Configuration des agents logiciels

Configuration des agents logiciels

Qu'est-ce qu'un agent logiciel

Comme leur nom l'indique, ce sont de petits logiciels qui sont installés sur les systèmes d'exploitation et qui restent exécutés dessus pour extraire les informations de supervision et les envoyer régulièrement au serveur Pandora FMS.

Ils utilisent les commandes et les outils du système d'exploitation dans lequel ils sont installés pour obtenir les informations. Ils conforment les données dans un fichier au format XML et l'envoient au serveur de données Pandora FMS, qui les traite et les stocke dans la base de données.

Chacun des contrôles individuels est appelé un *module* .

Introduction à la configuration de l'agent logiciel

Le fonctionnement de l'agent logiciel est déterminé par son fichier de configuration, appelé `pandora_agent.conf`, situé dans le répertoire d'installation (valeur par défaut) `%ProgramFiles%\pandora_agent` sur les systèmes MS Windows® et dans `/etc` sur les systèmes GNU/Linux®. Le fichier de configuration contient tous les paramètres de fonctionnement et les modules de cet agent.

Paramètres généraux de l'agent

Les paramètres généraux de configuration de l'agent logiciel sont définis dans cette section. La plupart sont communs aux systèmes MS Windows® et GNU/Linux®.

Le codage du fichier de configuration de l'agent est UTF-8 sur les systèmes GNU/Linux® et MS Windows®. Si vous apportez des modifications manuelles à ce fichier, vérifiez que l'encodage est correct avant de l'écraser. Si l'encodage n'est pas UTF-8 et que vous utilisez des symboles (tels que des accents ou des symboles étendus), l'agent logiciel les interprétera de manière erronée, ne pouvant garantir une interprétation correcte de votre configuration.

La première fois que des données sont reçues de l'agent logiciel, toutes les informations sont enregistrées dans la base de données. Pour les envois successifs d'informations (et selon que le mode apprentissage est activé) seuls les champs XML suivants seront mis à jour : `version` , `timestamp` (date) et `os_version` (version du système d'exploitation), ainsi que les paramètres suivants du fichier de configuration `gis_exec` , `latitude` , `longitude` , `altitude` , `parent_agent_name` , `timezone_offset` , `address` , `custom_field`

Vous pouvez trouver plus d'informations sur le format de fichier XML Pandora FMS dans ce lien . Vous pouvez également accéder à la génération de données de test (outils d'analyse de capacité).

Sous Unix, pour redémarrer, arrêter ou démarrer l'Agent Logiciel après avoir modifié un paramètre général :

```
/etc/init.d/pandora_agent_daemon restart
```

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

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

server_ip

Adresse IP ou nom du serveur Pandora FMS auquel les données seront envoyées.

server_path

Route du serveur où il reçoit les fichiers de données envoyés par les agents. Par défaut `/var/spool/pandora/data_in` .

temporal

Chemin où l'agent logiciel stocke les fichiers de données avant qu'ils ne soient envoyés au serveur et supprimés localement.

description

Il envoie la description de l'Agent Logiciel dans le XML, et Pandora FMS importe cette description

lorsqu'il crée l'Agent Logique.

