Конфигурация программных агентов

Конфигурация программных агентов

Вернуться в оглавление Pandora FMS

Программные агенты Pandora FMS

Что такое Программный Агент Software Agent

Как видно из названия, это небольшие программы, которые устанавливаются в операционные системы и работают в них для извлечения информации мониторинга и регулярной отправки ее на сервер Pandora FMS.

Для получения информации они используют команды и инструменты операционной системы, в которой они установлены. Они формируют данные в файл в формате XML и отправляют их на сервер данных Pandora FMS, который обрабатывает и сохраняет их в базе данных.

Каждая индивидуальная проверка называется *Модуль* .

Введение в конфигурацию Программного Агента

Работа программного агента определяется его конфигурационным файлом, называемым `pandora_agent.conf`, расположенным в каталоге установки (значение по умолчанию) `%ProgramFiles%\pandora_agent` в системах MS Windows®, и в `/etc` в системах GNU/Linux®. Файл конфигурации содержит все функциональные параметры и модули данного Агента.

Общие параметры Агента

Общие параметры конфигурации программного агента определены в этом разделе. Большинство из них являются общими для систем MS Windows® и GNU/Linux®.

Кодировка файла конфигурации агента - UTF-8 как в системах GNU/Linux®, так и в MS Windows®. Если вы редактируете этот файл вручную, проверьте правильность кодировки, прежде чем перезаписывать его. Если кодировка не UTF-8 и вы используете символы (например, ударение или расширенные символы), Программный агент неправильно их

интерпретирует и не может гарантировать правильную интерпретацию вашей конфигурации.

При первом получении данных от программного агента вся информация сохраняется в базе данных. Для последовательной отправки информации (и в зависимости от того, включен ли режим обучения) будут обновляться только следующие поля XML: `version` (версия), `timestamp` (дата) у `os_version` (версия операционная система), а также следующие параметры конфигурационного файла `gis_exec`, `latitude`, `longitude`, `altitude`, `parent_agent_name`, `timezone_offset`, `address`, `custom_field`

Вы можете найти дополнительную информацию о формате XML файлов Pandora FMS по этой ссылке. Вы также можете получить доступ к генерации тестовых данных (инструменты анализа производительности).

server_ip

IP-адрес или имя сервера Pandora FMS, на который будут отправлены данные.

server_path

Путь к серверу, где он получает файлы данных, отправленные агентами. По умолчанию `/var/spool/pandora/data_in`.

temporal

Путь, где Программный Агент хранит файлы данных перед их отправкой на сервер и локальным удалением.

description

Отправит описание программного агента в XML, и Pandora FMS импортирует это описание при создании Программного Агента.

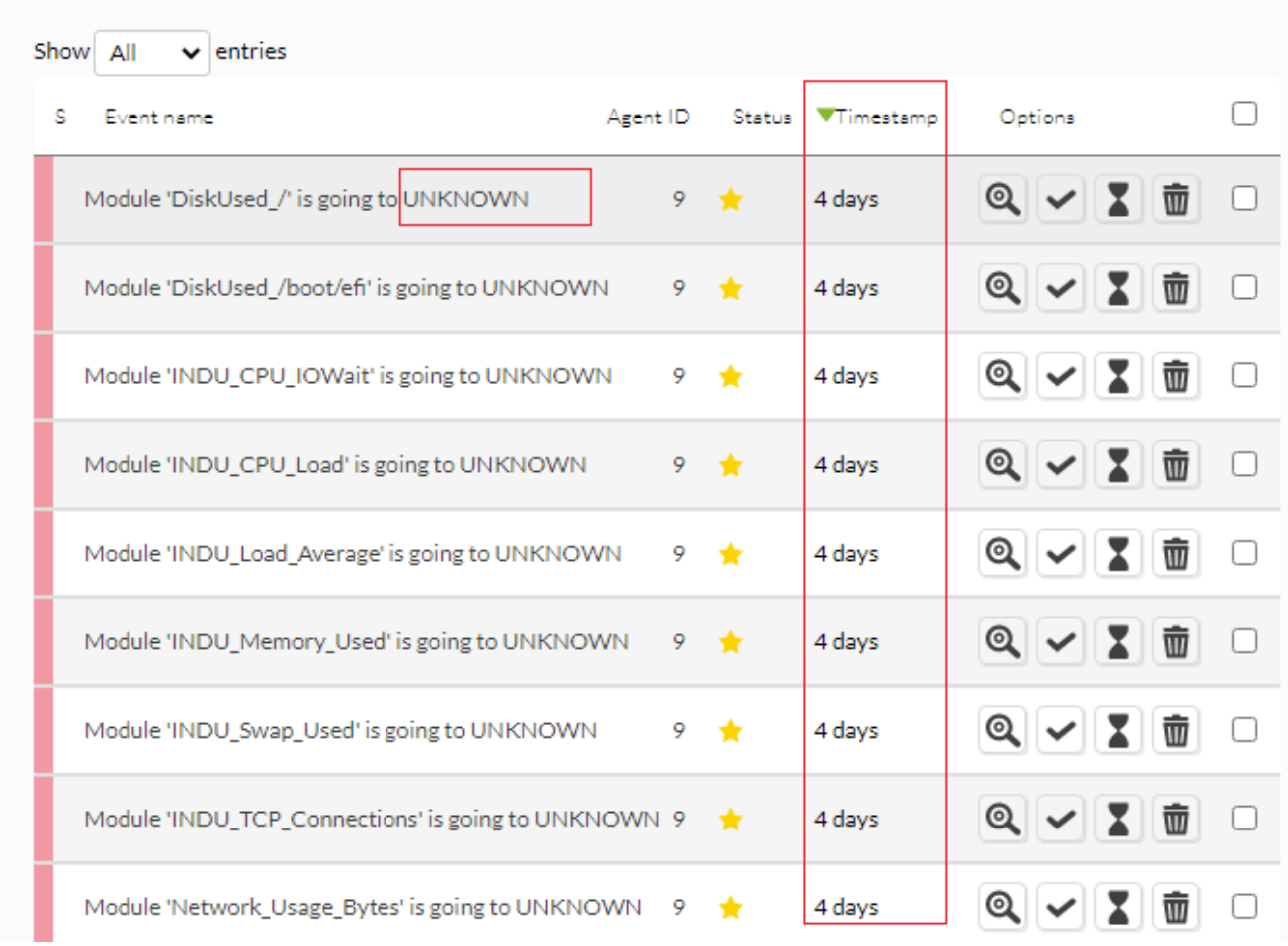
group

Если существует группа с именем, указанным в этом параметре, агент будет создан внутри этой группы, если только сервер не заставит создать всех агентов в определенной группе.

temporal_min_size

Если свободное пространство (в мегабайтах) раздела, в котором находится временный каталог, меньше этого значения (по умолчанию один мегабайт), генерация пакетов данных прекращается. Это предотвращает заполнение диска, если по какой-то причине соединение с первичным и/или вторичным сервером PFMS будет потеряно на длительный период времени.

В Просмотре событий Pandora FMS вы сможете своевременно узнать, если программный агент, независимо от причины, перестал передавать данные на свой основной и/или дополнительный сервер PFMS, поскольку в таких случаях его статус становится «неизвестным» (`UNKNOWN`). Смотрите также, если хотите, создание условий оповещения.



В данном конкретном примере с четырьмя днями автономной работы данные

мониторинга будут сохраняться до тех пор, пока на диске не останется всего 1 мегабайт (или любое другое установленное значение) свободного места.

В связи с вышесказанным, всегда следует помнить, что когда Программный агент наконец-то подключится к первичному и/или вторичному серверу PFMS, объем накопленных данных может перегрузить сервер данных PFMS.

logfile

Путь *журнала* Агента Pandora FMS.

interval

В секундах - время выборки агента. Каждый раз, когда этот интервал завершается, Агент будет собирать информацию и отправлять ее на сервер Pandora FMS.

disable_logfile

Только для MS Windows®: отключает запись в `pandora_agent.log`.

debug

Если активно(1), файлы данных агента хранятся и переименовываются во временном каталоге и не удаляются после отправки на сервер, что позволяет открывать XML-файлы и анализировать их содержимое.

agent_name

Позволяет задать пользовательское имя. Если он не включен, именем агента будет *имя хоста* устройства.

agent_name_cmd

Версия 5.1 SP2 или выше.

Определите имя Агента с помощью внешней команды. Если `agent_name_cmd` определен, то `agent_name` игнорируется. Команда должна вернуть имя агента через STDOUT. Если вы возвращаете более одной строки, будет использована только первая.

agent_alias_cmd

Версия NG 7 или выше.

Определите алиас Агента с помощью внешней команды. Если `agent_alias_cmd` определен, то `agent_alias` игнорируется. Команда должна вернуть имя агента через STDOUT. Если вы возвращаете более одной строки, будет использована только первая.

address

Это IP-адрес, связанный с программным агентом. Это может быть IPv4 с форматом `X.X.X.X`, доменным именем, например `localhost` или `auto`. Если это IP или доменное имя, оно будет добавлено в коллекцию адресов Агента и установлено в качестве основного. Если это `auto`, IP-адрес устройства будет получен и добавлен к агенту таким же образом, как и в предыдущем случае.

encoding

Установите тип кодировки локальной системы, например `ISO-8859-15`, или `UTF-8`.

server_port

Порт, в котором сервер Tentacle Pandora FMS прослушивает прием файлов данных, `41121` по умолчанию.

transfer_mode

Режим передачи файлов данных на сервер Pandora FMS. Значение по умолчанию `tentacle`.

transfer_timeout

Версия 6.0 или выше.

Время ожидания(*timeout*) для передачи файла; если превышено указанное количество секунд, а передача не завершена, то она будет отменена.

server_pwd

Пароль сервера для аутентификации: Специальная для Windows FTP® и для режима передачи Tentacle, хотя для последнего пароль не обязателен.

server_ssl

Специальный для режима передачи Tentacle. Позволяет включить (`1`) или отключить (`0`) шифрование соединений с помощью SSL. Вы можете узнать больше о безопасном соединении с помощью Tentacle в этом разделе.

server_opts

Чтобы добавить дополнительные опции при запуске клиента Tentacle при передаче файлов. Используется для расширенных конфигураций Tentacle с опциями безопасности.

Начиная с версии агентов 3.2 клиент Tentacle поддерживает возможность использования HTTP-прокси для отправки данных на сервер. У этого HTTP-прокси должен быть включен метод CONNECT. Чтобы использовать вывод через *прокси*, используйте следующую опцию (например):

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

Эта опция заставляет клиента Tentacle отправлять данные через прокси, расположенный по адресу `proxy.inet` и использующий порт `8080`, с помощью пользователя `user` и пароля `pass` для аутентификации в этом *прокси*. Если, например, вам нужно использовать *proxy* без аутентификации, на сервере в `192.168.1.2` и с портом `9000`, опция будет следующей:

```
server_opts -y 192.168.1.2:9000
```

Вы можете узнать больше о безопасном соединении с помощью Tentacle в этом разделе.

delayed_startup

Отключен по умолчанию. Время ожидания (секунды или минуты), до начала работы агента после его запуска. Для всех программных агентов, кроме MS Windows®.

startup_delay

Отключено по умолчанию. Время ожидания, в секундах, до начала работы агента после его запуска. Только для MS Windows®.

pandora_nice

Доступно только для агентов Unix/Linux. Этот параметр позволяет указать приоритет, который будет иметь процесс Агента Pandora FMS в системе.

autotime

Если он включен (1), то будет отправлен специальный временной штамп (*timestamp*) выполнения (AUTO), который заставляет сервер использовать локальные дату и время сервера для установки времени данных, игнорируя время, отправленное Агентом. Это необходимо для тех агентов, которые по каким-либо причинам имеют неправильное или сильно отличающееся от сервера время.

cron_mode

С помощью этого параметра можно заставить Агента использовать crontab Linux® для запуска через определенный интервал времени вместо того, чтобы использовать собственную внутреннюю систему Агента для запуска время от времени. Деактивировано по умолчанию (`0`).

remote_config

Включает (1) или выключает (`0`) удаленную конфигурацию агентов. Его функционирование разрешено только в режиме передачи Tentacle.

xml_buffer

Если включено (1), Программный агент будет сохранять во временном каталоге XML-файлы, которые не удалось отправить на сервер в случае проблем с подключением. Они будут отправлены, когда связь будет восстановлена.

timezone_offset

Программный агент вполне может установить себе timezone offset вместе с сервером. Это позволяет серверу смещать время, выбранное Агентом, так, чтобы оно совпадало с местным временем сервера.

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

Смещение часового пояса рассчитывается путем вычитания часового пояса агента из часового пояса сервера. Например, если сервер находится в часовом поясе UTC+1, а агент - в UTC-5, то timezone_offset должен быть равен 6 = 1 - (-5).

parent_agent_name

Указывает на *родителя* Программного Агента. Это должно быть имя существующего агента в Pandora FMS.

agent_threads

Доступно только для агентов Unix/Linux: Количество потоков, которые запускает агент для параллельного выполнения нескольких модулей. По умолчанию модули выполняются один за другим без запуска дополнительных потоков. Пример:

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

include

```
include <file>
```

Позволяет включить файл (*file*) дополнительной конфигурации. Этот файл может включать в себя дополнительные модули и коллекции в дополнение к основному файлу. Файл может быть загружен теми пользователями, которые имеют список профилей на запись в Агентах (AW).

broker_agent

broker_agent <broker_name>

Включает функции агента брокера. Для его активации достаточно убрать из комментариев параметр и указать имя (<broker_name>), которое будет присвоено агенту брокеру.

pandora_user

```
pandora_user <user>
```

Этот параметр является необязательным и позволяет запустить Агент с системным пользователем(<user>), которого вы укажете. Этот пользователь должен иметь разрешения для выполнения Агента и связанных с ним ресурсов.

custom_id

Версия 5.x или выше.

