Reference for Pandora FMS Development

Reference for Pandora FMS Development

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

Development in Pandora FMS

Pandora FMS Code architecture

For a separate and detailed explanation about the Pandora FMS database structure go to the following article Pandora FMS Engineering.

How to make compatible links

For all links you must use the ui_get_full_url function.

- How to use ui_get_full_url
  Before the call you must include “functions_ui.php”.
- You need the url for the refresh:
  For example

```
$url_refresh = ui_get_full_url();
```

- You need the url for a relative path
  For example

Old method

```
$url_refresh = ui_get_full_url();
```

New method

$url = ui_get_full_url(“/relative/path/file_script.php”);

- And in javascript? It is just as easy.
  For example

Old method

```
<?php
...
$url = $config['homeurl'] . "/relative/path/file_script.php";
...
?>
```

```
<script type="text/javascript>
...
jQuery.post ('<?php $url; ?>',
 {
 ...
 });
 ...
</script>
```

New method

```
<?php
...
$url = ui_get_full_url("/relative/path/file_script.php");
...
?>
<script type="text/javascript>
...
jQuery.post ('<?php $url; ?>',
 {
 ...
 });
 ...
</script>
```

- Special cases:
  - For direct links to index.php it is not necessary to use this function.
    For example

```
echo '<form method="post"
action="index.php?param=111&param=222&param=333&param=444&param=555&param=666">'
;
```

The entry points of execution in Pandora Console

Pandora Console only has a small amount of entry points to execute the web application.

This is unlike other web applications like for example Wordpress that have only one entry point in the front end and another one in the back end. Or at the other end for example small web applications designed for SMB where each php file is usually an entry point.

Installation

This entry point is for the installation of Pandora Console and the data base. When the installation is finished Pandora Console advises the deletion of this file for security reasons.

```
install.php
```

Normal execution

All interactions between the user and the console by use of their browser are made through this entry point.

```
index.php
```

AJAX requests

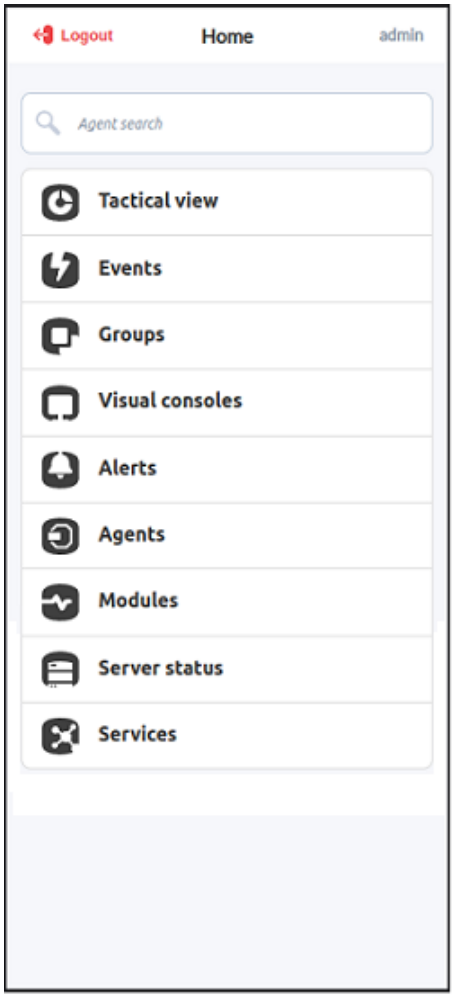
All AJAX requests are through this file, this is because it is necessary to enforce major caution (check the users permission) with this type of actions. It provides consistent structure while also allowing easy maintenance. The actions through this file must pass by means of a GET or POST the parameter “page” that is the relative direction of the script to be executed in the AJAX request.