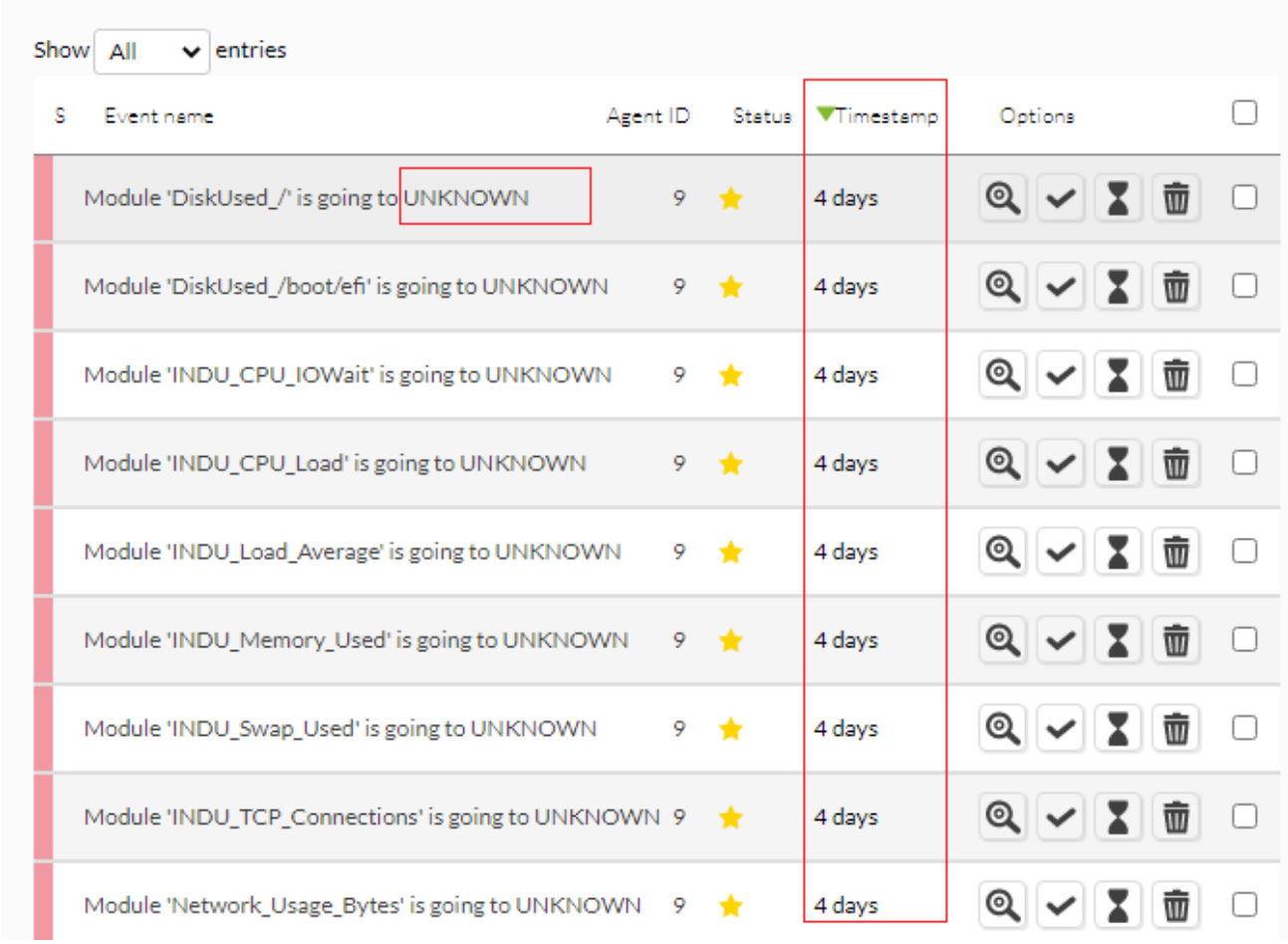
group

S'il existe un groupe portant le nom indiqué dans ce paramètre, l'Agent sera créé au sein de ce groupe sauf si le serveur force la création de tous les agents d'un groupe donné.

temporal_min_size

Si l'espace libre (en mégaoctets) de la partition sur laquelle se trouve le répertoire temporaire est inférieur à cette valeur (par défaut un mégaoctet), la génération des paquets de données est arrêtée . Cela empêche le disque de se remplir si, pour une raison quelconque, la connexion au serveur PFMS principal et/ou secondaire est perdue pendant une période prolongée.

Dans la vue des événements de Pandora FMS, vous pourrez savoir en temps opportun si un agent logiciel, *quelle qu'en soit la cause*, a cessé de fournir des données à son serveur Pandora FMS principal et/ou secondaire, car dans ces cas, son état devient “inconnu” (`UNKNOWN`). Voir aussi, si vous le souhaitez, créer des conditions d'alerte .



Dans cet exemple particulier avec quatre jours hors ligne, les données de supervision seront enregistrées jusqu'à ce qu'il ne reste plus qu'un mégaoctet (ou la valeur que vous avez définie) d'espace disque disponible.

En raison de tout ce qui précède, vous devez toujours garder à l'esprit que lorsque l'agent logiciel parvient enfin à se connecter à votre serveur Pandora FMS principal et/ou secondaire, la quantité de données accumulées peut surcharger le serveur de données Pandora FMS .

temporal_max_size

Taille maximale (en mégaoctets) autorisée pour le *buffer* XML, valeur par défaut `1024`.

Voir également temporal_min_size .

temporal_max_files

Nombre maximum de fichiers autorisés pour le *buffer* XML, valeur par défaut `1024`.

