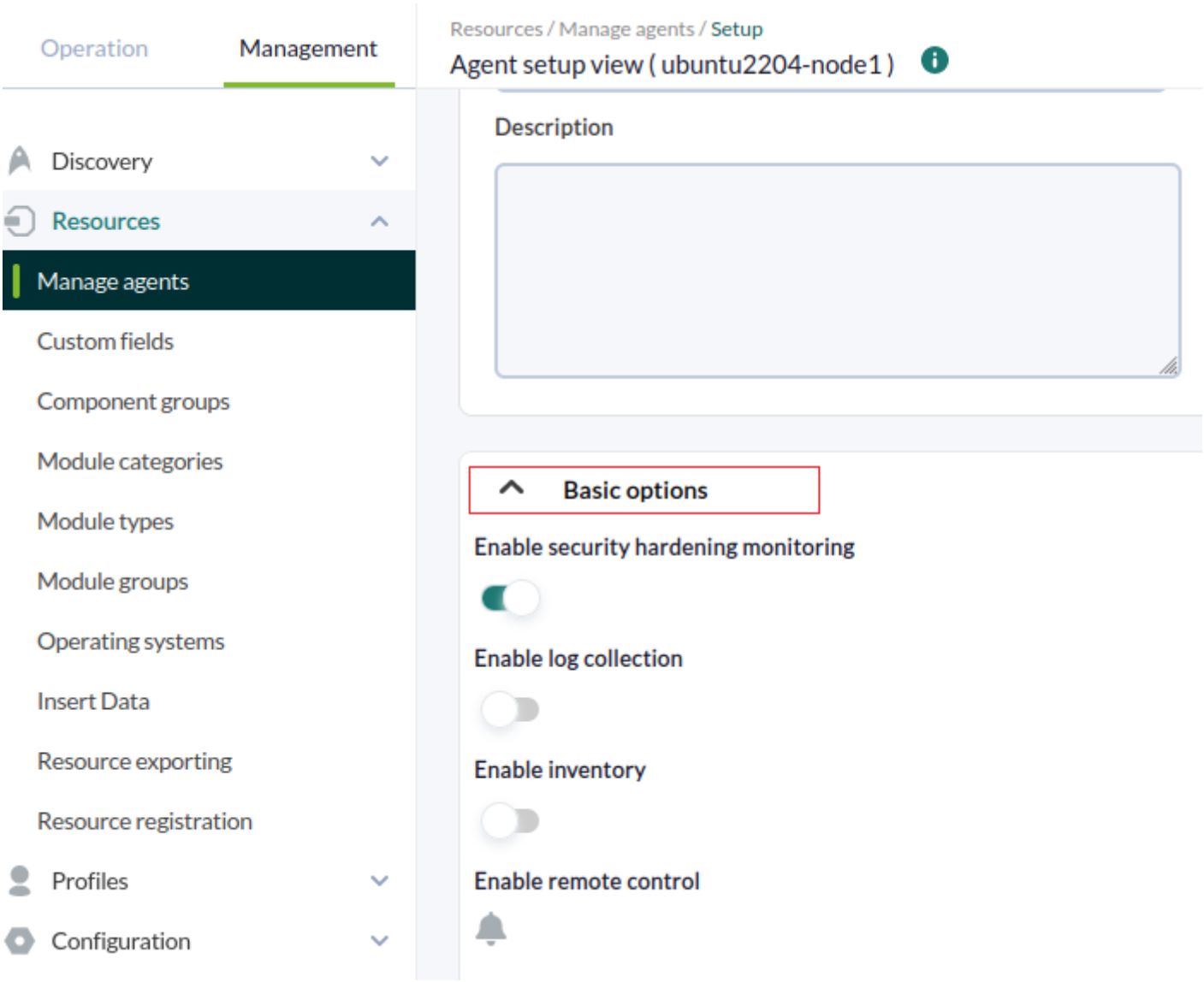
To stop the Agent, run:

```
/etc/init.d/pandora_agent_daemon stop
```

This startup script will be able to start or stop the Pandora FMS Agent, which when started will be running by default in the system as a daemon.

Basic agent options

If the software agent has remote configuration enabled and corresponds to version 774 or later, the following options can be enabled in the Basic options section of the agent configuration in the Web Console.



- Enable security hardening monitoring: It enables the plugin to strengthen security on the monitored device. The following options will be enabled in the configuration file:

```
#Hardening plugin for security compliance analysis. Enable to use it.
module_begin
module_plugin /usr/share/pandora_agent/plugins/pandora_hardening -t 150
module_absoluteinterval 7d
module_end
```

Where the parameters are set to a timeout of 150 seconds for execution (`-t 150`) at an interval of 7 days (`7d`).