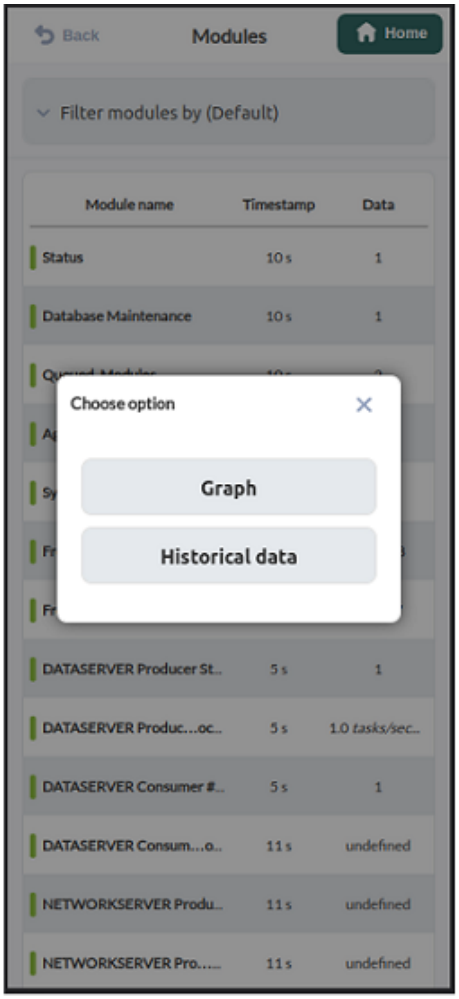
```
ajax.php
```

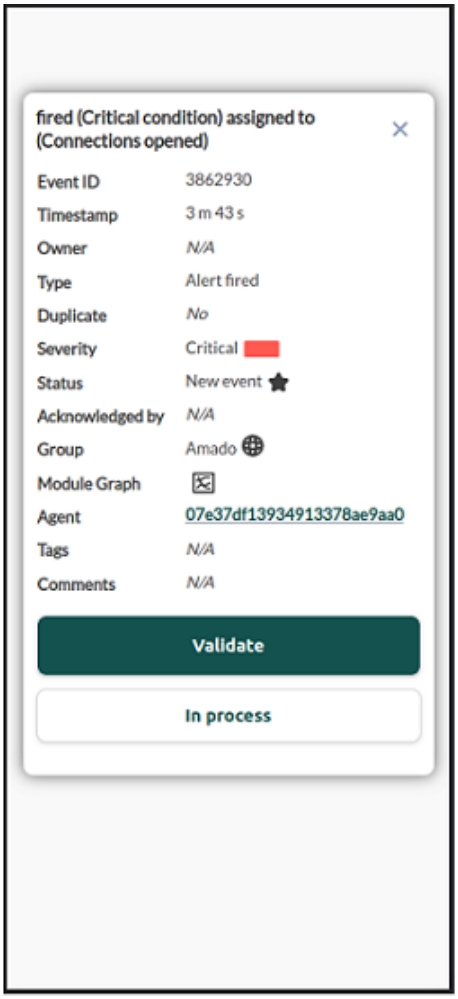
Mobile console

For mobile terminals that have a significantly smaller screen than a computer monitor, Pandora FMS offers a reduced version of the console for these devices, reduced both in visually and simplified functionality wise.

```
mobile/index.php
```







API

From version 3.1 of Pandora FMS, there is included an API of type REST so that third party apps can interact with Pandora FMS across port 80 using the HTTP protocol.

The script must follow these 3 security points:

- The client IP must be in the list of valid IPs or match with any regex in this list. This list is set in the Pandora FMS setup.
- Must pass the parameter with the API password, this password is also set in Pandora FMS setup.
- Must pass the user and password as parameters, this user must have permissions to execute these actions in the API.

```
include/api.php
```

Special cases

In Pandora Console there are several special cases for entry points, these are to avoid the interactive login or general process that make it the main entry point (index.php from root).

Extensión Cron Task

This extension is called by the wget command in the cron, and can execute a limited number of tasks without having logged in.

```
enterprise/extensions/cron/cron.php
```

External view Visual Console

This script generates a page with the view of the Visual Console in full screen (without menus), it doesn't require a login, although for the authentication a hash is needed, this hash is generated by each Visual Console.

```
operation/visual_console/public_console.php
```

Popup detail of Console Networkmap

A popup window that shows the agent detail for any item in the Networkmap Console. This uses for authentication the session values from the user logged into Pandora Console.

```
enterprise/operation/agentes/networkmap_enterprise.popup.php
```

Popup Module Graph

A popup window that shows a module graph, this window has parameters that can be configured to change how the graph is shown. This uses for authentication the session values from the user logged into Pandora Console.

```
operation/agentes/stat_win.php
```