Voir également temporal_min_size.

logfile

Chemin d'accès au *log* de l'agent Pandora FMS.

interval

En secondes, temps d'échantillonnage de l'agent. Chaque fois que cet intervalle est terminé, l'Agent collecte des informations et les envoie au serveur Pandora FMS.

disable_logfile

Pour MS Windows® uniquement : Il désactive l'écriture dans `pandora_agent.log` .

debug

S'il est actif (`1`), les fichiers de données de l'Agent sont stockés et renommés dans le répertoire

temporaire et ne sont pas supprimés après avoir été envoyés au serveur, permettant d'ouvrir les fichiers XML et d'analyser leur contenu.

agent_name

Il vous permet de définir un nom personnalisé. S'il n'est pas activé, le nom de l'agent sera le *hostname* de la machine.

agent_name_cmd

Il définit le nom de l'agent à l'aide d'une commande externe. Si `agent_name_cmd` est défini, `agent_name` est ignoré. La commande doit retourner le nom de l'agent par STDOUT. S'il renvoie plus d'une ligne, seule la première sera utilisée.

agent_alias_cmd

Il définit le nom de l'agent à l'aide d'une commande externe. Si `agent_name_cmd` est défini, `agent_alias` est ignoré. La commande doit retourner le nom de l'agent par STDOUT. S'il renvoie plus d'une ligne, seule la première sera utilisée.

address

Il s'agit de l'adresse IP associée à l'agent logiciel. Il peut s'agir d'un IPv4 au format `X.X.X.X` , d'un nom de domaine comme `localhost` ou `auto`. S'il s'agit d'une adresse IP ou d'un nom de domaine, il sera ajouté à la collection d'adresses de l'agent et défini comme principal. S'il est `auto`, l'adresse IP de la machine sera obtenue et ajoutée à l'agent de la même manière que dans le cas précédent.

encoding

Il installe le type de codage du système local, tel que `ISO-8859-15` ou `UTF-8` .

server_port

Port sur lequel le serveur Pandora FMS Tentacle écoute pour recevoir les fichiers de données, 41121 par défaut.

transfer_mode

Mode de transfert des fichiers de données vers le serveur Pandora FMS. Valeur par défaut `tentacle` .

transfer_timeout

Délai d'expiration (*timeout*) pour le transfert de fichiers ; Si le nombre de secondes indiqué est dépassé sans terminer le transfert, celui-ci sera annulé.

server_pwd

Mot de passe du serveur pour l'authentification : Spécifique pour Windows® FTP et le mode de transfert Tentacle, bien que le mot de passe dans ce dernier soit facultatif.

server_ssl

Spécifique au mode de transfert Tentacle. Il permet d'activer (`yes`) ou désactiver (`no`) le chiffrement des connexions à l'aide de SSL. Vous pouvez en savoir plus sur la communication sécurisée avec Tentacle dans cette section.

server_opts

Pour ajouter des options supplémentaires lors de l'exécution du client Tentacle dans le transfert de fichiers. Utilisé pour les configurations avancées de Tentacle avec options de sécurité.

Depuis la version 3.2 des agents, le client Tentacle prend en charge une option permettant d'utiliser un proxy HTTP pour envoyer les données au serveur. La méthode CONNECT doit être activée sur ce proxy HTTP. Pour pouvoir utiliser la sortie via un *proxy*, utilisez l'option suivante (par exemple) :

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

Cette option force le client Tentacle à envoyer les données via un proxy situé sur `proxy.inet` et utilisant le port `8080`, en utilisant le nom d'utilisateur `user` et le mot de passe `pass` pour s'authentifier auprès de ce *proxy*. Si, par exemple, vous devez utiliser un *proxy* sans authentification, sur un serveur sur `192.168.1.2` et avec le port `9000`, l'option serait la suivante :

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

Vous pouvez en savoir plus sur la communication sécurisée avec Tentacle dans cette section.

delayed_startup

Désactivé par défaut. Délai d'attente (secondes ou minutes) jusqu'à ce que l'agent commence à fonctionner une fois démarré. Pour tous les agents logiciels à l'exception de MS Windows®.

startup_delay

Désactivé par défaut. Délai d'attente en secondes jusqu'à ce que l'agent commence à fonctionner une fois démarré. Pour MS Windows® uniquement.

pandora_nice

Il n'est disponible que pour les agents Unix/Linux. Ce paramètre vous permet de spécifier la priorité que le processus Pandora FMS Agent aura sur le système.

autotime

Si activé (`1`) il envoie un *horodatage* d'exécution spécial (AUTO) qui oblige le serveur à utiliser la date et l'heure locales du serveur pour définir l'heure des données, ignorant l'heure envoyée par l'agent. Ceci est nécessaire dans les agents qui, pour une raison quelconque, ont un temps incorrect ou très différent de celui du serveur.

cron_mode

Avec ce paramètre, il est possible de faire en sorte que l'agent utilise le crontab Linux® pour s'exécuter à un certain intervalle au lieu d'utiliser le propre système interne de l'agent pour s'exécuter de temps en temps. Désactivé par défaut (`0`).

remote_config

Il active (`1`) ou désactive (`0`) la configuration de l'agent distant. Son fonctionnement n'est autorisé qu'avec le mode de transfert Tentacle.

xml_buffer

S'il est activé (1), l'agent logiciel enregistrera dans son répertoire temporaire les fichiers XML qu'il n'a pas pu envoyer au serveur en cas de problème de connectivité. Ils seront envoyés lorsque les communications seront rétablies.

timezone_offset

L'agent logiciel peut très bien installer son fuseau horaire décalé avec le serveur. Cela permet au serveur d'effectuer un décalage de l'heure collectée par l'agent, de sorte qu'elle corresponde à l'heure locale du serveur.