Пользовательский идентификатор Агента для внешних приложений.

url_address

Версия 5.x или выше.

Пользовательский URL для открытия из агента в Консоли.

custom_fieldX_name

Версия 5.x или выше.

Имя пользовательского поля агентов, которое уже существует в системе. Если не существует, будет проигнорировано. Пример:

```
custom_field1_name Model
```

custom_fieldX_value

Версия 5.x или выше.

Значение для пользовательского поля custom_fieldX_name определенного в предыдущем параметре. Пример:

```
custom_field1_value C1700
```

macro

Версия 5.1 или выше Unix/Linux.

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

Определяет Макросы локального запуска, который может быть использован в определении модуля. Эти макросы используются в системе Метаконсоли и в локальной системе компонентов модуля, чтобы не сталкиваться со сложностью использования модуля путем прямого редактирования кода, представляя менее продвинутому пользователю локальный интерфейс, позволяющий «заполнять» значения. Эти значения используются ниже, с помощью системы макросов, относительно похожей на систему макросов локальных *плагинов*.

Локальные макросы выполнения начинаются с `_fieldx_`

Пример:

```
module_begin
 module_name Particion_opt
 module_type generic_data
 module_exec df -kh _field1_ | tail -1 |  awk '{ print $5}' | tr -d "%"
 module_macro_field1_ /opt
 module_end
```

group_password

Версия 6.0 SP5 или выше.

```
group_password <password>
```

Пароль (*password*) для группы агентов. Если группа не защищена паролем, вы должны оставить эту строку в качестве комментария.

ehorus_conf

Версия NG 7 или выше.

```
ehorus_conf <path>
```

Абсолютный путь (*path*) к действительному файлу конфигурации для агента eHorus. Агент создаст пользовательское поле `eHorusID`, которое будет содержать идентификационный ключ агента eHorus.

transfer_mode_user

Версия NG 7.0 OUM713 или выше.

```
transfer_mode_user <user>
```

Пользователь (*user*) файлов, копируемых в режиме локальной передачи. В папках Консоли этот пользователь должен иметь права на чтение и запись, чтобы удаленная конфигурация работала правильно. По умолчанию `apache`.

secondary_groups

Версия NG 7.0 OUM721 или выше.

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

Имя вторичных групп (*group name*), назначенных агенту. Вы можете указать несколько подгрупп, разделенных запятыми. Если одна из групп не существует на сервере, на который отправляется информация, эта группа не будет назначена, но на создание Агента это не повлияет.

standby

Версия NG 7.0 OUM728 или выше.

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

Если у агента включен режим ожидания (`standby 1`), агент не будет выполнять никаких проверок, отправлять или генерировать XML. Эта директива конфигурации имеет смысл в установках Enterprise, где есть удаленная конфигурация. Благодаря этому агента можно поставить в тихий режим по желанию, просто отключив его.

Режим отладки debug переписывает эту функцию, и Агент работает нормально.

Второстепенный сервер

Вы можете определить вторичный сервер, на который будут отправляться данные в двух возможных ситуациях в зависимости от конфигурации:

- on_error: Отправляет данные на вторичный сервер, только если он не может отправить их на первичный.
- always: Всегда отправляет данные на вторичный сервер, независимо от того, может ли он связаться с первичным сервером или нет.

Пример конфигурации:

```
# Secondary server configuration
# ==============================

# If secondary_mode is set to on_error, data files are copied to the secondary
# server only if the primary server fails. If set to always, data files are
# always copied to the secondary server.
#secondary_mode on_error
#secondary_server_ip localhost
#secondary_server_path /var/spool/pandora/data_in
#secondary_server_port 41121
#secondary_transfer_mode tentacle
#secondary_transfer_timeout 30
#secondary_server_pwd mypassword
#secondary_server_ssl no
#secondary_server_opts
```

Сервер UDP

Помните, что UDP по своей природе небезопасен (но эффективен для отправки сообщений без ущерба для определенного ответа).

Программный агент Pandora FMS может быть настроен для прослушивания удаленных команд. Этот сервер прослушивает UDP-порт, указанный пользователем, и позволяет получать приказы от удаленной системы, обычно от консоли Pandora FMS, посредством

выполнения предупреждений на сервере.

Чтобы настроить удаленный сервер UDP, имеются следующие опции в его конфигурационном файле `pandora_agent.conf>`

- udp_server: Чтобы включить сервер UDP, установите значение 1. По умолчанию деактивировано.
- udp_server_port: Номер порта прослушивания.
- udp_server_auth_address: IP-адреса, авторизованные для отправки приказов. Чтобы указать несколько адресов, разделяйте их запятыми. Если установлено значение 0.0.0.0, принимает команды с любого адреса.

Хотя его можно установить в *0.0.0.0* для приема из всех источников, такая практика не рекомендуется. Если у вас несколько серверов PFMS и/или вы используете IPv6, вы можете разместить различные IP-адреса, разделенные запятыми. Например, если вы имеете в IPv6:`2001:0db8:0000:130F:0000:0000:087C:140B` и его сокращение `2001:0db8:0:130F::87C:140B`, используйте оба варианта, разделенные запятыми.

- process_<name>_start <command>: Команда, запускающая определенный пользователем процесс.
- process_<name>_stop <command>: Команда, останавливающая процесс.
- service_<name> 1: Позволяет останавливать или запускать службу <name> удаленно с сервера UDP.

Пример конфигурации:

```
udp_server 1
udp_server_port 4321
udp_server_auth_address 192.168.1.23
process_firefox_start firefox
process_firefox_stop killall firefox
service_messenger 1
```

Сервер принимает следующие команды:

`<START|STOP> SERVICE <service name>`

Запускает или останавливает указанную службу (*service name*).

`<START|STOP> PROCESS <process name>`

Запускает или завершает указанный процесс (*process name*).

```
REFRESH AGENT <agent name>
```

Вызывает выполнение указанного Агента (*agent name*), обновляя данные.

Например:

```
STOP SERVICE messenger
START PROCESS firefox
REFRESH AGENT 007
```

В сервере, в `/util/udp_client.pl`, есть *скрипт*, который используется Pandora FMS Server в качестве команды предупреждения, для запуска процессов или служб. Он имеет следующий синтаксис:

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

Например, для перезапуска агента:

```
./udp_client.pl 192.168.50.30 41122 "REFRESH AGENT"
```

Для получения дополнительной информации см. главу в настройках предупреждений.

Определение модулей

Модули локальной реализации определены в файле конфигурации `pandora_agent.conf`. Общий синтаксис выглядит следующим образом:

```
module_begin
 module_name <module name>
 module_type generic_data
 module_exec <local command>
 module_end
```

Где *module name* - имя модуля, а *local command* - команда для выполнения. Существует множество дополнительных опций для модулей, в данном примере мы использовали только общие и обязательные строки для большинства случаев.

Вы можете узнать больше в обучающих видеороликах:

- «Создание локального модуля Linux в Pandora FMS»(на английском языке).
- «Как создать модуль Windows в Pandora FMS» (на английском языке) .

В следующих разделах подробно рассматривается каждый из них.

Элементы, общие для всех модулей

Поля модуля (за исключением данных модуля, описания и расширенной информации) обновляются только при создании модуля, они никогда не обновляются, если модуль уже существует.

module_begin

Стартовая этикетка модуля. Обязательно.

module_name

```
module_name <name>
```

Имя (*name*) Модуля. Это имя должно быть уникальным и единственным в Агенте. Обязательно.

module_type

```
module_type <type>
```

Тип (*type*) данных, которые будут возвращены Модулем. Обязательно. Доступны следующие типы:

- Числовой (`generic_data`): Простые числовые данные, с плавающей запятой или целые числа.
- Инкрементный (`generic_data_inc`): Числовые данные равны разнице между текущим и предыдущим значением, деленной на количество прошедших секунд. Когда эта разница отрицательна, значение сбрасывается, это означает, что когда разница снова станет положительной, будет принято предыдущее значение до тех пор, пока приращение не вернется к положительному значению.
- Абсолютный инкрементный (`generic_data_inc_abs`): Числовые данные, равные разнице между текущим и предыдущим значением, без деления на количество прошедших секунд, для измерения общего приращения вместо приращения в секунду. Когда эта разница отрицательна, значение сбрасывается, это означает, что когда разница снова становится положительной, используется последнее значение, от которого получено положительное текущее приращение.
- Буквенно-цифровой (`generic_data_string`): Собирает буквенно-цифровые текстовые строки.
- Логические (`generic_proc`): Для значений, которые могут быть только правильно или положительно (1) или неправильно или отрицательно (0). Полезно для проверки того, активно ли устройство, запущен ли процесс или служба. Отрицательное значение (0) приводит к

критическому состоянию, в то время как любое более высокое значение считается правильным.
- Буквенно-цифровой асинхронный (`async_string`): Для текстовых строк асинхронного типа. Такой синхронный мониторинг зависит от событий или изменений, которые могут произойти или не произойти, поэтому данный тип модуля никогда не находится в неизвестном состоянии.
- Логический асинхронный (`async_proc`): Для булевых значений асинхронного типа.
- Числовой асинхронный (`async_data`): Для числовых значений асинхронного типа.

module_min

```
module_min <value>
```

Минимальное значение (*value*), которое должен вернуть модуль, чтобы быть принятым. В противном случае он будет отброшен сервером.

module_max

```
module_max <value>
```

Максимальное значение (*value*), которое должен вернуть модуль, чтобы быть принятым. В противном случае он будет отброшен сервером.

module_min_warning

```
module_min_warning <value>
```

Минимальное значение (*value*) порога предупреждения `warning`.

module_max_warning

```
module_max_warning <value>
```

Максимальное значение (*value*) порога предупреждения `warning`.

module_min_critical

```
module_min_critical <value>
```

Минимальное значение (*value*) критического порога `critical`.

module_max_critical

```
module_max_critical <value>
```

Максимальное значение (*value*) критического порога `critical`.

module_disabled

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

Указывает, включен ли модуль (`0`) или отключен (`1`).

module_min_ff_event

```
module_min_ff_event <value>
```

Значение (*value*) защиты "flip flop« от ложных срабатываний. Количество изменений статуса, указанное в этом значении, будет необходимо для того, чтобы модуль визуально изменил свой статус в веб-консоли.

module_each_ff

Версия 6.0 SP4 или выше.

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

Если включено (`1`), вместо `module_min_ff_event` будут использоваться пороги "flip flop" для каждого состояния:

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

module_min_ff_event_normal

Версия 6.0 SP4 или выше.

```
module_min_ff_event_normal <value>
```

Значение (*value*) защиты "flip flop"для перехода в состояние `normal`.

module_min_ff_event_warning

Версия 6.0 SP4 или выше.

```
module_min_ff_event_warning <value>
```

Значение (*value*) защиты "flip flop" для перехода в состояние `warning`.

module_min_ff_event_critical

Версия 6.0 SP4 или выше.

```
module_min_ff_event_critical <value>
```

Значение (*value*) защиты "flip flop" для перехода в состояние `critical`.

module_ff_timeout

Версия 6.0 SP4 или выше.

```
module_ff_timeout <seconds>
```

Сбрасывает счетчик порога "flip flop" через заданное количество секунд. Это означает, что количество изменений состояния, определенное в `module_min_ff_event`, должно произойти за интервал `module_ff_timeout` секунд до того, как визуальное состояние в консоли изменится.

module_ff_type

Версия NG 734 или выше.

```
module_ff_type <value>
```

Это расширенный вариант "Flip Flop" для контроля состояния модуля. С помощью Keep `counters` установите значения счетчика для перехода из одного состояния в другое в

зависимости не от значения, а от состояния модуля с полученным значением.

Указывает активирован (`1`) или деактивирован (`0`) `Keep counters`.

module_description

```
module_description <text>
```

Общедоступный текст с информацией о модуле.

module_interval

```
module_interval <factor>
```

Индивидуальный модульный интервал. Это значение является множителем интервала агента, а не временем перерыва. Например, если агент имеет интервал 300 секунд (5 минут) и требует, чтобы модуль обрабатывался только каждые 15 минут, вы должны добавить эту строку:

```
module_interval 3
```

Таким образом, модуль будет обрабатываться каждые 300 x 3 секунд = 900 секунд (15 минут).

Чтобы `module_interval` работал в Агентах Брокерах, он должен иметь тот же интервал, что и Агент, из которого он исходит. Несоблюдение этого требования может привести к неисправности.

module_timeout

```
module_timeout <secs>
```

В секундах - максимальное время, отведенное на выполнение модуля. Если это время превышено, а работа с модулем не была завершена, она будет прервана.

module_postprocess

```
module_postprocess <factor>
```

Числовое значение, на которое будут умножены данные, возвращаемые модулем. Удобно использовать для преобразования единиц измерения.

module_save

```
module_save <var name>
```

Сохраняет значение, возвращенное модулем, в переменной с именем, указанным в этом параметре (`<var name>`). Это значение может быть использовано позже в других модулях.

Пример в Unix/Linux:

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

Сохранит значение «41121» в переменной «ECHO_1».

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

Этот второй модуль покажет содержимое переменной «$ECHO_1», равное «41121».

В Программных Агентах Windows® синтаксис модуля должен быть сформирован путем заключения переменной в символы процентов `%var%` вместо `$var$`. Согласно приведенному примеру:

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

module_crontab

Начиная с версии 3.2 вы можете запланировать запуск выполнения модулей на определенные даты.

Для этого нужно определить module_crontab используя формат, аналогичный файлу crontab.

module_crontab <минута> <час> <день> <месяц> <день недели>

Это выглядит следующим образом:

- Минута 0-59.
- Час 0-23 .
- День месяца 1-31 .
- Месяц 1-12 .
- День недели 0-6 (за 0 принимается Воскресенье) .

Также можно указать интервалы, используя символ - в качестве разделителя.