Static graphs

The static graphs are image files that are generated by PHP script, if there is a large amount of data then it saves a serialid in special files that the script creates, these serialized files have a life time so as to avoid bad access and DOS attack. The execution of this file doesn't require authentication in Pandora.

```
include/graphs/fgraph.php
```

Reports

CSV Reports

This script generates a text file that contains the data in CSV format. This script uses the authentication of the logged in user.

```
enterprise/operation/reporting/reporting_viewer_csv.php
```

PDF Report

This script generates a PDF file. This script uses the authentication of the logged in user.

```
enterprise/operation/reporting/reporting_viewer_pdf.php
```

Events

Poput Sound Events

This popup window checks periodically for new events and informs with sound events. This script uses the authentication of the logged in user.

```
operation/events/sound_events.php
```

CSV Events

This script generates a text file that contains the data in CSV format. This script uses the authentication of the logged in user.

```
operation/events/export_csv.php
```

RSS events

This script generates a text file that contains the events in RSS format. This script uses the authentication of the logged in user.

```
operation/events/events_rss.php
```

Basic functions for agent, module and group status

Status criteria and DB encoding

Agent status description:

- Critical (red color): 1 or more modules in critical status.
- Warning (yellow color): 1 or more modules in warning status and none in critical status.
- Unknown (grey color): 1 or more modules in unknown status and none in critical or warning status.
- OK (green color): all modules in normal status.

Internal DB status encoding:

- Critical: 1
- Warning: 2
- Unknown: 3
- Ok: 0

Agents

Status functions

These functions return the number of monitors filtered by status or alert fired by an agent.

For all functions the filter parameter was added to make the function more flexible. The filter content is added at the end of the sql query for all functions. With this filter you can add some specific sql clauses to create filters using tables: tagente_estado, tagente and tagente_modulo.

- agents_monitor_critical ($id_agent, $filter =“”): Returns the number of critical modules for this agent.
- agents_monitor_warning ($id_agent, $filter =“”): Returns the number of warning modules for this agent.
- agents_monitor_unknown ($id_agent, $filter =“”): Returns the number of modules with unknown status.
- agents_monitor_ok ($id_agent, $filter =“”): Returns the number of modules with normal status.
- agents_get_alerts_fired ($id_agent, $filter =“”): Returns the number of alerts fired for this agent.

Auxiliar functions

These functions perform some typical tasks related to agents in some views:

- agents_tree_view_alert_img ($alert_fired): Returns the path to alerts image for tree view depending on the number of alert fired.
- agetns_tree_view_status_img ($critical, $warning, $unknown): Returns the path to status image for tree view

Groups

These functions return the statistics of agents and modules based on agent groups defined in Pandora.

Be careful! The server and console functions must use the same sql queries in order to ensure the result is calculated in the same way

Server functions

- pandora_group_statistics: This function calculates the group statistics when parameter Use realtime statistics is switched off.

Console functions

The console functions calculate the satistics based on an array of agents groups. These functions don't return disabled agents or modules.

- groups_agent_unknown ($group_array): Returns the number of agents with unknown status for a given set of groups.
- groups_agent_ok ($group_array): Returns the number of agents with normal status for a given set of groups.
- groups_agent_critical ($group_array): Returns the number of agents with critical status for a given set of groups.
- groups_agent_warning ($group_array): Returns the number of agents with warning status for a given set of groups.

These functions calculate statistics for modules. Doesn't use disabled modules or agents.

- groups_monitor_not_init ($group_array): Returns the number of monitors with non-init status for a given set of groups.
- groups_monitor_ok ($group_array): Returns the number of monitors with normal status for a given set of groups.
- groups_monitor_critical ($group_array): Returns the number of monitors with critical status for a given set of groups.
- groups_monitor_warning ($group_array): Returns the number of monitors with warning status for a given set of groups.
- groups_monitor_unknown ($group_array): Returns the number of monitors with unknown status for a given set of groups.
- groups_monitor_alerts ($group_array): Returns the number of monitors with alerts for a given set of groups.
- groups_monitor_fired_alerts ($group_array): Returns the number of monitors with alerts fired for a given set of groups.

Modules

These functions return the statistics based on module name. Doesn't use disabled agents or modules for the stats.