- Enable log collection: This will collect the log files for forensic analysis and store all logs. The following options will be enabled in the configuration file:

```
# This is for LOG COLLECTION monitoring, different than log monitoring.
module_plugin grep_log_module /var/log/messages Syslog \.\*
```

- Enable inventory: It enables the option of inventory monitoring. The following options will be enabled in the configuration file:

```
# Plugin for inventory on the agent.
module_plugin inventory 1 cpu ram video nic hd cdrom software init_services
filesystem users route
```

Modify the way Unix agents obtain system information

There are some Modules that obtain the information by default without the need to indicate a command with `module_exec`. These modules are:

- module_procmem
- module_freedisk
- module_freepercentdisk
- module_cpuproc
- module_proc
- module_procmem
- module_cpuusage
- module_freememory
- module_freepercentmemory

It is possible to modify the operation of these Modules by default by directly editing the Agent executable (by default `/usr/bin/pandora_agent`). The Pandora FMS Agent is generally located in `/usr/bin/pandora_agent`.

Look for the string `Commands to retrieve` which carries the code containing the internal commands. Well you can make the modifications you need to adapt them to the system.

```
# Commands to retrieve total memory information in kB
use constant TOTALMEMORY_CMDS => {
    linux => 'cat /proc/meminfo | grep MemTotal: | awk \'{ print $2 }\,
    solaris => 'MEM=`prtconf | grep Memory | awk \'{print $3}\'` bash -c \'echo
$(( 1024 * $MEM ))\,
    hpux => 'swapinfo -t | grep memory | awk \'{print $2}\
};

# Commands to retrieve partition information in kB
use constant PART_CMDS => {
     # total, available, mount point
     linux => 'df -P | awk \'NR> 1 {print $2, $4, $6}\,
     solaris => 'df -k | awk \'NR> 1 {print $2, $4, $6}\,
     hpux => 'df -P | awk \'NR> 1 {print $2, $4, $6}\,
     aix => 'df -kP | awk \'NR> 1 {print $2, $4, $6}\
};
```

To change any of the predefined commands, simply edit the code to modify the command, but be careful about the following:

1. Verify that the `{ };` blocks always end in a semicolon.
2. Verify that the commands are enclosed in single quotes.

3. At the same time inside said quotes, it may be that you need another additional quote with ` ` (note the previous example carefully).
4. Verify that any single quotes you want to use in the command are pre-escaped with the \ character, ie \'. For example, this command would normally be:

```
df -P | awk 'NR> 1 {print $2, $4, $6}'
```

You should write it as:

```
df -P | awk \'NR> 1 {print $2, $4, $6}\'
```

Pandora FMS Windows Agents

Configuration of Pandora FMS Agent for Windows

The fundamental paths and directories in the installations of the Agent for MS Windows® are found in the directory where the Agent has been installed, by default `%ProgramFiles%`.

The most important ones to keep in mind are:

`%ProgramFiles%\pandora_agent`

Where the Pandora FMS Agent, its executable and its directories are installed.

`%ProgramFiles%\pandora_agent\pandora_agent.conf`

Agent main configuration file. Local Runtime Modules and Agent plugins are configured here.

`%ProgramFiles%\pandora_agent\PandoraAgent.exe`

Agent executable binary.

`%ProgramFiles%\pandora_agent\util\tentacle_client.exe`

Tentacle executable binary, for file transfer to the server.

`%ProgramFiles%\pandora_agent\scripts`

Scripts to start/stop/restart the Pandora FMS Agent.

`%ProgramFiles%\pandora_agent\pandora_agent.log`

Text file where the activity of the Pandora FMS Agent is saved, when the agent runs in debugging mode.