Например, чтобы модуль запускался каждый понедельник с 12:00 до 15:00, можно использовать следующую конфигурацию:

```
module_begin
 module_name crontab_test
 module_type generic_data
 module_exec script.sh
 module_crontab * 12-15 * * 1
 module_end
```

Чтобы выполнять команду каждый час, в заданный час и 10 минут:

```
module_begin
 module_name crontab_test3
 module_type generic_data
 module_exec script.sh
 module_crontab 10 * * * *
 module_end
```

module_condition

module_condition <операция> <команда>

Позволяет определить действия, которые будут выполняться агентом в зависимости от значения, возвращаемого модулем. Доступно только для числовых значений. Синтаксис

выглядит следующим образом:

- > [значение]: Выполняет команду, если значение модуля больше заданного значения.
- < [значение]: Выполняет команду, если значение модуля меньше заданного значения.
- = [значение]: Выполняет команду, когда значение модуля равно заданному значению.
- != [значение]: Выполняет команду, если значение модуля отличается от заданного значения.
- =~ [регулярное|выражение]: Выполняет команду, если значение модуля соответствует заданному регулярному выражению.
- (значение, значение): Выполняет команду, когда значение модуля находится между заданными значениями.

Для одного и того же модуля можно задать несколько условий. В следующем случае выполняется `script_1.sh`, если значение, возвращаемое модулем, находится между 1 и 3, и выполняется `script_2.sh`, если значение модуля больше 5.5, поэтому в данном случае, будучи значением, возвращаемым в строке `module_exec` 2.5, выполняется только первое условие `script_1.sh>`

```
module_begin
module_name condition_test
module_type generic_data
module_exec echo 2.5
module_condition (1, 3) script_1.sh
module_condition> 5.5 script_2.sh
module_end
```

Примеры, применяемые к возможным реальным случаям:

```
module_begin
module_name MyProcess
module_type generic_data
module_exec tasklist | grep MyProcess | wc -l
module_condition> 2 taskkill /IM MyProcess* /F
module_end
```

```
module_begin
module_name Service_Spooler
module_type generic_proc
module_service Spooler
module_condition = 0 net start Spooler
module_end
```

- Примечание: В операционной системе Windows® рекомендуется добавлять префикс cmd.exe /c к команде, чтобы убедиться, что она будет выполнена правильно. Например:

```
module_begin
module_name condition_test
module_type generic_data
module_exec echo 5
module_condition (2, 8) cmd.exe /c script.bat
```

```
 module_end
```

module_precondition

module_precondition <операция> <команда>

Позволяет определить, запускать или не запускать модуль в зависимости от результата данного запуска. Синтаксис:

- > [значение]: Выполняет команду, если значение модуля больше заданного значения.
- < [значение]: Выполняет команду, если значение модуля меньше заданного значения.
- = [значение]: Выполняет команду, когда значение модуля равно заданному значению.
- != [значение]: Выполняет команду, если значение модуля отличается от заданного значения.
- =~ [регулярное|выражение]: Выполняет команду, если значение модуля соответствует заданному регулярному выражению.
- (значение, значение): Выполняет команду, когда значение модуля находится между заданными значениями.

В следующем примере модуль выполняется (*monitoring_variable.bat*), только если результат выполнения, указанный в предусловии, находится между 2 и 8. В данном случае результат выполнения, указанный в строке `module_precondition` равен 5, значение между 2 и 8, поэтому правильно выполняется *monitoring_variable.bat*:

```
module_begin
 module_name Precondition_test1
 module_type generic_data
 module_precondition (2, 8) echo 5
 module_exec monitoring_variable.bat
 module_end
```

Как и в случае с постусловиями, их может быть несколько, и модуль будет выполнен только в том случае, если все они выполнены:

```
module_begin
 module_name Precondition_test2
 module_type generic_data
 module_precondition (2, 8) echo 5
 module_precondition < 3 echo 5
 module_exec monitoring_variable.bat
 module_end
```

- ПРИМЕЧАНИЕ: В операционной системе Windows рекомендуется добавлять префикс cmd.exe /c к команде, чтобы обеспечить ее правильный запуск. Например:

```
module_begin
 module_name Precondition_test3
 module_type generic_data
```

```
module_precondition (2, 8) cmd.exe /c script.bat
module_exec monitoring_variable.bat
module_end
```

module_unit

Версия 5.x или выше.

```
module_unit <string>
```

Единицы измерения, выраженные в текстовой строке (*string*), для отображения рядом со значением, полученным модулем. Пример:

```
module_unit %
```

module_group

Версия 5.x или выше.

```
module_group <value>
```

Позволяет указать группу Модулей (*value*), к которой будет назначен Модуль. Пример:

```
module_group Networking
```

module_custom_id

Версия 5.x или выше.

```
module_custom_id <value>
```

Эта директива является пользовательским идентификатором для модуля. Пример:

```
module_custom_id host101
```

module_str_warning

Версия 5.x или выше.

```
module_str_warning <value>
```

Позволяет указать регулярное выражение для определения порога предупреждения `warning` в буквенно-цифровых модулях (*string*). Пример:

```
module_str_warning .*NOTICE.*
```

module_str_critical

Версия 5.x или выше.

```
module_str_critical <value>
```

Позволяет указать регулярное выражение для определения критического порога `critical` в модулях буквенно-цифрового типа (*string*). Пример:

```
module_str_critical .*ERROR.*
```

module_warning_instructions

Версия 5.x или выше.

```
module_warning_instructions <value>
```

На информативном уровне он указывает инструкции, которые должны быть отображены в событии, генерируемом модулем, когда он переходит в статус предупреждения `warning`.

Пример отображения сообщения, указывающего на повышение приоритета проблемы:

```
module_warning_instructions Raise advocacy priority
```

module_critical_instructions

Версия 5.x или выше.

```
module_critical_instructions <value>
```

На информативном уровне он указывает инструкции, которые должны быть отображены в событии, генерируемом модулем, когда он переходит в критическое `critical` состояние.

Пример сообщения для оповещения системного отдела:

```
module_critical_instructions Call the systems department
```

module_unknown_instructions

Версия 5.x или выше.

```
module_unknown_instructions <value>
```

В информационных целях указывает инструкции, которые будут отображаться в событии, генерируемом модулем, когда он переходит в неизвестный статус unknown.

Пример сообщения для открытия инцидента:

```
module_unknown_instructions Open incident
```

module_tags

Версия 5.x или выше.

```
module_tags <value>
```

Теги или *tags*, которые вы хотите присвоить модулю, разделенные запятыми.

Пример:

```
module_tags tag1,tag2,tag3
```

module_warning_inverse

Версия 5.x или выше.

```
module_warning_inverse <value>
```

Позволяет включить (1) обратный интервал для порога предупреждения `warning`.

Пример:

```
module_warning_inverse 1
```

module_critical_inverse

Версия 5.x или выше.

```
module_critical_inverse <value>
```

Позволяет включить (1) обратный интервал для активации критического порога. `critical`.

Пример:

```
module_critical_inverse 1
```

module_native_encoding

Только на Win32.

```
module_native_encoding <value>
```

Этот конфигурационный *токен* влияет только на модули, которые выполняются через командную директиву, т.е. присутствует `module_exec`.

MS Windows® использует три кодировки для своих процессов: кодировка командной строки (OEM), системная кодировка (ANSI) и UTF-16. Эти кодировки согласуются по основным

символам, но различаются по менее распространенным, таким как ударения. С помощью этого *токена* агент Pandora FMS преобразует вывод команды в кодировку, указанную в *encoding* конфигурационного файла.

`module_native_encoding` имеет четыре допустимых значения:

- `module_native_encoding OEM`: Для кодирования командной строки.
- `module_native_encoding ANSI`: Для системного кодирования.
- `module_native_encoding UTFLE`: Для UTF-16 little-endian.
- `module_native_encoding UTFBE`: Для UTF-16 big-endian.

Если `module_native_encoding` не появляется, повторное кодирование выполняться не будет.

module_quiet

Версия 5.x или выше.

```
module_quiet <value>
```

Если включено (1), модуль будет работать в тихом режиме: он не будет генерировать события или запускать оповещения, а также хранить исторические данные.

Пример:

```
module_quiet 1
```

module_ff_interval

Версия 5.x или выше.

```
module_ff_interval <value>
```

Позволяет указать порог "Flip Flop" в Модуле, например:

```
module_ff_interval 2
```

module_macro

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

Применимо только к локальным компонентам из Консоли. Он не используется в конфигурационном файле.

module_alert_template

```
module_alert_template <template_name>
```

Данный макрос присваивает созданному Модулю шаблон предупреждения, соответствующий названию, введенному в качестве параметра (см. Шаблоны предупреждения).

Пример:

```
<module>
 <name><![CDATA[CPU usage]]></name>
 <type>generic_data</type>
 <module_interval>1</module_interval>
 <min_critical>91</min_critical>
 <max_critical>100</max_critical>
 <min_warning>70</min_warning>
 <max_warning>90</max_warning>
 <alert_template><![CDATA[Critical condition]]></alert_template>
 <data><![CDATA[92]]></data>
 </module>
```

module_end

Этикетка конца модуля. Обязательно.

Конкретные директивы по получению информации

Ниже описаны конкретные директивы, которые могут быть указаны в каждом модуле для получения информации. В каждом модуле *может использоваться только один из этих типов*.

module_exec

```
module_exec <команда>
```

Общая строка выполнения команд. Для получения информации только в одной строке

необходимо указать желаемое исполнение.

В GNU/Linux команда выполняется командным интерпретатором по умолчанию. Интерпретатор по умолчанию определяется символической ссылкой в `/bin/sh`. Обычно ссылка указывает на bash, но в системах типа Ubuntu это не так (в этом случае она указывает на dash). Поэтому может случиться так, что команда проверяется в терминале, а затем выдает ошибку, когда Software Agent выполняет ее. Решением, которое сработает в большинстве случаев, является принудительное выполнение команды в bash следующим образом:

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

Если при выполнении команды возвращается код ошибки (return code), отличный от 0, то интерпретируется, что команда не работает и полученные данные отбрасываются.

Для агента Windows существует больше директив для получения данных, они описаны ниже.

module_service

```
module_service <service>
```

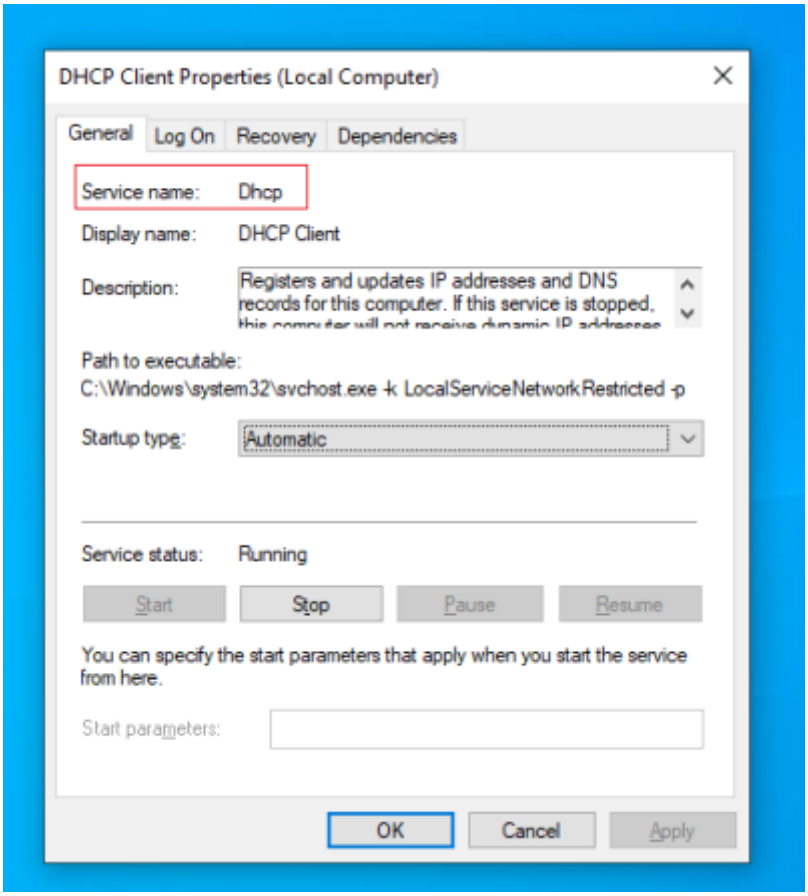
Проверяет, запущена ли на устройстве определенная служба.

В MS Windows

Если имя службы содержит пробелы, необходимо использовать кавычки » «.

```
module_begin
 module_name Service_Dhcp
 module_type generic_proc
 module_service Dhcp
 module_description Service DHCP Client
 module_end
```

Служба идентифицируется коротким именем службы (<код>Service name<код>), таким же, каким оно появляется в менеджере служб Windows.



Асинхронный режим

Pandora FMS, в общем, выполняет ряд тестов (каждый из которых определен в модуле) каждые X секунд (по умолчанию 300 секунд = 5 минут), поэтому если сервис не отвечает сразу после выполнения Pandora FMS, потребуется еще как минимум 300 секунд, чтобы узнать, что он не функционирует. Подобные синхронные модули заставляют Pandora FMS *постоянно* уведомлять о падении данного сервиса. Мы называем такой режим работы *асинхронным*. Для этого просто добавьте директиву.

```
module_async yes
```

Эта функциональность не поддерживается в Агентах Broker.

В версиях Windows Home Edition® эта асинхронная функция не поддерживается, и только в этих версиях Pandora FMS Agent выполняет периодический запрос, чтобы узнать, запущена служба или нет. Это может быть довольно ресурсоемким, поэтому рекомендуется использовать синхронную версию, если вы следите за большим количеством сервисов.