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

Il est calculé en soustrayant le fuseau horaire de l'agent du fuseau horaire du serveur. Par exemple, si le serveur se trouve dans le fuseau horaire UTC+1 et que l'agent est dans le fuseau horaire UTC-5, timezone_offset doit être 6 = 1 - (-5).

parent_agent_name

Il indique le *parent* de l'agent logiciel. Il doit s'agir du nom d'un agent existant dans Pandora FMS.

agent_threads

Il n'est disponible que pour les agents Unix/Linux : Nombre de threads que l'agent lancera pour exécuter des modules en parallèle. Par défaut, les modules s'exécutent les uns après les autres sans lancer de threads supplémentaires. Exemple :

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

include

```
include <file>
```

Il permet d'inclure un fichier (*file*) de configuration supplémentaire. Ce fichier peut inclure des modules et des collections supplémentaires à ceux du fichier principal. Le fichier peut être téléchargé par les utilisateurs qui ont des autorisations d'écriture sur les agents (« AW »).

broker_agent

broker_agent <broker_name>

Il active la fonctionnalité Broker Agent. Pour l'activer, il suffit de supprimer le paramètre des commentaires et d'indiquer le nom (<broker_name>) qui sera attribué à l'agent broker.

pandora_user

```
pandora_user <user>
```

Ce paramètre est facultatif et vous permettra d'exécuter l'agent avec l'utilisateur système (<user>) que vous spécifiez. Cet utilisateur doit disposer des autorisations nécessaires pour exécuter l'agent et ses ressources associées.

custom_id

Identificateur personnalisé de l'agent pour les applications externes.

url_address

URL personnalisée à ouvrir à partir de l'agent dans la console.

custom_fieldX_name

Nom d'un champ personnalisé Agents qui existe déjà dans le système. S'il n'existe pas, il sera ignoré. Exemple :

```
custom_field1_name Model

<code>
```

custom_fieldX_value

Valeur du champ personnalisé custom_fieldX_name définie dans le paramètre précédent. Exemple :

```
custom_field1_value C1700
```

module_macro

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

Il définit une macro d'exécution locale qui peut être utilisée dans la définition d'un module. Ces macros sont utilisées dans le système Métaconsole et dans le système de composants Module local pour « abstraire » la difficulté d'utiliser un Module en éditant directement le code, en présentant à un utilisateur moins avancé une interface locale qui permet de « remplir » les valeurs. Ces valeurs sont utilisées ci-dessous, en utilisant un système de macro, relativement similaire au système de macro des *plugins* locaux.

Les macros d'exécution locale commencent par `_fieldx_`

Exemple :

```
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

```
group_password <password>
```

Mot de passe (*password*) du groupe de l'Agent. Si le groupe n'est pas protégé par mot de passe, vous devez laisser cette ligne sous forme de commentaire.

ehorus_conf

```
ehorus_conf <path>
```

Chemin absolue (*path*) à un fichier de configuration valide d'un agent eHorus. L'agent crée un champ personnalisé appelé `eHorusID` qui contient la clé d'identification de l'agent eHorus.

transfer_mode_user

Version NG 7.0 OUM713 ou supérieure.

```
transfer_mode_user <user>
```

Utilisateur (*user*) des fichiers copiés en mode de transfert local. Dans les dossiers de la console, cet utilisateur doit disposer d'autorisations de lecture et d'écriture pour que la configuration à distance fonctionne correctement. Par défaut, il s'agit d'`apache`.

secondary_groups

Version NG 7.0 OUM721 ou supérieure.

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

Nom des groupes secondaires (*group name*) attribués à l'agent. Vous pouvez spécifier plusieurs groupes enfants séparés par des virgules. Si l'un des groupes n'existe pas sur le serveur auquel les informations sont envoyées, ce groupe ne sera pas attribué, mais la création de l'agent ne sera pas affectée.

standby

Version NG 7.0 OUM728 ou supérieure.

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

Si un agent a le mode veille activé (`standby 1`), l'agent n'effectue aucune vérification et n'envoie ni ne génère de code XML. Cette stratégie de configuration est logique dans les installations d'entreprise où il existe une configuration à distance. Par conséquent, vous pouvez désactiver un agent à volonté en le désactivant simplement.

Le mode de débogage remplace cette fonctionnalité et l'agent s'exécute normalement.

Serveur secondaire

Vous pouvez définir un serveur secondaire vers lequel les données seront envoyées dans deux situations possibles en fonction de la configuration :

- on_error : Il envoie des données au serveur secondaire uniquement s'il ne peut pas les envoyer au serveur principal.
- always: Il envoie toujours des données au serveur secondaire, qu'il puisse ou non contacter le serveur principal.

Exemple de configuration :