- modules_agents_unknown ($module_name): Returns the number of agents with unknown status that have a module with the given name.
- modules_agents_ok ($module_name): Returns the number of agents with normal status that have a module with the given name.
- modules_agents_critical ($module_name): Returns the number of agents with critical status that have a module with the given name.

- modules_agents_warning ($module_name): Returns the number of agents with warning status that have a module with the given name.

These functions return the statistics based on module groups. Doesn't use disabled agents or modules for the stats.

- modules_group_agent_unknown ($module_group): Returns the number of agents with unknown status which have modules that belong to the given module group.
- modules_group_agent_ok ($module_group): Returns the number of agents with normal status which have modules that belong to the given module group.
- modules_group_agent_critical ($module_group): Returns the number of agents with critical status which have modules that belong to the given module group.
- modules_group_agent_warning ($module_group): Returns the number of agents with warning status which have modules that belong to the given module group.

Policies

These functions return the number of agents with each status for a given policy. Doesn't use disabled agents or modules to calculate the result.

- policies_agents_critical ($id_policy): Returns the number of agents with critical status which belong to given policy.
- policies_agents_ok ($id_policy): Returns the number of agents with normal status which belong to given policy.
- policies_agents_unknown ($id_policy): Returns the number of agents with unknown status which belong to given policy.
- policies_agents_warning ($id_policy): Returns the number of agents with warning status which belong to given policy.

OS

These functions calculate the statistics for agents based on Operating Systems. Doesn't use disabled agents or modules.

- os_agents_critical ($id_os): Return the number of agents with critical status which has the given OS.
- os_agents_ok($id_os): Return the number of agents with critical normal which has the given OS.
- os_agents_warning ($id_os): Return the number of agents with critical warning which has the given OS.
- os_agents_unknown ($id_os): Return the number of agents with critical unknown which has the given OS.

Development

Most extensions have been described as independent index, specific for the creation of: server plugin, Unix agent plugin and console extensions. In this section it is described how to collaborate in Pandora FMS and how to compile the Window agent from source. In the future, any other subject related with the development that doesn't have a specific index will be in this chapter.

Cooperating with Pandora FMS project

This project is supported by voluntary developers that support the project. New developers,documentation editors, or people who want to cooperate is always welcome. A good way to start is to subscribe to our mail list and/or to the forum.

Bugs / Failures

Reporting errors helps us to improve Pandora FMS. Please, before sending an error report, check our database for bugs and in case of detecting a non reported one, send it using the Sourceforge tool for tracking and reporting of errors on the Project
WEB:http://sourceforge.net/projects/pandora/

Mailing Lists

Mailing Lists are good, and they are also an easy way of keeping up-to-date. We have a public mailing list for users and news (with low traffic) and a developer mail list for technical debates and notifications (sometimes daily) of the development through our SVN (code version control system) automatic notification system.

Compiling Windows agent from source

Get the latest source

To get the latest source from our repository you will need a Subversion client. Then execute this:

```
svn co https://svn.sourceforge.net/svnroot/pandora pandora
```

Windows

In order to build from source, you will need the latest Dev-Cpp IDE version, with the MinGW tools. Download it from here.

Open `PandoraAgent.dev` with Dev-Cpp and construct the project. Everything should compile for a default installation.

If you encounter any problem when building from source, please contact us by email (ramon.novoa@artica.es) or the SourceForge project web.

Cross-compiling from Linux

To cross-compile the Pandora FMS Windows Agent from Linux follow this steps:

Installing MinGW for Linux

For Ubuntu/Debian:

```
sudo aptitude install mingw32
```

For SUSE or RPM compatible environments (with Zypper of manually) from this URL

```
http://download.opensuse.org/repositories/CrossToolchain:/mingw/openSUSE_11.1/
```

Installing the extra libraries needed by the agent

- win32api
- odbc++
- curl
- openssl
- zlib
- Boost C++ libraries (http://sourceforge.net/projects/boost/files/)

For example, to install Openssl package:

Go to http://sourceforge.net/projects/devpaks/files and download the file

```
openssl-0.9.8e-1cm.DevPak
```

Uncompress the file openssl-0.9.8e-1cm.DevPak:

```
tar jxvf openssl-0.9.8e-1cm.DevPak
```