Сторожевой таймер сервисов

Для сервисов существует режим сторожевого таймера *watchdog*, чтобы агент мог запустить их снова, если они остановятся. В этом случае перезапущенная служба не требует никакого параметра, поскольку Windows® уже знает, как это сделать. В этом случае конфигурация проще, а следующее может быть ее примером:

```
module_begin
 module_name ServiceSched
 module_type generic_proc
 module_service Schedule
 module_description Service Task scheduler
 module_async yes
 module_watchdog yes
 module_end
```

В Unix

В Unix работает так же, как и в MS Windows®, только для Unix процесс и сервис - это одно и то же понятие, например, чтобы узнать, активен ли в системе процесс bash, просто выполните:

```
module_begin
 module_name Service_bash
 module_type generic_proc
 module_service /bin/bash
 module_description Process bash running
 module_end
```

Режим *watchdog* и асинхронное обнаружение невозможны в Unix Agent.

module_proc

```
module_proc <process>
```

Проверяет, работает ли заданное имя процесса на этом устройстве.

В MS Windows

Кавычки для имени процесса не нужны. Обратите внимание, что имя процесса должно иметь расширение `.exe`. Модуль вернет количество процессов, выполняющихся с этим именем.

Это будет примером мониторинга процесса `cmd.exe>`

```
 module_begin
```

```
module_name CMDProcess
module_type generic_proc
module_proc cmd.exe
module_description Process Command line
module_end
```

Асинхронный Режим

По аналогии с сервисами, мониторинг процессов в некоторых случаях может иметь решающее значение. Программный агент для Windows® теперь поддерживает асинхронные проверки для параметра `module_proc`. В этом случае агент уведомляет немедленно, когда процесс меняет состояние, не дожидаясь истечения интервала выполнения агента. Таким образом, вы сможете узнать о появлении критических процессов практически сразу после их возникновения. Пример асинхронного мониторинга процесса:

```
module_begin
module_name Notepad
module_type generic_proc
module_proc notepad.exe
module_description Notepad
module_async yes
module_end
```

Разница заключается в конфигурации *token*. `module_async yes`. Данная функция не поддерживается на Агентах Broker.

Сторожевой таймер процессов

Сторожевой таймер (Watchdog) - это система, которая позволяет действовать немедленно при сбое процесса, как правило, поднимая процесс, в котором произошел сбой. Программный Агент Pandora FMS для Windows®, может действовать как *Watchdog* в случае сбоя процесса.

Поскольку для запуска процесса могут потребоваться некоторые параметры, для этого типа модуля существуют некоторые дополнительные параметры конфигурации.

Важно отметить, что режим *Watchdog* работает только
тогда, когда тип модуля *асинхронный*.

Пример конфигурации `module_proc` со сторожевым таймером:

```
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
```

Это определение дополнительных параметров для `module_proc` с *Watchdog*:

module_retries

Количество последовательных попыток, при которых модуль будет пытаться запустить процесс перед отключением *Watchdog*. При достижении лимита механизм *Watchdog* для данного модуля будет отключен и больше никогда не будет пытаться запустить процесс до тех пор, пока Агент не будет перезапущен. Значение по умолчанию: неограничен.

module_startdelay

Количество миллисекунд, в течение которых модуль ждет перед первым запуском процесса.

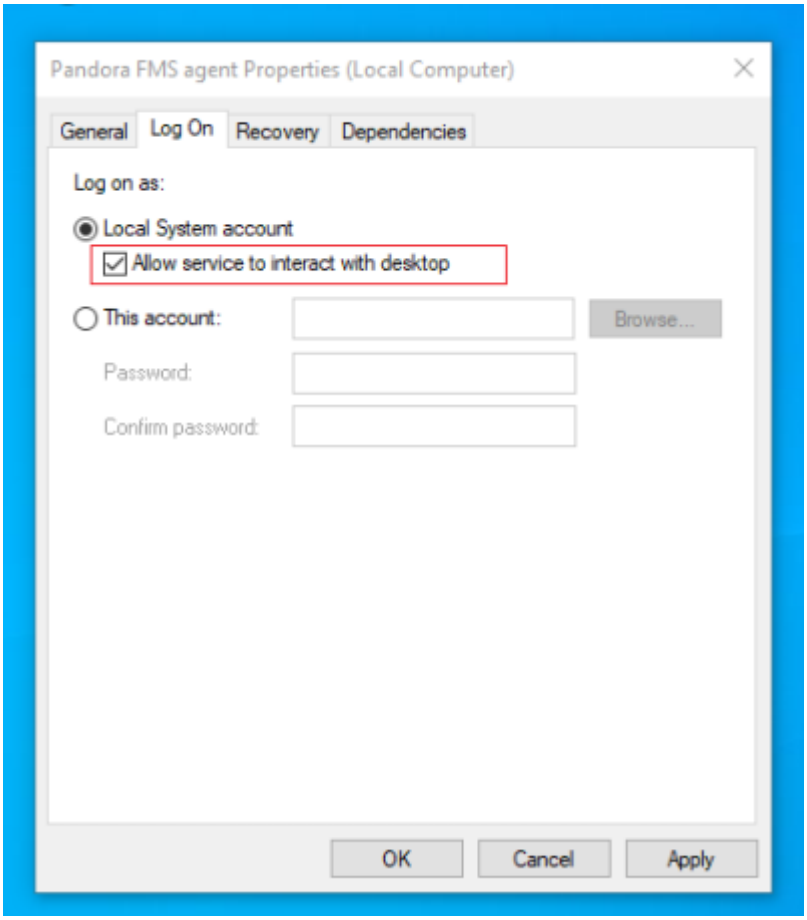
module_retrydelay

Количество миллисекунд, в течение которых модуль ожидает перед попыткой запуска процесса при каждой повторной попытке.

module_user_session

Контролирует, в какой сессии вы хотите, чтобы процесс был запущен. Если установлено значение no, процесс будет запущен в сеансе служб и, следовательно, останется в фоновом режиме (опция по умолчанию). В противном случае, если установлено значение `yes`, процесс будет запущен в сессии пользователя и будет виден с рабочего стола ПК.

Для версий до Windows Vista® *токен* `module_user_session` можно настроить общим способом, включив в свойствах службы Pandora FMS флажок «Интерактивный доступ к рабочему столу» (Allow service to interact with desktop):



Pandora FMS, как служба, выполняется под учетной записью SYSTEM, а выполняемый процесс будет делать это под этим пользователем и с этой средой, поэтому если вам нужно выполнить какой-то конкретный процесс, который нужно использовать с определенным пользователем, вы должны заключить предыдущие процессы в *скрипт* (`.bat` или аналогичные) для инициализации среды, переменных среды и т.д., и запустить этот *скрипт* в качестве действия *Watchdog*.

В Unix

В Unix это работает точно так же, как и в module_service. Он также не поддерживает асинхронный режим или *watchdog*.

module_cpuproc

Только для Unix

```
module_cpuproc <process>
```

Возвращает специфическое использование процессора в данном процессе. Пример:

```
module_begin
 module_name myserver_cpu
 module_type generic_data
 module_cpuproc myserver
 module_description Process Command line
 module_end
```

module_memproc

```
module_memproc <process>
```

Только Unix. Возвращает конкретное потребление памяти процессом.

```
module_begin
 module_name myserver_mem
 module_type generic_data
 module_memproc myserver
 module_description Process Command line
 module_end
```

module_freedisk

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

Проверьте свободное пространство на диске.

В Windows

Вы должны поместить `:` после буквы диска (`<disk_letter:>`).

```
module_begin
 module_name freedisk
 module_type generic_data
 module_freedisk C:
 module_end
```

В Unix

Проверяемый объем, например `/var`.

```
module_begin
 module_name disk_var
 module_type generic_data
 module_freedisk /var
 module_end
```

module_freepercentdisk

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

Этот модуль возвращает процент свободного диска на логическом диске.

В Windows

Вы должны поместить `:` после буквы диска (`<disk_letter:>`).

```
module_begin
 module_name freepercentdisk
 module_type generic_data
 module_freepercentdisk C:
 module_end
```

В Unix

Проверяемый объем, например `/var`.

```
module_begin
 module_name disk_var
 module_type generic_data
 module_freepercentdisk /var
 module_end
```

module_occupiedpercentdisk

```
module_occupiedpercentdisk <vol>
```

Только для Unix. Этот модуль возвращает занятый процент диска, например:

```
module_begin
 module_name disk_var
 module_type generic_data
 module_occupiedpercentdisk /var
```

```
 module_end
```

module_cpuusage

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

Возвращает использование процессора в номере процессора. Если имеется только один процессор, не устанавливайте никакого значения или используйте значение `all`. Для Windows® и Unix.

Также можно получить среднее использование всех процессоров в многопроцессорной системе:

```
module_begin
 module_name CPU_use
 module_type generic_data
 module_cpuusage all
 module_description CPU average use
 module_end
```

Для проверки использования CPU #1:

```
module_begin
 module_name CPU_1
 module_type generic_data
 module_cpuusage 1
 module_description CPU #1 average use
 module_end
```

module_freememory

Он работает как в Unix, так и в Windows®. Возвращает свободную память во всей системе.

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

module_freepercentmemory

Он работает как в Unix, так и в Windows®. Этот модуль возвращает процент свободной памяти в системе:

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

module_tcpcheck

Только MS Windows®. Этот модуль инициирует соединение с указанным IP-адресом и портом. Возвращает 1 в случае успеха и 0 в противном случае. Вы должны указать время истечения срока действия с помощью `module_timeout`. Пример:

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

module_regexp

Только для MS Windows®. Этот модуль отслеживает файл журнала (*log*) в поисках совпадений, используя регулярные выражения, отбрасывая существующие строки при запуске мониторинга. Данные, возвращаемые модулем, зависят от типа модуля:

- generic_data_string, async_string: Возвращает все строки, соответствующие регулярному выражению.
- generic_data: Возвращает количество строк, соответствующих регулярному выражению.
- generic_proc: Возвращает 1, если есть совпадение, 0 в противном случае.
- module_noseekeof: по умолчанию неактивно0. Если *token* конфигурации активен 1, при каждом запуске, независимо от изменений в файле *log*, модуль перезапускает свою проверку без поиска окончания файла (*flag*EOF). Таким образом, вы всегда будете получать в XML все те строки, которые соответствуют шаблону поиска. Пример:

```
module_begin
 module_name regexp
 module_type generic_data_string
 module_regexp %SystemRoot%\my.log
 module_pattern ^\[error].*
```

```
 module_noseekeof 1
 module_end
```

module_wmiquery

Только Windows®. Модули WMI позволяют локально выполнить любой запрос или em>query WMI без использования внешнего инструмента. Он настраивается с помощью двух параметров:

- module_wmiquery: WQL *query* применено. В результате вы можете получить несколько строк, которые будут вставлены как несколько данных.
- module_wmicolumn: Имя столбца, который будет использоваться в качестве источника данных.

Например, для списка установленных служб:

```
module_begin
 module_name Services
 module_type generic_data_string
 module_wmiquery Select Name from Win32_Service
 module_wmicolumn Name
 module_end
```

Чтобы получить текущую загрузку процессора:

```
module_begin
 module_name CPU_speed
 module_type generic_data
 module_wmiquery SELECT LoadPercentage FROM Win32_Processor
 module_wmicolumn LoadPercentage
 module_end
```

module_perfcounter

Только для MS Windows®.

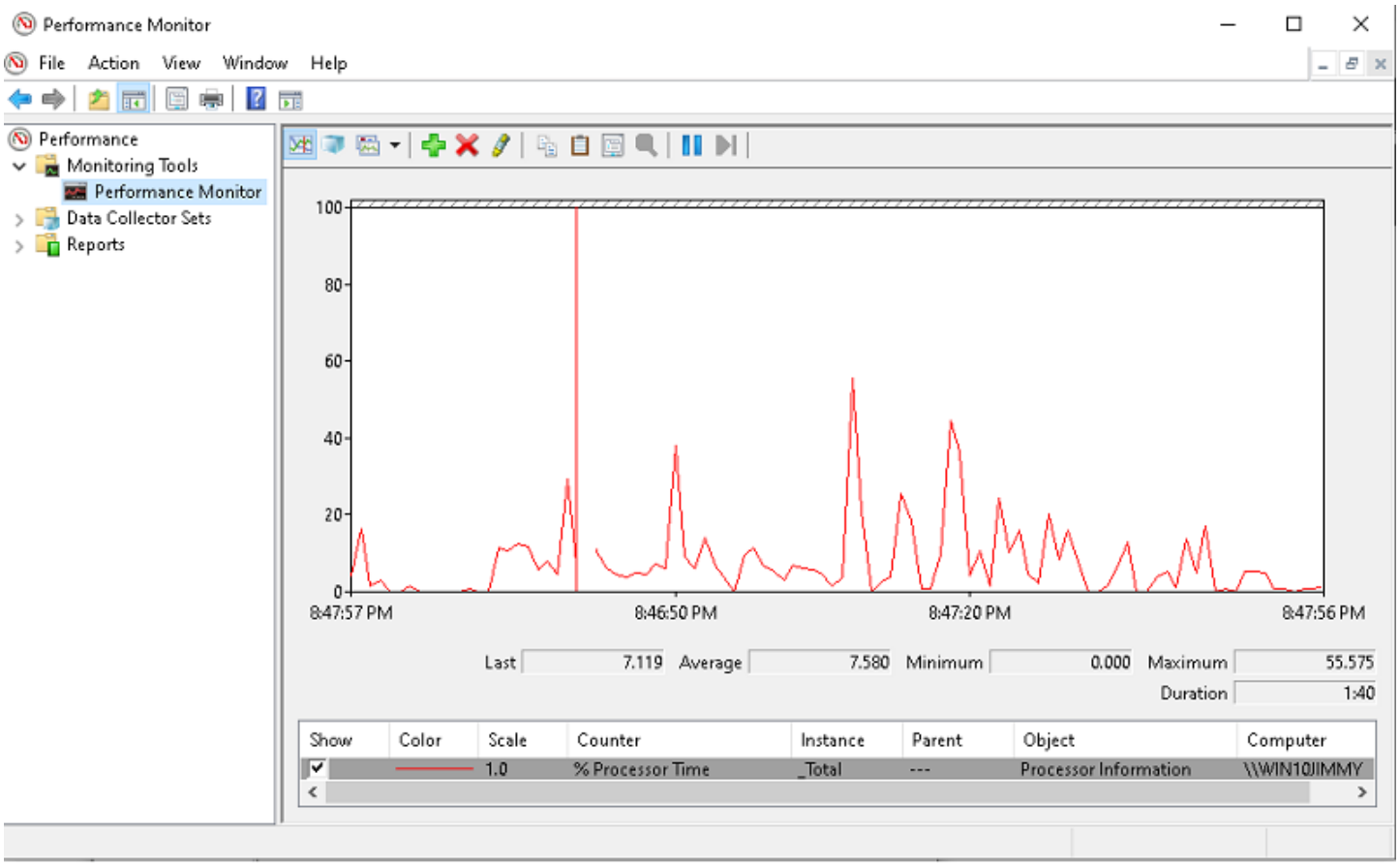
Получает данные из счетчика производительности (performance counter Сведения о счетчиках производительности) через интерфейс PDH. Библиотека `pdh.dll` должна быть установлена в системе. PDH.DLL является библиотекой Windows, если она недоступна, вы должны установить инструмент анализа производительности Windows®, который обычно включен по умолчанию.

```
module_begin
```