```
# 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
```

Serveur UDP

Gardez à l'esprit que UDP est par nature non sécurisé (mais efficace pour envoyer des messages sans compromettre une certaine réponse).

L'agent logiciel Pandora FMS peut être configuré pour écouter les commandes à distance. Ce serveur écoute sur un port UDP spécifié par l'utilisateur et vous permet de recevoir des commandes d'un système distant, généralement à partir de la console Pandora FMS, en exécutant des alertes sur le serveur.

Pour configurer le serveur distant UDP, les options suivantes existent dans votre fichier de configuration `pandora_agent.conf`

- udp_server : Pour activer le serveur UDP, définissez la valeur sur 1. Par défaut, il est désactivé.
- udp_server_port : Numéro de port d'écoute.
- udp_server_auth_address : Adresses IP autorisées à envoyer des commandes. Pour spécifier plusieurs adresses, séparez-les par des virgules. S'il est configuré avec 0.0.0.0, il accepte les commandes de n'importe quelle adresse.

Bien qu'il puisse être réglé sur *0.0.0.0* pour accepter de toutes les sources, cette pratique n'est pas recommandée. Si vous avez plusieurs serveurs Pandora FMS et/ou si vous utilisez IPv6, vous pouvez placer différentes adresses IP séparées par des virgules. Par exemple, si vous avez dans IPv6 :`2001:0db8:0000:130F:0000:0000:087C:140B` et son abréviation est `2001:0db8:0:130F::87C:140B` utilisez les deux séparés par des virgules.

- process_<name>_start <command>: Commande qui démarre un processus défini par l'utilisateur.
- process_<name>_stop <command> : Commande qui arrêtera le processus.
- service_<name> 1 : Il permet d'arrêter le service <name> qu'il soit arrêté ou démarré à distance à partir du serveur UDP.

Exemple de configuration :

```
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
```

Le serveur accepte les commandes suivantes :

<START|STOP> SERVICE

Il démarre ou arrête un service spécifié (*service name*).

<START|STOP> PROCESS

Il démarre ou termine un processus spécifié (*process name*).

REFRESH AGENT

Il force l'exécution de l'agent spécifié (*agent name*), en actualisant les données.

Par exemple :

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

Il existe un *script* sur le serveur, dans `/util/udp_client.pl` qui est utilisé par Pandora FMS Server comme commande pour une alerte, pour démarrer des processus ou des services. Il a cette syntaxe :

```
./udp_client.pl
```

<command>

Par exemple, pour redémarrer un agent :

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

Pour plus d'informations, consultez le chapitre Configuration des alertes.

Définition des modules

Les modules d'exécution locale sont définis dans le fichier de configuration `pandora_agent.conf`. La syntaxe générale est la suivante :

```
module_begin
module_name
module_type generic_data
module_exec
module_end
```

Où *module name* est le nom du module et *local command* est la commande à exécuter. Il existe de nombreuses options supplémentaires pour les modules, dans cet exemple, seules les lignes communes et obligatoires ont été utilisées dans la plupart des cas.

Pour plus d'informations, regardez les vidéos suivantes:

- «How to configure a Linux Module in Pandora FMS».
- «Windows Local Module Creation | Pandora FMS».

Les sections suivantes couvrent chacun d'eux en détail.

Éléments communs de tous les modules

Les champs du module (à l'exception des données du module, de la description et des informations étendues) ne sont mis à jour que lors de la création du module, ils ne seront jamais mis à jour une fois que le module existe déjà.

module_begin

Étiquette de démarrage d'un module. Obligatoire.

module_name

```
module_name
```

Nom (*name*) du module. Ce nom doit être unique et singulier dans l'agent. Obligatoire.

module_type