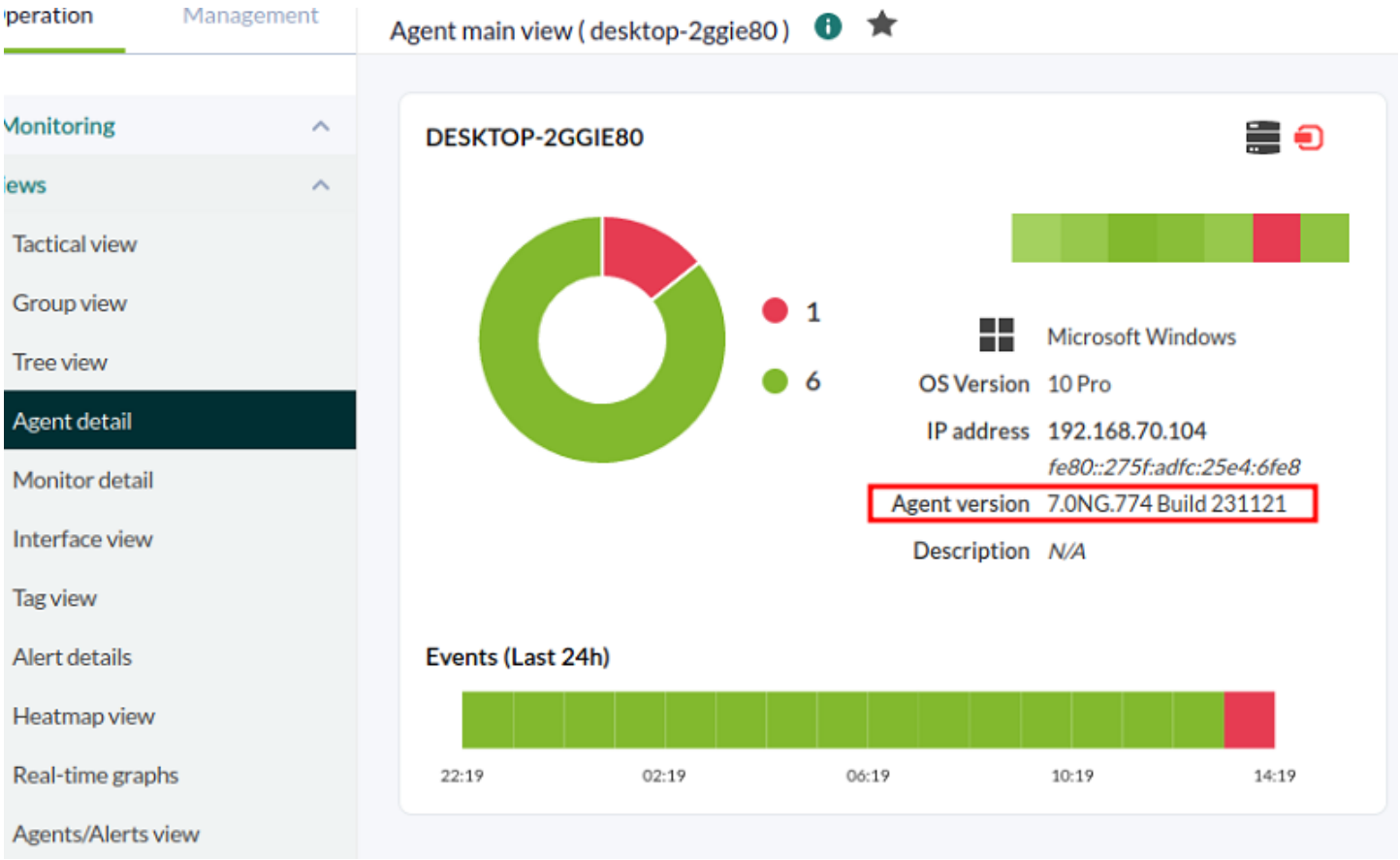
```
%ProgramFiles%\pandora_agent\util
```

Directory containing the agent plugins.

```
%ProgramFiles%\pandora_agent\collections
```

Directory containing the Agent collections.

Basic agent options for MS Windows



When the software agent has Remote configuration enabled and the installed version is 774 or later, the following options can be enabled in the Basic options section of the agent configuration in the Web Console.

- Enable inventory: It enables the inventory monitoring option. The following parameters will be enabled in the configuration file:

```
module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\cpuinfo.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
```

```
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\moboinfo.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\diskdrives.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\cdromdrives.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\videocardinfo.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\ifaces.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\monitors.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\printers.vbs"
module_crontab * 12-15 * * 1
module_end
module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\raminfo.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\software_installed.vbs"
module_crontab * 12-15 * * 1
module_end
```

```
module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\userslogged.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\productkey.vbs"
module_crontab * 12-15 * * 1
module_end

module_begin
module_plugin cscript.exe //B //t:20
"%PROGRAMFILES%\Pandora_Agent\util\productID.vbs"
module_crontab * 12-15 * * 1
module_end
```

- Enable security hardening monitoring: It enables the plugin to strengthen security on the monitored device. The following parameters will be enabled in the configuration file:

```
#Hardening plugin for security compliance analysis. Enable to use it.
module_begin
module_plugin "%PROGRAMFILES%\Pandora_Agent\util\pandora_hardening.exe -t 150"
module_absoluteinterval 7d
module_end
```

Where the options are set to a timeout of 150 seconds for execution (`-t 150`) in an interval of 7 days (7d) as agent period.