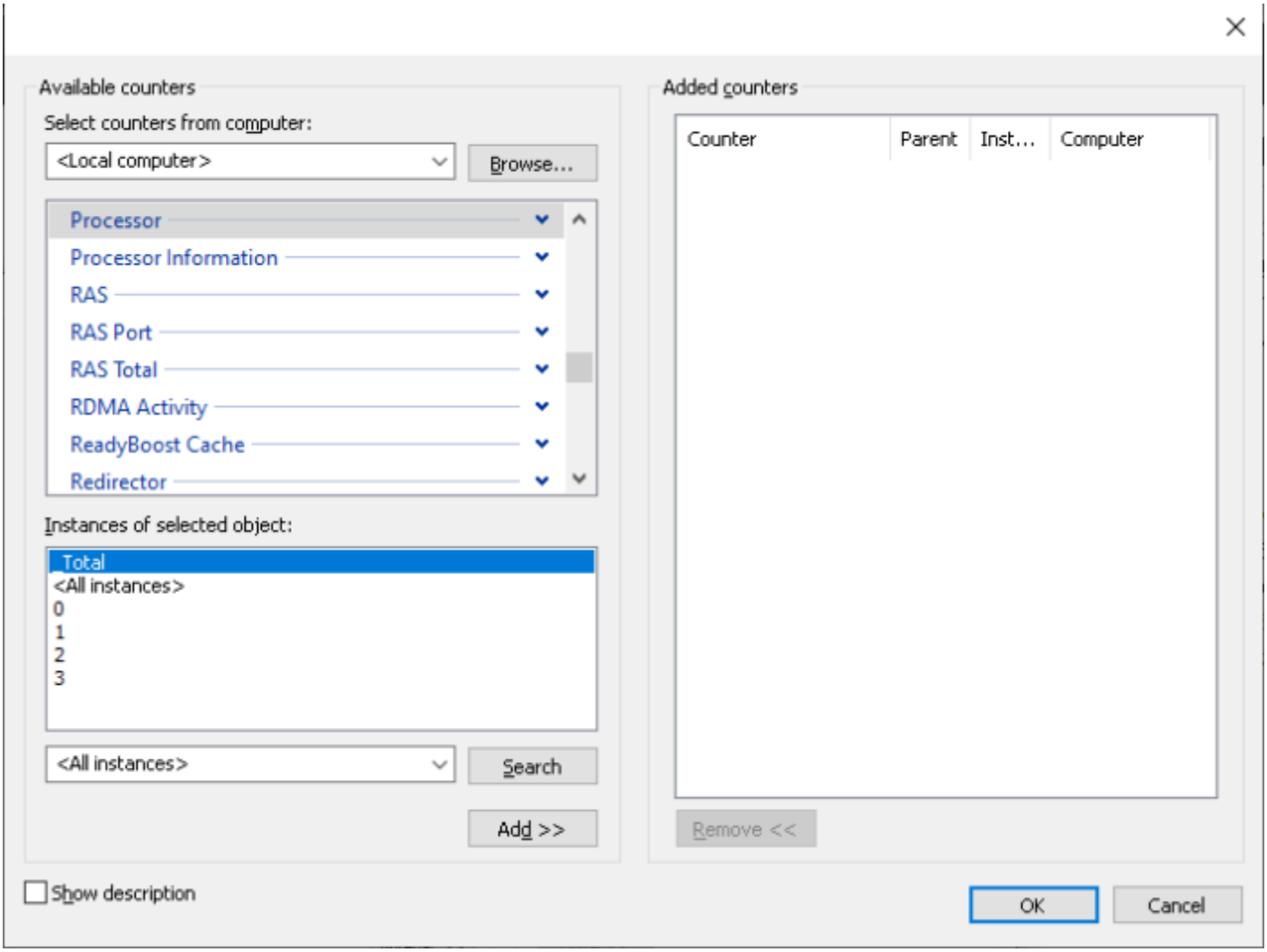
```
module_name perfcounter
module_type generic_data
module_perfcounter \Memory\Pages/sec
module_end
```

Устройство проверки производительности Windows® является очень мощным инструментом и имеет сотни параметров, которые могут быть использованы для мониторинга. Кроме того, каждый производитель устанавливает свои собственные счетчики.

Вы можете наблюдать за счетчиками производительности с помощью инструмента Производительность (*Performance*):



Новые счетчики производительности можно добавить с помощью инструмента системы. Его конфигурация имеет иерархическую структуру с элементами и подэлементами. В данном случае *Процессор*, *% процессорного времени*. и *_Всего*:



Конфигурация модуля для этой конкретной проверки следующая:

```
module_begin
 module_name Processor_Time
 module_type generic_data_inc
 module_perfcounter \Процессор(_Всего)\% процессорного времени
 module_end
```

По умолчанию отображается необработанное значение счетчика, для получения *приготовленного* значения можно добавить параметр module_cooked 1:

```
module_begin
 module_name Disk_E/S_Seg
 module_type generic_data
 module_cooked 1
 module_perfcounter \Физический диск (_Всего)\E/S  разделить на сек.
 module_end
```

Большая часть возвращаемых им данных - это счетчики, поэтому в качестве типа данных следует использовать generic_data_inc. Он также может возвращать значения в очень больших масштабах данных (несколько миллионов), так что вы можете уменьшить эти значения, используя постобработку модуля, со значениями типа 0.000001 или

аналогичными.

module_inventory

В настоящее время эта функциональность заменена на инвентаризацию от plugins Агента как в системах Windows®, так и на Linux/Unix®.

module_logevent

Только для MS Windows®. Позволяет получить информацию из *журнала* событий Windows® на основе указанных шаблонов, позволяя фильтровать информацию по источнику и типу события.

Общий формат этого модуля следующий:

```
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
```

Чтобы избежать показа повторяющейся информации, учитываются только события, произошедшие с момента последнего выполнения Агента.

`module_logevent` принимает следующие параметры, все из которых требуют правильного ввода заглавных и строчных букв (//case-sensitive)//:

- module_source: Происхождение события (System, Application, Security). Обязательное поле.
- module_eventtype: Тип события (error, information...). Необязательное поле.
- module_pattern: Образец для поиска (подстрока). Необязательное поле.
- module_eventcode: Идентификационный номер события. Необязательное поле.
- module_application: Приложение, которое инициирует событие, зарегистрированное в *журнале*. Нужно понимать отличие от `module_source`, который указывает имя источника или от *log* файла, в котором происходит поиск событий.

Например, чтобы отобразить все системные события типа error:

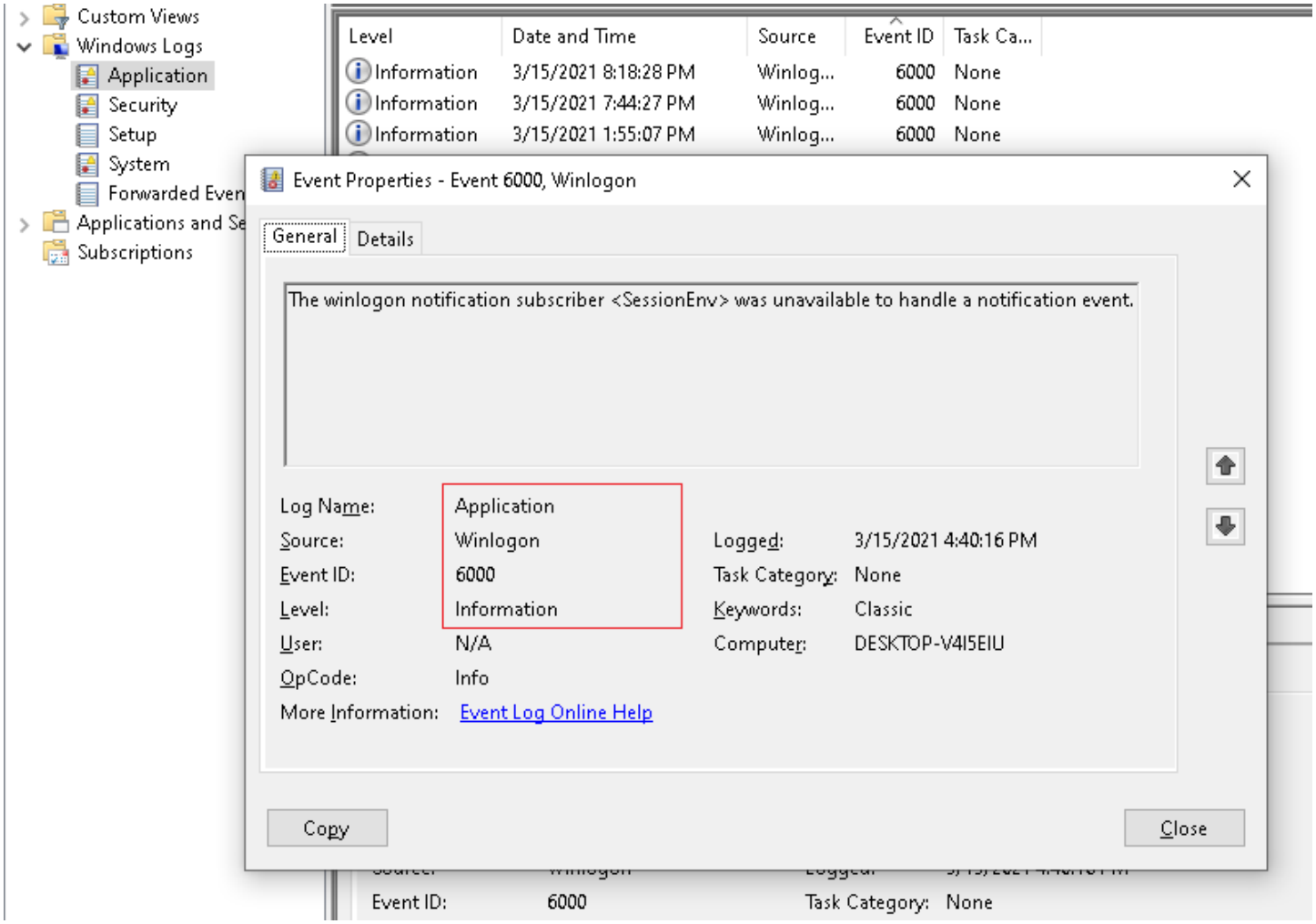
```
module_begin
```

```
 module_name log_events
 module_type generic_data_string
 module_description System errors
 module_logevent
 module_source System
 module_eventtype error
 module_end
```

Чтобы отобразить все события, содержащие слово PandoraAgent>

```
module_begin
 module_name log_events_pandora
 module_type async_string
 module_description PandoraAgent related events
 module_logevent
 module_source System
 module_pattern PandoraAgent
 module_end
```

Пример фильтрации следующего события:




```
module_begin
```

```
module_name MyEvent
module_type async_string
module_logevent
module_source Application
module_eventtype Information
module_eventcode 6000
module_application Winlogon
module_pattern unavailable to handle
module_description
module_end
```

module_logchannel

Версия NG 715 или выше, только для MS Windows®.

Тип модуля, позволяющего получать информацию из каналов *logs* Windows®. Хотя `module_logevent` имеет доступ только к *журналам* Windows Logs®, этот тип модуля позволяет извлекать данные из других файлов *журналов*, которые настроены как каналы. Таким образом, можно получить *журналы*, включенные в Реестры приложений и служб.

Общий формат этого модуля следующий:

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

Чтобы избежать повторного отображения информации, учитываются только те события, которые произошли с момента запуска Агента.