```
module_type
```

Type (*type*) de données que le module renverra. Obligatoire. Les types disponibles sont les suivants :

- Numérique (`generic_data`) :
- Données numériques simples, virgule flottante ou entiers.

- Incremental (`generic_data_inc`): Données numériques égales à la différence entre la valeur actuelle et la valeur précédente divisée par le nombre de secondes écoulées. Lorsque cette différence est négative, la valeur est réinitialisée, ce qui signifie que lorsque la différence est à nouveau positive, la valeur précédente sera prise chaque fois que l'augmentation revient pour donner une valeur positive.
- Absolute incremental (`generic_data_inc_abs`) : Données numériques égales à la différence entre la valeur actuelle et la valeur précédente, sans être divisées par le nombre de secondes écoulées, pour mesurer l'incrément total au lieu de l'incrément par seconde. Lorsque cette différence est négative, la valeur est réinitialisée, ce qui signifie que lorsque la différence est à nouveau positive, la dernière valeur à partir de laquelle l'augmentation actuelle obtenue est positive sera utilisée.
- Alphanumérique (`generic_data`) :
- Il collecte les chaînes de texte alphanumériques.

- Booléens (`generic_proc`) :
- Pour les valeurs qui ne peuvent être correctes ou affirmatives (1) ou incorrectes ou négatives (0). Utile pour vérifier si un ordinateur est actif ou si un processus ou un service est en cours d'exécution. Une valeur négative (0) apporte l'état critique préaffecté, tandis que toute valeur supérieure sera considérée comme correcte.

- Alphanumérique asynchrone (`async_string`) :
- Pour les chaînes de texte de type asynchrone. La supervision asynchrone dépend d'événements ou de modifications qui peuvent ou non se produire, de sorte que ces types de modules ne sont jamais dans un état inconnu.

- Booléen asynchrone (`async_proc`) :
- Pour les valeurs booléennes de type asynchrone.

- Numérique asynchrone (`async_data`) :
- Pour les valeurs numériques de type asynchrone.

module_min

```
module_min <value>
```

Valeur (*value*) minimale que le Module doit retourner pour être accepté. Sinon, il sera ignoré par le serveur.

module_max

```
module_max <value>
```

Valeur (*value*) maximale que le Module doit retourner pour être accepté. Sinon, il sera ignoré par le serveur.

module_min_warning

```
module_min_warning <value>
```

Valeur (*value*) minimale du seuil d'avertissement `warning`.

module_max_warning

```
module_max_warning <value>
```

Valeur (*value*) maximale du seuil d'avertissement `warning`.

module_min_critical

```
module_min_critical <value>
```

Valeur (*value*) minimale du seuil critique `critical`.

module_max_critical

```
module_max_critical <value>
```

Valeur (*value*) maximale du seuil critique `critical`.

module_disabled

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

Il indique si le module est activé (`0`) ou désactivé (`1`).

module_min_ff_event

```
module_min_ff_event <value>
```

Valeur (*value*) de la protection flip flop pour les faux positifs. Le nombre de changements d'état indiqués dans cette valeur sera requis pour que le module puisse modifier visuellement son état

dans la console Web.

module_each_ff

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

Si cette option est activée (`1`), au lieu d'utiliser `module_min_ff_event` les seuils de bascule seront utilisés par état :

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

module_min_ff_event_normal

```
module_min_ff_event_normal <value>
```

Valeur (*value*) de la protection flip flop pour entrer en état `normal`.

module_min_ff_event_warning

```
module_min_ff_event_warning <value>
```

Valeur (*value*) de la protection flip flop pour entrer en état `warning`.

module_min_ff_event_critical

```
module_min_ff_event_critical <value>
```

Valeur (*value*) de la protection flip flop pour entrer en état `critical`.

module_ff_timeout

```
module_ff_timeout
```

Redémarrez le compteur de de seuil flip flop après le nombre de secondes donné. Cela implique que le nombre de changements d'état déterminés dans `module_min_ff_event` doit se produire dans un intervalle de `module_ff_timeout` quelques secondes avant que l'état ne change dans la console au niveau visuel.

module_ff_type

Version NG 734 ou supérieure.

```
module_ff_type <value>
```

Il s'agit d'une option avancée du Flip Flop pour le contrôle de l'état d'un module. Par le biais de `Keep counters` définissez des valeurs de compteur pour passer d'un état à un autre en fonction, au lieu de la valeur, de l'état du module avec la valeur reçue.

Il indique si `Keep counters` est activé (`1`) ou désactivé (`0`).

module_ff_event

```
module_ff_event X
```

Il s'agit du seuil d'exécution flip flip du module (en secondes).

module_description

```
module_description
```

Texte libre avec des informations sur le module.

module_interval

```
module_interval
```

Intervalle de module individuel. Cette valeur est un facteur multiplicateur de l'intervalle de l'agent, pas un temps libre. Par exemple, si l'agent a un intervalle de 300 secondes (5 minutes) et a besoin d'un module qui n'est traité que toutes les 15 minutes, vous devez ajouter cette ligne :

```
module_interval 3
```

De cette façon, le module sera traité toutes les 300 secondes x 3 = 900 secondes (15 minutes).

Pour que l'agent `module_interval` travaille dans Broker Agents, il doit avoir le même intervalle que l'agent d'où il provient. Sinon, il risque de ne pas fonctionner.

module_timeout