Copy the libraries and include files to your crossed compiled environment with MinGW:

```
 cp lib/*.a /usr/i586-mingw32msvc/lib/
 cp -r include/* /usr/i586-mingw32msvc/include/
```

There is a faster alternative, but you need to solve problems with dependencies/libraries yourself: We have made a tarball with all needed libraries and included files available on official Pandora FMS project download site. This is called *mingw_pandorawin32_libraries_9Oct2009.tar.gz*

Compiling and linking

After installing compiler, includes and libraries, go to the Pandora FMS Agent source directory and run:

```
./configure --host = i586-mingw32msvc && make
```

This should create the .exe executable, ready to be used.

External API

There is an external API for Pandora FMS in order to link other applications with Pandora FMS, both to obtain information from Pandora FMS and to enter information into Pandora FMS. All this documentation is at Pandora FMS External API

Pandora FMS XML data file format

Knowing the format of Pandora FMS XML data files can help you improve agent plugins, create custom agents or just feed custom XML files to Pandora FMS Data Server.

As any XML document, the data file should begin with an XML declaration:

```
<?xml version ='1.0' encoding ='UTF-8'?>
```

Next the agent_data element, that defines the agent sending the data. It supports the following attributes:

- *description*: Agent description.
- *group*: Name of the group the agent belongs to (it must exist in Pandora FMS database).
- *os_name*: Name of the operating system the agent runs in (it must exist in Pandora FMS database).
- *os_version*: Free string describing the version of the operating system.
- *interval*: Agent interval (in seconds).
- *version*: Agent version string.
- *timestamp*: Timestamp indicating when the XML file was generated (YYYY/MM/DD HH:MM:SS).
- *agent_name*: Name of the agent.
- *timezone_offset*: Offset that will be added to the timestamp (in hours). Useful if you are working with UTC.
- *address*: Agent IP address (or FQN).
- *parent_agent_name*: Name of the agent's parent.
- *agent_alias*: Agent's alias.

- *agent_mode*: Agent's working mode (0: Normal mode, 1: Learning mode, 2: Autodisable mode) .
- *secondary_groups*: Secondary groups added to the agent.
- *custom_id*: Custom agent ID.
- *url_address*: Agent access URL.

Let's see an example of an XML header:

```
 <agent_data description= group= os_name='linux' os_version='Ubuntu 10.10'
interval='30' version='3.2(Build 101227)' timestamp='2011/04/20 12:24:03'
agent_name='foo' timezone_offset='0' parent_agent_name='too'
address='192.168.1.51' custom_id='BS4884' url_address='http://mylocalhost:8080'>
```

Then you need a module element per module, and the following elements can be nested to define the module:

- name: Module name.
- description: Module description.
- tags: Tags associated to the module.
- type: Module type (it must exist in Pandora FMS database).
- data: Module data.
- max: Module's maximum value.
- min: Module's minimum value.
- post_process: Post-process value.
- module_interval: Module interval (interval in seconds/agent interval).
- min_critical: Minimum value for critical status.
- max_critical: Maximum value for critical status.
- min_warning: Minimum value for warning status.
- max_warning: Maximum value for warning status.
- disabled: It disables (0) or enables (1) the module. Disabled modules are not processed.
- min_ff_event: FF threshold.
- status: Module status (NORMAL, WARNING or CRITICAL). Warning and critical thresholds are ignored if the status is set.
- datalist: Sends the module data in datalist format (one database entry for each of the values received) [0/1].
- unit: Module unit. Supports the _Timeticks_ macro to transform a data in timeticks format to dd/hh/mm/ss.
- timestamp: Sets a timestamp on the data received from the module.
- module_group: Group of modules to which the module will be added.
- custom_id: Module custom ID.
- str_warning: Warning threshold for string modules.
- str_critical: Critical threshold for string modules.
- critical_instructions: Module Critical instructions.
- warning_instructions: Module Warning instructions.
- unknown_instructions: Module Unknown instructions.
- critical_inverse: Activates the Inverse interval at the critical threshold of the module. [0/1].
- warning_inverse: Activates the Inverse interval at the warning threshold of the module. [0/1].
- quiet: Activates the Quiet mode of the module [0/1].
- module_ff_interval: Specifies a value of FF Interval of the module.
- alert_template: It associates an alert template to the module.
- crontab: Specifies a crontab in the module.
- min_ff_event_normal: FF threshold value on change of state to NORMAL.