`module_logchannel` принимает следующие параметры, все из которых требуют правильного ввода заглавных и строчных букв (case-sensitive):

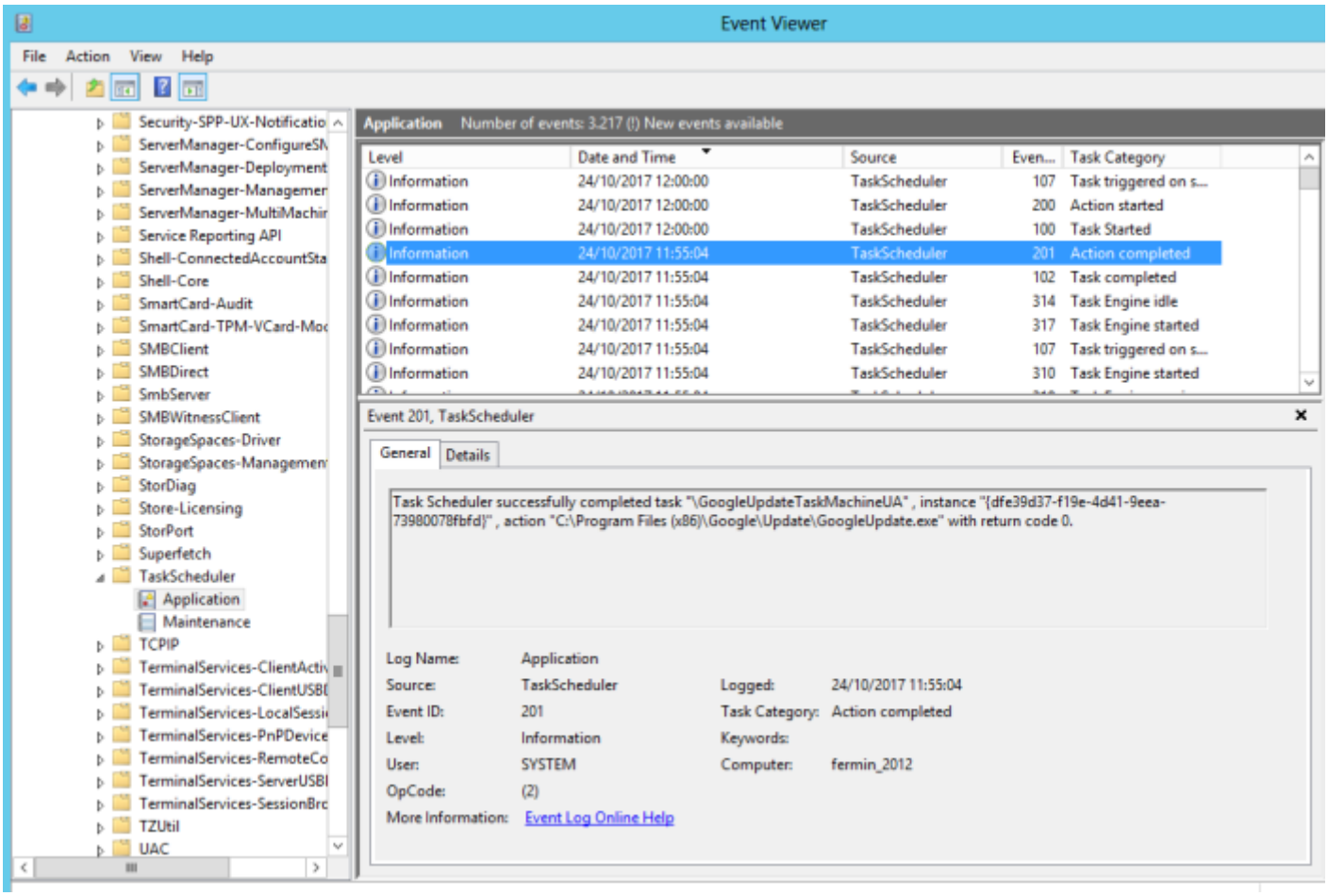
- module_source: Канал события С помощью команды `wevtutil.exe enum-logs` вы получите список всех локальных каналов *logs* устройства. Обязательное поле.
- module_eventtype: Тип события (`critical`, `error`, `warning`, `info` о `verbose`). Необязательное поле.
- module_pattern: Образец для поиска (подстрока). Необязательное поле.
- module_eventcode: Идентификационный номер события. Необязательное поле.

- module_application: Применение происхождения события. Нужно понимать отличие от `module_source`, который указывает имя источника или от *log* файла, в котором происходит поиск событий.

Например, для отображения всех событий канала используется следующий модуль *Microsoft-Windows-TaskScheduler/Operational*, типа *information*, с кодом *201* и текст журнала должен содержать текст *code 0*:

```
module_begin
 module_name New logs
 module_type async_string
 module_logchannel
 module_description Successfully completed tasks
 module_source Microsoft-Windows-TaskScheduler/Operational
 module_eventtype information
 module_eventcode 201
 module_pattern code 0
 module_end
```

При такой конфигурации модуля агент Pandora FMS может принимать следующий журнал *log*:



Чтобы получить имя канала события, нужно щелкнуть на нем правой кнопкой мыши, выбрать Свойства и

скопировать параметр Full name, который необходим для события. `module_source`.

module_plugin

Для выполнения *плагинов* Агента. Это особый случай, поскольку он не требует никаких других тегов типа `module_begin` или `module_end` и не требует указывать тип модуля.

Синтаксис с соответствующими параметрами:

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


</file>

Однако его также можно использовать между обычными тегами модуля для добавления дополнительных опций, таких как условия или интервал:

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


</file>

Для каждого //плагина//  используются свои параметры, поэтому необходимо обратиться к уточняющей документации. Для описания работы одного из //плагинов//, поставляемых по умолчанию с Агентом, в качестве примера можно привести //плагин//  ''grep_log''  для поиска совпадений в файле:

<code>module_plugin grep_log /var/log/syslog Syslog ssh
```

В этом примере имя плагина, «grep_log» и он будет искать в файле »/var/log/syslog« регулярное выражение «ssh» и сохранит его в модуле под названием «Syslog».

Пример вызова *плагина* в Агенте Windows:

```
module_plugin cscript.exe //B "%ProgramFiles%\Pandora_Agent\util\df_percent.vbs"
```

Вы можете узнать больше из обучающего видеоролика «Мониторинг с помощью плагина агента».

module_ping

Только для Windows®.

```
module_ping <host>
```

Этот модуль выполняет ping указанного хоста или *host* и возвращает 1 если находится в сети.

Параметры конфигурации:

- module_ping_count x: Количество пакетов ECHO_REQUEST для отправки (1 по умолчанию).
- module_ping_timeout x: Таймаут в миллисекундах для ожидания каждого ответа (по умолчанию 1000).
- module_advanced_options: Расширенные опции для `ping.exe`.

Пример:

```
module_begin
 module_name Ping
 module_type generic_proc
 module_ping 192.168.1.1
 module_ping_count 2
 module_ping_timeout 500
 module_end
```

module_snmpget

Только для MS Windows®.

```
module_snmpget
```

Этот модуль выполняет запрос SNMP `get` и возвращает запрошенное значение. Параметры конфигурации должны быть указаны в последующих строках таким образом:

- module_snmpversion [1_2c_3]: Версия SNMP (1 по умолчанию).
- module_snmp_community <community>: Сообщество SNMP (*public* по умолчанию).
- module_snmp_agent <host>: Целевой Агент SNMP.
- module_snmp_oid <oid>: Целевой OID.
- module_advanced_options: Расширенные опции для `snmpget.exe`.

Пример:

```
module_begin
 module_name SNMP get
 module_type generic_data
 module_snmpget
```

```
module_snmpversion 1
module_snmp_community public
module_snmp_agent 192.168.1.1
module_snmp_oid .1.3.6.1.2.1.2.2.1.1.148
module_end
```

module_wait_timeout

Только для MS Windows®.

```
module_wait_timeout X
```

Время истечения (таймаут), используемое при проверке выхода модулей module_exec и module_plugin. Значение по умолчанию 500 миллисекунд. Измените значение на 5, если выполнение модуля, генерирующего большое количество выходных данных, происходит медленно. Во всех остальных случаях рекомендуется не использовать.

Автоматическая настройка Агентов

Введение

Версия NG 725 или выше.

В процессе автоконфигурации агента вы можете задать ряд правил для автоматической настройки, и это работает следующим образом:

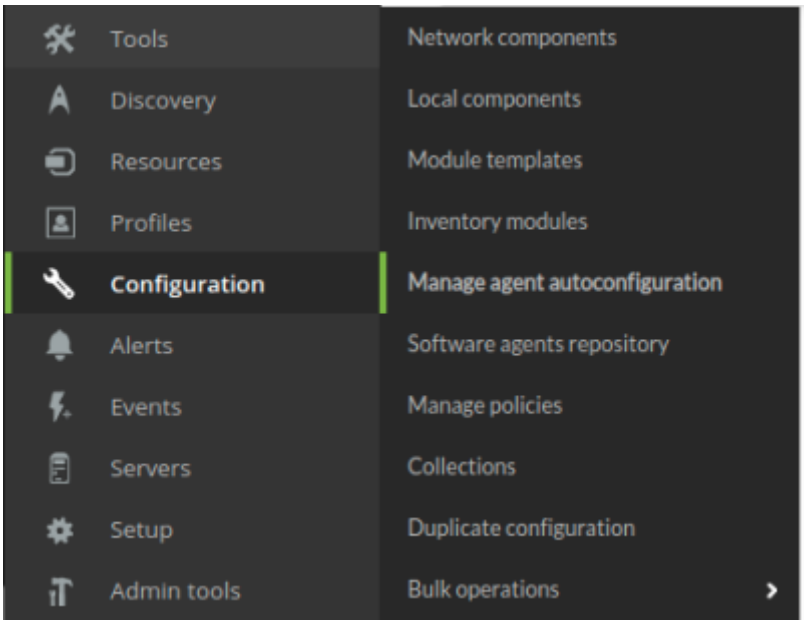
1. Подготовьте автоматические конфигурации в Консоли Pandora FMS или в Метаконсоли Pandora FMS.
2. Установите Агентов, сообщая об этом в вашу Pandora FMS (если у вас есть Метаконсоль с настроенной системой автоматической инициализации, установите в качестве сервера саму Метаконсоль).
3. Pandora FMS Server впервые получит XML (`.data`) с данными Агента.
4. Правила будут оцениваться для определения автоматической конфигурации, которая будет применяться./li>
5. Агент примет новую конфигурацию и сообщит об этом в следующем цикле с обновленной конфигурацией.

Создание/редактирование автоконфигурации

Консоль

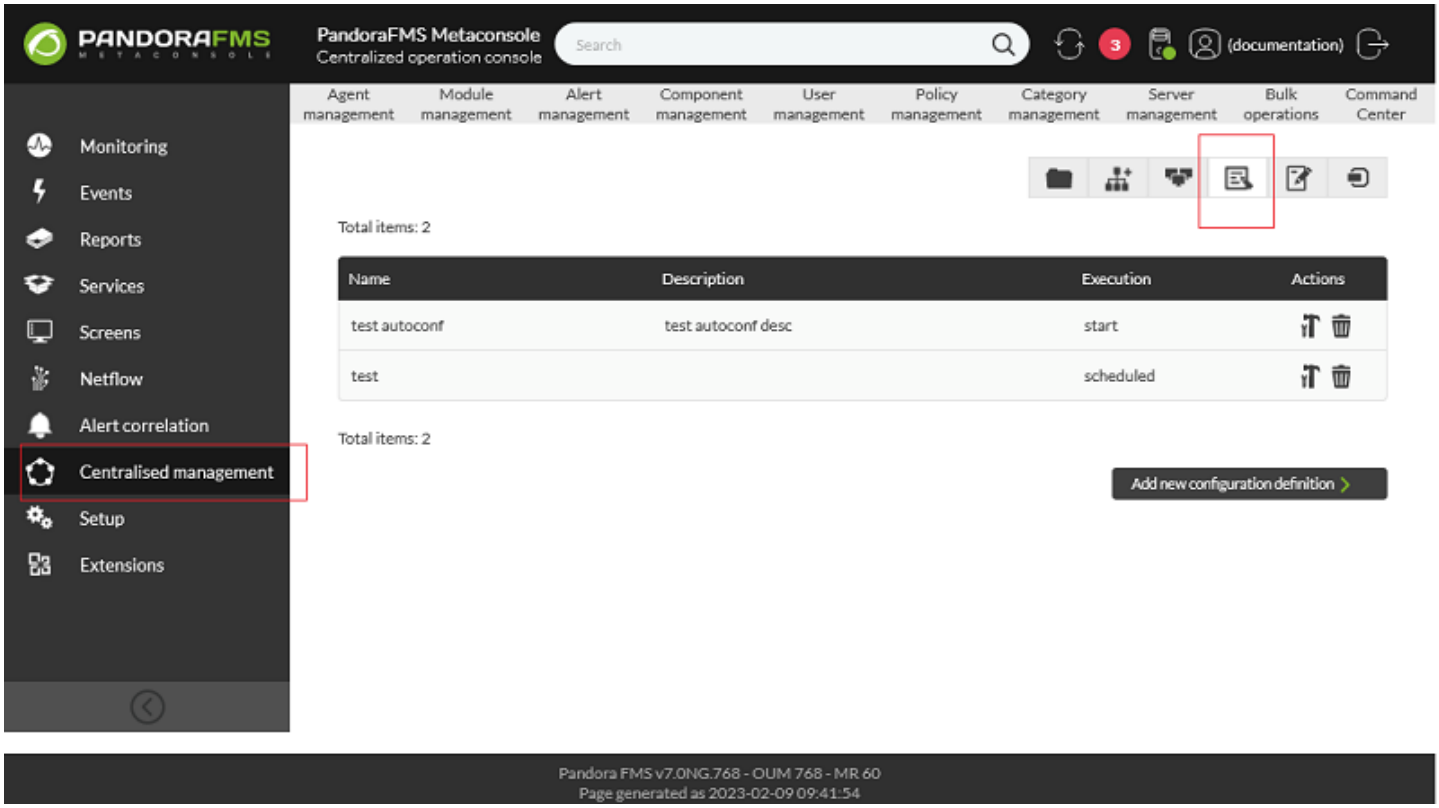
Доступ к управлению автоконфигурациями через Configuration → Manage agent

autoconfiguration:

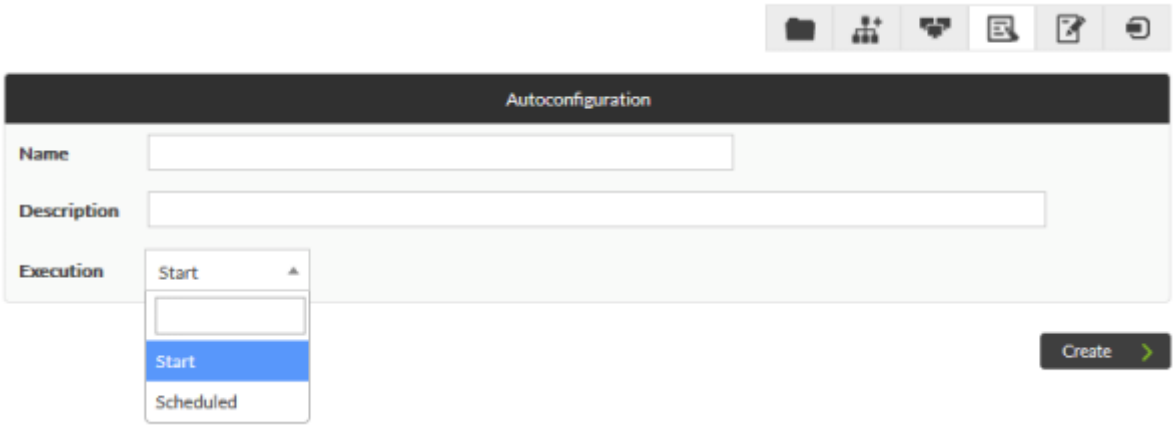


Метаконсоль

Перейдите в раздел Advanced → Agent management → значок конфигурации автоматического агента:



Получив доступ к странице администрирования, вы можете создавать новые автоматические конфигурации, нажав кнопку «Создать автоматической конфигурации» (Add new configuration definition). Вы должны выбрать имя и описание для вашей автоматической конфигурации.

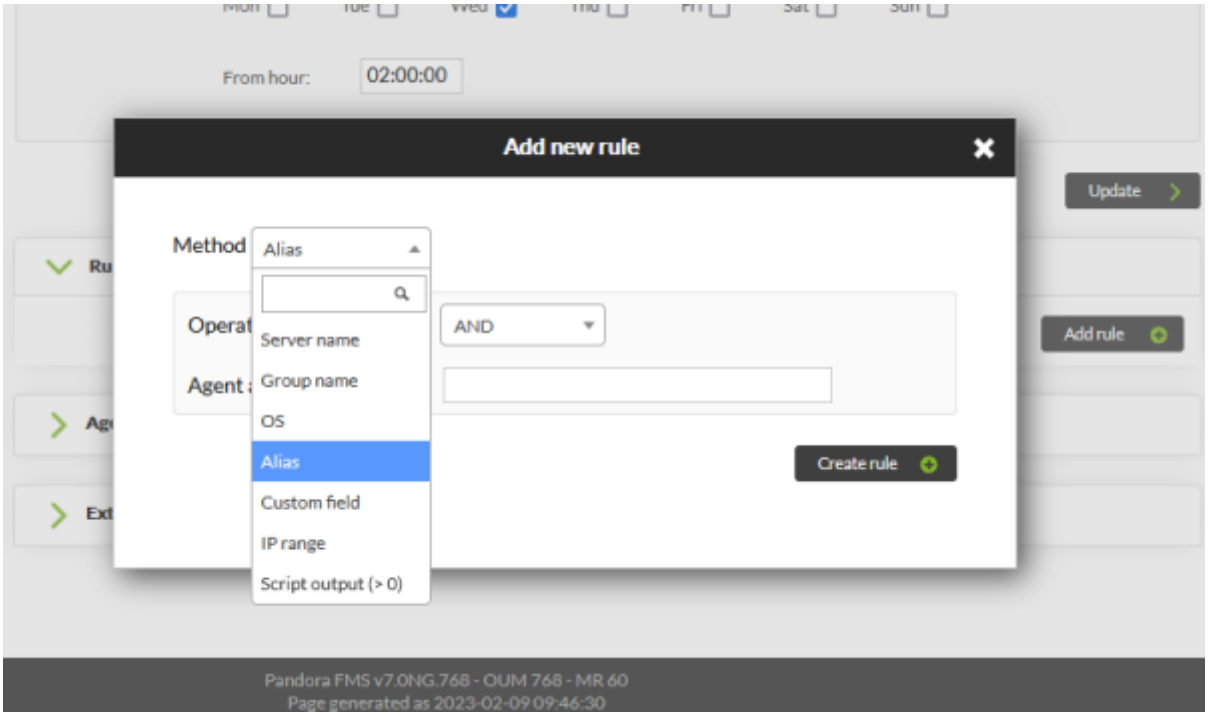


После создания новой автоматической конфигурации вы можете показать формы конфигурации, нажав на нужный вам раздел:

Правила

Чтобы определить Агенты, к которым будет применяться автоматическая конфигурация, вы можете сначала добавить правила для их идентификации.

Откройте раздел правил в автоматической конфигурации и выберите «Добавить новое правило» (Add new rule). Вы сможете выбрать в селекторе правил ряд опций, чтобы определить агентов, которые будут настроены.



Server name

Совпадение в имени сервера.

Group name

Совпадение в имени группы.

OS

Сопоставление имен операционных систем с помощью регулярных выражений.

Custom field

Сопоставление по ключу/значению на основе пользовательского поля, о котором сообщает агент. Укажите имя пользовательского поля и значение, которое оно должно иметь.

IP range

Сопоставление по диапазону IP-адресов (сеть), используйте примечание IP/маска, например:

```
192.168.1.0/24
```

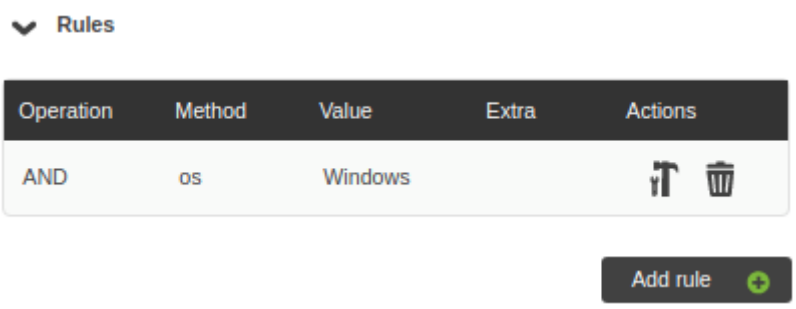
Script output (> 0)

Предназначен для запуска *скрипта*, результат выполнения которого оценивается как действительный, когда стандартный вывод больше 0.

Вызов *скрипта* правил.

Поддерживает следующие макросы в поле 'аргументы' (вы можете выбирать между операторами AND и OR для изменения логики правил):

- _agent_ : Заменяется именем агента.
- _agentalias_ : Будет заменено псевдонимом (алиас) агента.
- _address_ : Будет заменено основным IP-адресом, сообщенным Агентом.
- _agentgroup_ : Будет заменено именем группы, о которой сообщил агент.
- _agentos_ : Будет заменено операционной системой агента.



Если вы не добавите никаких правил, автоматическая конфигурация не будет применена;. Если вам нужна

единая конфигурация для всех агентов, вы можете использовать следующее регулярное выражение для соответствия любому из них *alias*: `.*`

Конфигурации

В этом разделе вы можете настроить:

Группа Агента

Вы можете оставить ее неизменной или принудительно определить ее.

Вторичные группы

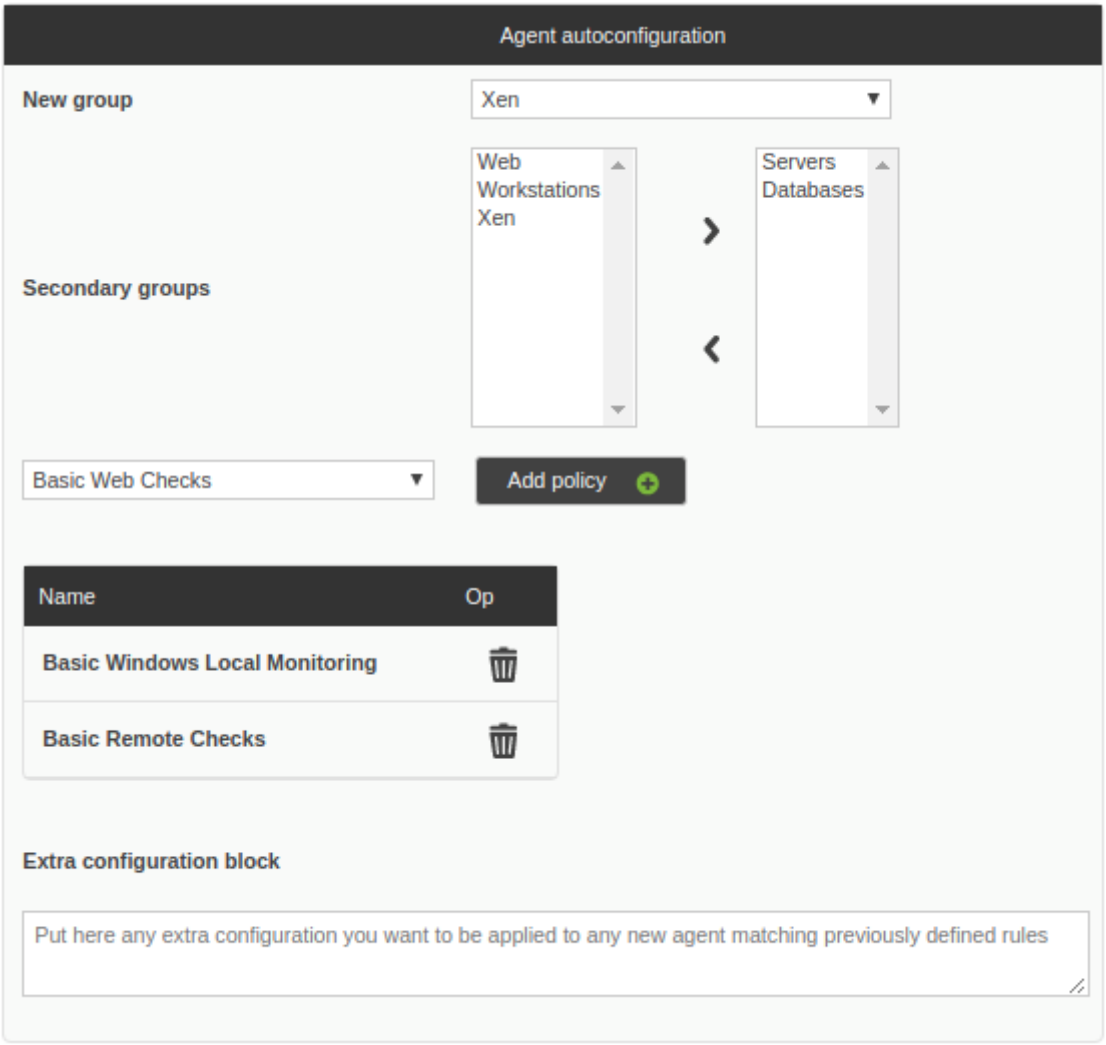
Группы, выбранные здесь, будут добавлены в качестве подгрупп к Агенту.

Политики

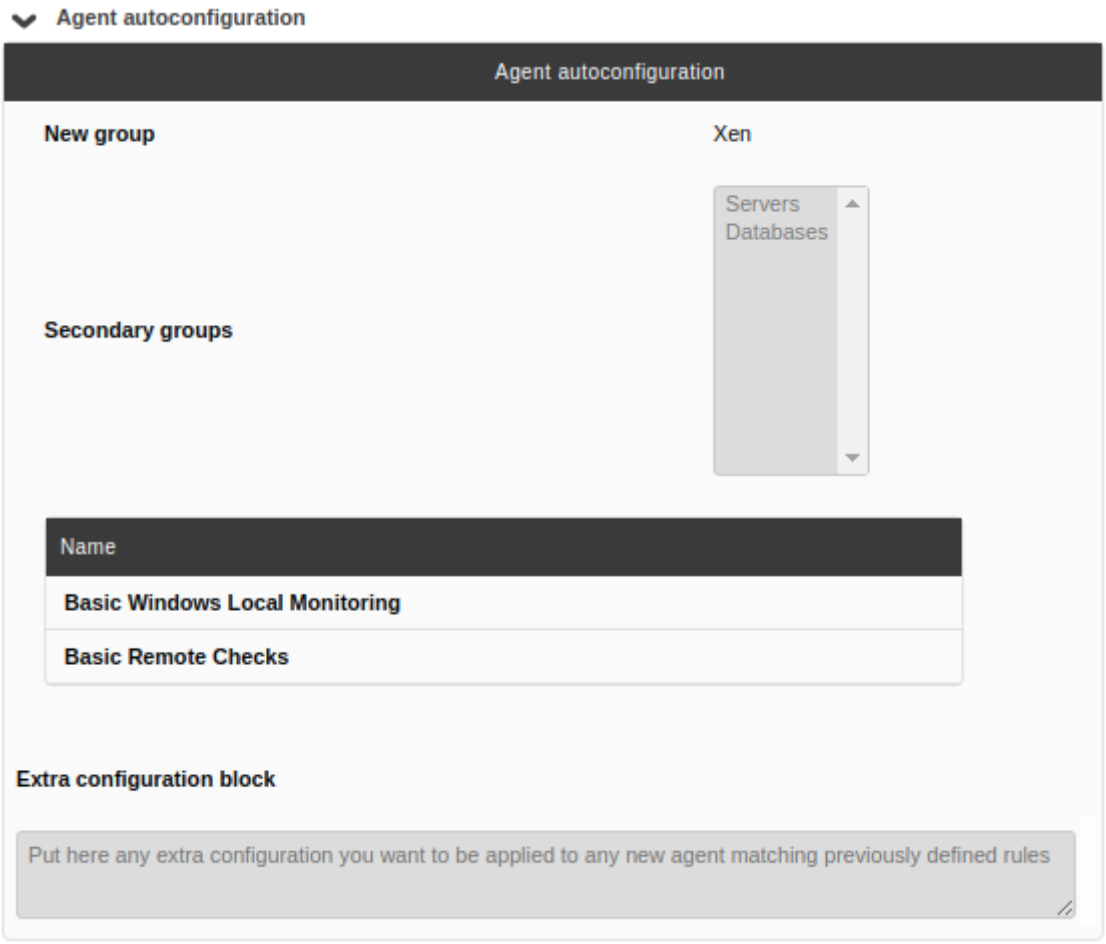
Вы можете выбрать политики, которые будут применяться автоматически, когда агент достигнет сервера.

Блок конфигурации

Добавляет дополнительную необработанную конфигурацию в файл конфигурации агента.



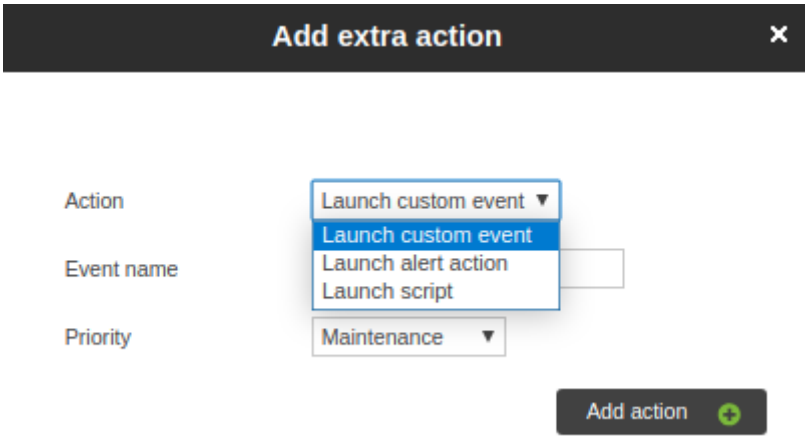
Примечание: Если вы попытаетесь получить доступ к автоматическому администрированию конфигурации с узла, который принадлежит Метаконсоли, с активным централизованным администрированием, его показ будет доступен только для чтения:



Дополнительные действия

В этом разделе вы можете связать другие действия с автоконфигурацией, например:

1. Запустить пользовательское событие (Launch custom event).
2. Выполнение действия предупреждения(Launch alert action).
3. Выполнить сценарий или *script* (Launch script).



Система поддерживает следующие макросы:

agent

Будет заменено именем Агента.

agentalias

Будет заменено псевдонимом агента.

address

Будет заменено на основной IP-адрес, сообщенный агентом.

agentgroup

Будет заменено именем группы, о которой сообщил агент.

agentos

Будет заменено операционной системой Агента.

agentid

Заменяется идентификатором агента.

Агенты Unix/Linux

Конфигурация Агентов Unix Pandora FMS

Основными путями и каталогами, которые необходимо учитывать, являются:

- `/usr/share/pandora_agent` : Где установлен агент Pandora FMS. В системах, где это не разрешено политикой, рекомендуется создать ссылку на этот путь из фактического пути установки, например `/opt/pandora` → `/usr/share/pandora_agent`.
- `/etc/pandora/pandora_agent.conf` : Основной конфигурационный файл агента. Здесь настраиваются модули локального исполнения и *плагины* Агента;.
- `/usr/local/bin/pandora_agent>` Исполняемый двоичный файл агента. Как правило, есть ссылка на `/usr/bin/pandora_agent`.
- `/usr/local/bin/tentacle_client>` Исполняемый двоичный файл Tentacle, для передачи файлов на сервер. Как правило, есть ссылка на `/usr/bin/tentacle_client`.
- `/etc/init.d/pandora_agent_daemon>` *Скрипт* запуска/остановки/перезапуска. В системах AIX демоном является`/etc/rc.pandora_agent_daemon`.
- `/var/log/pandora/pandora_agent.log>` Текстовый файл, в котором сохраняется активность агента Pandora FMS, когда агент выполняется в режиме отладки.
- `/etc/pandora/plugins>` Каталог, содержащий *плагины* Агента. Он связан с каталогом ссылкой `/usr/share/pandora_agent/plugins`.
- `/etc/pandora/collections>` Каталог, содержащий коллекции, развернутые в Агенте. Связан с каталогом ссылкой `/usr/share/pandora_agent/collections`.

Первоначальное выполнение агента Unix

Для запуска агента достаточно запустить:

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

Чтобы остановить Агента, выполните:

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

Этот *скрипт* запуска может запустить или остановить Агента Pandora FMS, который при запуске будет по умолчанию запущен в системе как демон.

Изменение способа получения системной информации агентами Unix

Есть несколько модулей, которые получают информацию предопределенным образом без необходимости указывать команду с `module_exec`. К этим модулям относятся:

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

Можно изменить работу этих модулей по умолчанию, отредактировав непосредственно исполняемый файл агента (по умолчанию `/usr/bin/pandora_agent`). Агент Pandora FMS обычно находится по адресу `/usr/bin/pandora_agent`.

Найдите строку `Commands to retrieve` , которая содержит код, содержащий внутренние команды. Вы можете внести любые необходимые изменения, чтобы адаптировать их к системе.