```
module_timeout <secs>
```

En secondes, durée maximale autorisée pour l'exécution du Module. Si ce délai est dépassé avant que son exécution ne soit terminée, il sera interrompu.

module_postprocess

```
module_postprocess <facteur>
```

Valeur numérique par laquelle les données renvoyées par le module seront multipliées. Utile pour les conversions d'unités.

module_save

```
module_save <var nom>
```

Il stocke la valeur renvoyée par le Module dans une variable dont le nom est indiqué dans ce paramètre (). Cette valeur peut être utilisée ultérieurement dans d'autres modules.

Exemple sous Unix/Linux :

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

Il stockera la valeur « 41121 » dans la variable « ECHO_1 ».

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

Ce deuxième module affichera le contenu de la variable « $ECHO_1 », soit « 41121 ».

Dans les agents logiciels sous Windows®, la syntaxe du Module doit être formée en enfermant la variable entre les symboles de pourcentage `%var%` au lieu de `$var`. En suivant l'exemple donné :

```
module_begin
module_name echo_2
```

```
module_type generic_data
module_exec echo %ECHO_1%
module_end
```

module_crontab

Depuis la version 3.2, vous pouvez programmer les modules pour qu'ils s'exécutent à certaines dates.

Pour ce faire, vous devez définir le module_crontab en utilisant un format similaire à celui du fichier crontab.

module_crontab ;

Étant :

- Minute 0-59.
- Heure 0-23 .
- Jour du mois 1-31
- Mois 1-12 .
- Jour de la semaine 0-6 (0 est dimanche).

Il est également possible de spécifier des intervalles en utilisant le caractère - comme séparateur.

Par exemple, pour qu'un module s'exécute tous les lundis entre 12 et 15 heures, vous pouvez utiliser les paramètres suivants :

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

Pour exécuter une commande toutes les heures, à l'heure et toutes les 10 minutes :

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

module_condition

module_condition <opération> <commande>

Il vous permet de définir les actions qui seront exécutées par l'agent en fonction de la valeur renvoyée par le module. Disponible uniquement pour les valeurs numériques. La syntaxe générale est la suivante :

- > [valeur] : Il exécute la commande lorsque la valeur du module est supérieure à la valeur donnée.
- < [valeur] : Il exécute la commande lorsque la valeur du module est mineure à la valeur donnée.
- = [valeur] : Il exécute la commande lorsque la valeur du module est égale à la valeur donnée.
- != [valeur] : Il exécute la commande lorsque la valeur du module est différente à la valeur donnée.
- =~ [expression|régulière] :
- Il exécute la commande lorsque la valeur du module correspond à l'expression régulière donnée.

- (valeur, valeur) : Il exécute la commande lorsque la valeur du module figure parmi les valeurs données.

Plusieurs conditions peuvent être spécifiées pour le même module. Dans le cas suivant,`script_1.sh` sera exécuté si la valeur renvoyée par le module est comprise entre 1 et 3, et `script_2.sh` sera exécuté si la valeur du module est supérieure à 5,5, donc dans ce cas, étant la valeur renvoyée à la ligne `module_exec` 2.5, seule la première condition `script_1.sh`>sera exécutée.

```
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
```

Exemples appliqués à des cas réels possibles :

```
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
```

- Remarque : Dans le système d'exploitation Windows®, il est conseillé de mettre cmd.exe /c avant la commande pour s'assurer qu'elle fonctionne correctement. Par exemple :

```
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 <opération> <commande>

Il permet de déterminer si le module est exécuté ou non en fonction du résultat d'une exécution donnée. Syntaxe :

- > [valeur] : Il exécute la commande lorsque la valeur du module est supérieure à la valeur donnée.
- < [valeur] : Il exécute la commande lorsque la valeur du module est mineure à la valeur donnée.
- = [valeur] : Il exécute la commande lorsque la valeur du module est égale à la valeur donnée.
- != [valeur] : Il exécute la commande lorsque la valeur du module est différente à la valeur donnée.
- =~ [expression|régulière] :
- Il exécute la commande lorsque la valeur du module correspond à l'expression régulière donnée.
- (valeur, valeur) : Il exécute la commande lorsque la valeur du module figure parmi les valeurs données.