- Enable log collection: This will collect the log files for forensic analysis and store all logs. The following parameters will be enabled in the configuration file:

```
module_begin
module_name PandoraAgent_log
module_type generic_data_string
module_regexp C:\archivos de programa\pandora_agent\pandora_agent.log
module_description This module will return all lines from the specified logfile
module_pattern .*
module_end
```

Agent security options for MS Windows

For PFMS software agent version 775 or later a plugin is included which is disabled by default. To enable it, uncomment the following instructions in the configuration file:

```
# Pandora basic security check plugin for windows.
#module_begin
#module_plugin "%PROGRAMFILES%\Pandora_Agent\util\pandora_security_win.exe"
```

```
#module_end
```

Once the software agent has been restarted, the following agent modules will be picked up:

- Antivirus installed and running, either Microsoft or third party, and if your virus definitions are up to date (two modules). On MS Windows server® this feature is not available so modules are not created.
- Check whether the automatic screen lock is active (Lock screen status), this protects the user's account when leaving the computer unattended, without mouse and keyboard activity (one module).
- Check whether MS Windows® was updated (Windows updated®) one week ago or less (one module).
- Firewall status, activated or not (three modules, one for each firewall profile: Domain network, Private network and Public network).
- Verification that all local accounts have a password set (one module).
- A module is dedicated to monitor whether the log of failed session attempts is active or not (only for OS installed in English and Spanish).

Automatic deployment of software agents

You can deploy Software Agents using the deployment center through the Discovery system, more information in this link.

Auto-update Software Agents

Using the file collections and the `pandora_update` tool can provide a way to “auto-update” Software Agents.

The `pandora_update` tool needs the Perl module `Digest::MD5` to work. As of Perl version 5.14, this module is built in by default, but in earlier versions you will need to install it manually.

It works as follows:

1. Agents receive new binaries in the collections input directory.

Example on MS Windows®:

```
%ProgramFiles%\pandora_agent\collections\fc_1\PandoraAgent.exe
```

Example in GNU/Linux®:

```
/etc/pandora/collections/fc_1/pandora_agent
```

2. The Agent executes the plugin `pandora_update`. This plugin takes a single parameter: the short name of the collection (in this example, `fc_1`). It will scan the collection directory looking for the Agent binary (not the entire installer), it will compare the binary located in the collection with the one currently running, and if they are different, `pandora_update` stops the Agent, replaces the binary and restarts the Agent again using the new binary.

To update different architectures you will need to set a different collection for each architecture. For example, if you want to update 32-bit and 64-bit Windows® agents, you must create two collections and in each of them include the corresponding `PandoraAgent.exe` binary.

3. `Pandora_update` also writes to a small log the updated event, to be able to retrieve it on the next run and warn the user (by using an `async_string` Module) about the process Agent Update.

This implies that the Modules used to complete the update process may be configured to have a high interval.

Unix standard installation

```
  module_begin
  module_name Pandora_Update
  module_type async_string
  module_interval 20
  module_exec nohup /etc/pandora/plugins/pandora_update fc_1 2> /dev/null &&
tail -1 nohup.out 2> /dev/null
  module_description Module to check new version of pandora agent and update
itself
  module_end
```

Unix custom installation

```
  module_begin
  module_name Pandora_Update
  module_type async_string
  module_interval 20
  module_exec nohup /var/opt/PandoraFMS/etc/pandora/plugins/pandora_update fc_1
/var/opt/PandoraFMS 2> /dev/null && tail -1 nohup.out 2> /dev/null
  module_description Module to check new version of pandora agent and update
itself
  module_end
```

The `pandora_update` command accepts the path of the Pandora FMS installation directory as a second parameter; it is unnecessary to specify it if the installation was done in the default path.

MS Windows®

```
  module_begin
  module_name Pandora_Update
  module_type async_string
  module_interval 20
  module_exec pandora_update.exe fc_1
  module_description Module to check new version of pandora agent and update
itself
  module_end
```

Autocreation of Agents and Modules from XML

Agents can be configured from the Console in three work modes:

- Learning mode: If the XML received from the Software Agent contains new Modules, they will be automatically created. This is the default behavior.
- Normal mode: New Modules that arrive in the XML will not be created if they have not been previously declared in the console.
- Self-disabled mode: Similar to learning mode, in this mode, in addition, if all the Modules go to an unknown state, the Agent will automatically disable itself, becoming enabled again if it receives new information.

Data that is updated from an existing Module when receiving an XML

When an XML is received that contains information from an already existing Module, only the description and extended information are updated, in addition to the Module data.

GIS data is always updated (if enabled) regardless of whether learning mode is off.

Back to Pandora FMS documentation index