```
# 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}\
};
```

Чтобы изменить любую из предопределенных команд, просто отредактируйте код для изменения команды, но будьте внимательны к следующим аспектам:

1. Убедитесь, что блоки `{ };` всегда заканчиваются точкой с запятой.
2. Убедитесь, что команды заключены в одинарные кавычки: `' '` .
3. Внутри кавычек может понадобиться дополнительная кавычка с `` ` `` (см. пример выше).
4. Убедитесь, что все одинарные кавычки, которые вы хотите использовать в команде, предварительно заключены в символ , т.е. '. Например, эта команда, которая обычно выглядит следующим образом:

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

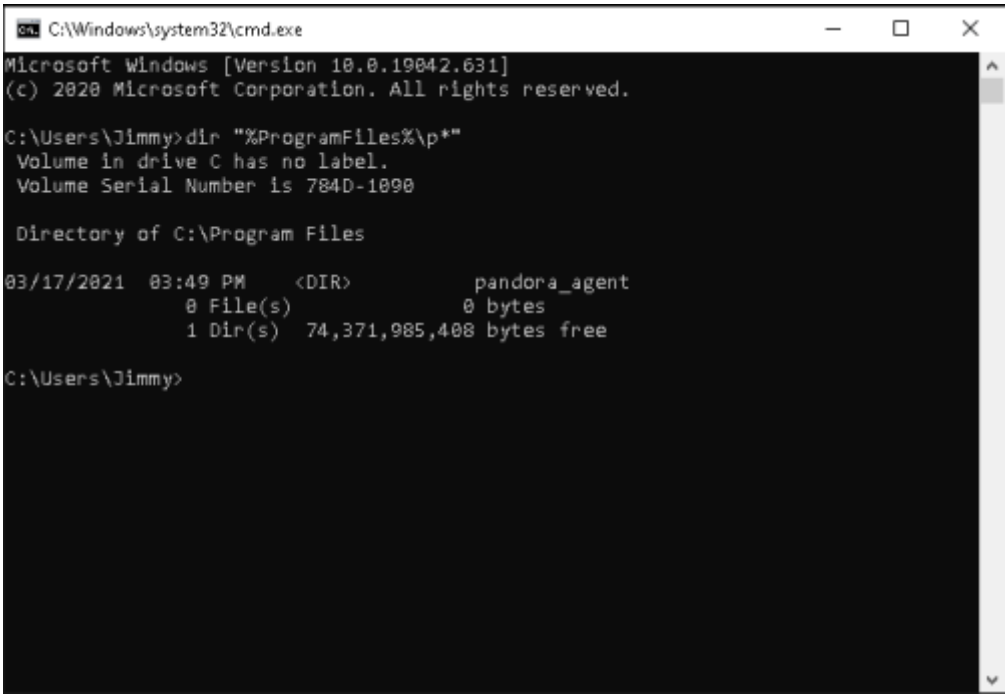
Вы должны написать ее как:

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

Агенты Windows Pandora FMS

Конфигурация Агента для Windows Pandora FMS

Основные пути и каталоги в установках Агента для MS Windows® по умолчанию находятся в том же каталоге, где установлен Агент `%ProgramFiles%`.



Наиболее важными для учета являются:

`%ProgramFiles%\pandora_agent`

То место, где установлены Pandora FMS Agent, его исполняемый файл и каталоги.

`%ProgramFiles%\pandora_agent\pandora_agent.conf`

Основной конфигурационный файл агента. Здесь настраиваются модули локального исполнения и *плагины* Агента;.

`%ProgramFiles%\pandora_agent\PandoraAgent.exe`

Исполняемый двоичный файл агента.

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

Исполняемый двоичный файл Tentacle, для передачи файлов на сервер.

`%ProgramFiles%\pandora_agent\scripts`

Скрипты запуска/остановки/перезапуска Агента Pandora FMS.

`%ProgramFiles%\pandora_agent\pandora_agent.log`

Текстовый файл, в котором сохраняется активность Агента Pandora FMS, когда агент находится в режиме отладки.

`%ProgramFiles%\pandora_agent\util`

Каталог, содержащий *плагины* агента.

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

Каталог, содержащий коллекции агента.

Автоматическое развертывание программных агентов

Вы можете развернуть программные агенты с помощью центра развертывания через систему Discovery, более подробная информация по этой ссылке.

Автоматическое обновление Программных Агентов

Используя коллекции файлов и `pandora_update`, вы можете обеспечить «автоматическое обновление» Программных агентов.

Инструменту `pandora_update` для работы необходим модуль Perl `Digest::MD5`. Начиная с версии Perl 5.14, этот модуль встроен по умолчанию, но в более ранних версиях вам придется установить его вручную.

Это работает следующим образом:

1. Агенты получают новые двоичные файлы во входной каталог коллекций.

Пример в Windows®:

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

Пример в Linux®:

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

2. Агент выполняет *плагин*. `pandora_update`. Этот плагин получает один параметр: краткое имя коллекции (в данном примере, `fc_1`). Он просканирует каталог коллекции на наличие двоичного файла агента (не весь инсталлятор), сравнит двоичный файл, находящийся в коллекции, с запущенным в данный момент, и, если они отличаются, `pandora_update` остановит Агента, заменяет двоичный файл и перезапускает Агент снова, используя новый двоичный файл.

Для обновления различных архитектур необходимо создать отдельную коллекцию для каждой архитектуры. Например, если вы хотите обновить агенты Windows® 32 и 64 бит, вам следует создать две коллекции и в каждую из них включить соответствующий бинарный файл `PandoraAgent.exe`.

3. Pandora_update также записывает в небольшой *журнал* обновленное событие, чтобы иметь возможность получить его при следующем выполнении и предупредить пользователя (с помощью модуля async_string) о процессе обновления агента.

Это означает, что модули, используемые для завершения процесса обновления, могут быть настроены на высокий интервал.

Стандартная установка Unix

```
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
```

Индивидуальная установка Unix

```
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
```

Команда pandora_update принимает в качестве второго параметра путь к каталогу установки Pandora FMS, его нет необходимости указывать, если установка была произведена по пути по умолчанию.

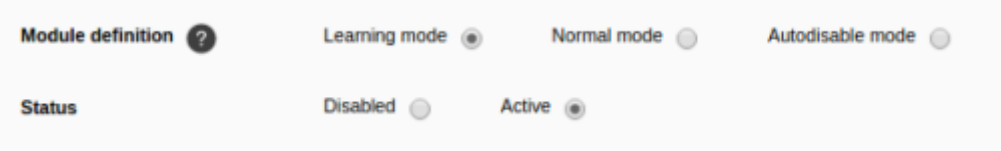
Windows®

```
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
```

Автосоздание Агентов и Модулей из XML

Агенты могут быть настроены с консоли в трех рабочих режимах:

- Режим обучения: Если XML, полученный от Программного Агента, содержит новые модули, они будут созданы автоматически. Это поведение по умолчанию.
- Нормальный режим: Новые модули, поступающие в XML, не будут созданы, если они не были ранее заявлены в консоли.
- Режим автоматического отключения: Подобно режиму *обучения*, в этом режиме, кроме того, если все модули перейдут в неизвестный статус, Агент будет отключен автоматически, и он будет снова включен, если получит новую информацию.



Данные, которые учитываются при создании Агента

Данные, которые автоматически включаются в Агент в момент его создания при получении XML, следующие:

- Имя Агента.
- IP-адрес Агента.
- Описание Агента.
- Родительский Агент.
- *Timezone offset*.
- Группа.
- Операционная система.
- Интервал.
- *Agent version*.
- *Custom fields*.
- *Custom ID*.
- URL Адрес.
- Режим агента: Обучение, Нормальный, Автоотключение.

Данные, которые будут обновлены у агента при получении XML (при включенном режиме обучения)

- IP-адрес Агента.
- Отец агента.
- Версия ОС.
- Версия Агента.
- Часовой пояс.
- *Custom fields*.

Данные ГИС всегда обновляются (если включено),

независимо от того, отключен ли *режим обучения*.

Кроме того, при активированном *режиме обучения* новые модули будут создаваться в Pandora FMS при их получении через XML.

Данные, которые включаются в создание модуля

Данные, которые включаются в модуль при первом получении XML, следующие:

- Имя.
- Тип.
- Описание.
- Максимальный, минимальный.
- Постпроцесс.
- Интервал Модуля.
- *Min/max Critical*.
- *Min/max Warning*.
- Деактивировано.
- Единицы.
- *Module group*.
- *Custom ID*.
- *Str. Warning/Critical*.
- Инструкции для критического состояния
- Инструкции для предупреждения.
- Инструкции для неизвестного состояния.
- *Теги*.
- Инверсия критического состояния.
- Инверсия состояния предупреждения.
- Режим «тихий».
- *Min. FF Threshold*.
- *Alert template*.
- Crontab.

Данные, которые обновляются из существующего модуля при получении XML

При получении XML, содержащего информацию из существующего модуля, обновляется только описание и расширенная информация, в дополнение к данным модуля.

Вернуться в оглавление Pandora FMS