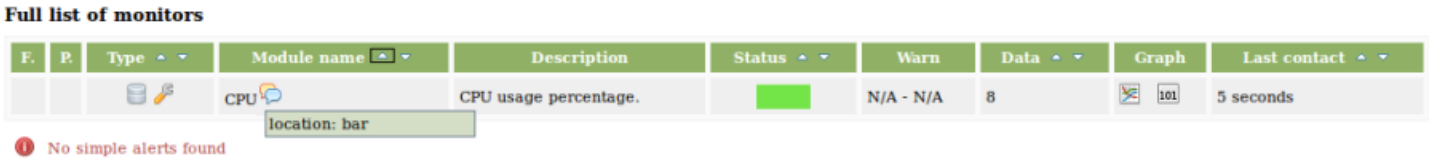
- min_ff_event_warning: FF threshold value on change of state to WARNING.
- min_ff_event_critical: FF threshold value on change of state to CRITICAL.
- ff_timeout: FlipFlop timeout value.
- each_ff: Activate option “Change each status”.
- module_parent: Name of the module in the same agent that will be the parent of this module.
- ff_type: Activates the Keep counter of the FF threshold. [0/1].

From Pandora FMS version 749 onwards new tokens are added to force thresholds:

- min_warning_forced: It forces min_warning to the new indicated value even if the module exists, it has priority over min_warning.
- max_warning_forced: It forces max_warning to the new indicated value even if the module exists, it has priority over max_warning.
- min_critical_forced: It forces min_critical to the new indicated value even if the module exists, it has priority over min_critical.
- max_critical_forced: It forces max_critical to the new indicated value even if the module exists, it has priority over max_critical.
- str_warning_forced: It forces Str_warning to the new indicated value even if the module exists, it has priority over str_warning.
- str_critical_forced: It forces str_critical to the new indicated value even if the module exists, it has priority over str_critical.

These tokens will only work for dataserver plugins.

Any other elements will be saved as extended module information in Pandora FMS database:



A module should at least have a name, type and data element.

For example:

```
<module>
  <name>CPU</name>
  <description>CPU usage percentage</description>
  <type>generic_data</type>
  <data>21</data>
</module>
```

There can be any number of module elements in an XML data file. Last, do not forget to close the agent_data tag!

There is a special case of multi item XML, based on a list of data. This is only applies to string types. The XML will be something like this:

```
  <module>
  <type>async_string</type>
  <datalist>
    <data><value><![CDATA[xxxxx]]></value></data>
    <data><value><![CDATA[yyyyy]]></value></data>
    <data><value><![CDATA[zzzzz]]></value></data>
  </datalist>
  </module>
```

A timestamp may be specified for each value:

```
  <module>
  <type>async_string</type>
  <datalist>
    <data>
      <value><![CDATA[xxxxx]]></value>
      <timestamp>1970-01-01 00:00:00</timestamp>
    </data>
    <data>
      <value><![CDATA[yyyyy]]></value>
      <timestamp>1970-01-01 00:00:01</timestamp>
    </data>
    <data>
      <value><![CDATA[zzzzz]]></value>
      <timestamp>1970-01-01 00:00:02</timestamp>
    </data>
  </datalist>
  </module>
```

These are some more examples involving the use of units and threshold definition:

```
<module>
        <name><![CDATA[Cache mem free]]></name>
        <description><![CDATA[Free cache memory in MB]]></description>
        <tags>tag</tags>
        <type>generic_data</type>
        <module_interval>1</module_interval>
        <min_critical>100</min_critical>
        <max_critical>499</max_critical>
        <min_warning>500</min_warning>
        <max_warning>600</max_warning>
        <unit><![CDATA[MB]]></unit>
        <data><![CDATA[3866]]></data>
</module>

<module>
        <name><![CDATA[Load Average]]></name>
        <description><![CDATA[Average process in CPU (Last minute)
]]></description>
        <tags>tag</tags>
        <type>generic_data</type>
```

```
        <module_interval>1</module_interval>
        <data><![CDATA[1.89]]></data>
</module>
```

Go back to Pandora FMS documentation index