L'exemple suivant n'exécutera que le Module (*monitoring_variable.bat*) si le résultat de l'exécution indiqué dans la condition préalable est compris entre 2 et 8. Dans ce cas, le résultat de l'exécution indiqué sur la ligne `module_precondition` est 5, une valeur comprise entre 2 et 8, il sera donc exécuté correctement *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
```

Comme pour les postconditions, il est possible d'en mettre plusieurs, et le module ne fonctionnera que si toutes sont remplies :

```
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
```

- Remarque : Dans le système d'exploitation Windows®, il est conseillé de mettre cmd.exe /c avant la commande pour s'assurer qu'elle fonctionne correctement. Par exemple :

```
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

```
module_unit <chaîne de caractères>
```

Unités exprimées dans une chaîne à afficher (*string*) à côté de la valeur obtenue par le module. Exemple :

```
module_unit %
```

module_group

```
module_group <value>
```

Il permet d'indiquer le groupe de modules (*value*) auquel le module sera affecté. Exemple :

```
module_group Networking
```

module_custom_id

```
module_custom_id <value>
```

Cette stratégie est un identificateur personnalisé pour le module. Exemple :

```
module_custom_id host101
```

module_str_warning

```
module_str_warning <value>
```

Il permet d'indiquer une expression régulière pour définir le seuil d'avertissement `warning` dans Modules alphanumériques (*string*). Exemple :

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

module_str_critical

```
module_str_critical <value>
```

Il permet d'indiquer une expression régulière pour définir le seuil critique `warning` dans des Modules alphanumériques (*string*). Exemple :

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

module_warning_instructions

```
module_warning_instructions <value>
```

Au niveau des informations, il indique les instructions qui seront affichées dans l'événement généré par le module lorsqu'il passe à l'état d'avertissement `warning`.

Exemple pour afficher un message indiquant l'augmentation de la priorité dans un incident :

```
module_warning_instructions Raise advocacy priority
```

module_critical_instructions

```
module_critical_instructions <value>
```

Au niveau des informations, il indique les instructions qui seront affichées dans l'événement généré par le module lorsqu'il passe à l'état d'avertissement `critical`.

Exemple de message pour alerter le département systèmes :

```
module_critical_instructions Call the systems department
```

module_unknown_instructions

```
module_unknown_instructions <value>
```

Au niveau des informations, il indique les instructions qui seront affichées dans l'événement généré par le module lorsqu'il passe à l'état inconnu unknown.

Exemple de message pour ouvrir une incidence :

```
module_unknown_instructions Open incident
```

module_tags

```
module_tags <value>
```

Étiquettes ou *tags* que vous souhaitez affecter au module, séparées par des virgules.

Exemple :

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

module_warning_inverse

```
module_warning_inverse <value>
```

Il permet d'activer (1) la plage inverse du seuil d'avertissement `warning`.

Exemple :

```
module_warning_inverse 1
```

module_critical_inverse

```
module_critical_inverse <value>
```

Il permet d'activer (1) l'intervalle inverse pour le seuil critique `critical`.

Exemple :

```
module_critical_inverse 1
```

module_native_encoding

Uniquement sur Win32.

```
module_native_encoding <value>
```

Ce *token* de configuration affecte uniquement les modules exécutés à l'aide d'une stratégie de commande, c'est-à-dire qu'il existe un `module_exec` présent.

MS Windows® gère trois codages pour ses processus: le codage en ligne de commande (OEM), l'encodage système (ANSI) et UTF-16. Ces codages coïncident dans les caractères de base, mais

diffèrent dans ceux moins courants, tels que les accents. Avec ce *token*, l'agent Pandora FMS convertit la sortie de la commande en encodage spécifié dans le *encoding* du fichier de configuration.

`module_native_encoding` a quatre valeurs valides :

- `module_native_encoding OEM` : Pour l'encodage en ligne de commande.
- `module_native_encoding OEM` : Pour le codage système.
- `module_native_encoding UTFLE` : Pour UTF-16 little-endian.
- `module_native_encoding UTFBE` : Pour UTF-16 little-endian.

Si `module_native_encoding` n'apparaît pas, aucun recodage ne sera effectué.

module_quiet

```
module_quiet <value>
```

S'il est activé (1) le module sera en mode silencieux : il ne générera pas d'événements ni ne déclenchera d'alertes, ni ne stockera de données historiques.

Exemple :

```
module_quiet 1
```

module_ff_interval

```
module_ff_interval <value>
```

Il permet d'indiquer un seuil Flip Flop dans le Module, par exemple :

```
module_ff_interval 2
```

module_macro

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

Applicable uniquement sur les composants locaux à partir de la console. Il n'a aucune utilité dans le fichier de configuration.

module_alert_template

```
module_alert_template
```

Cette macro affecte au Module créé le modèle d'alerte correspondant au nom saisi en paramètre (voir Modèles d'alerte).

Exemple :

```
<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>
```

intensive_interval

Intervalle de supervision intensive. Les modules utilisant `module_intensive_monitorig` peuvent signaler si leur état est incorrect dans cette plage.

module_intensive_condition

Condition pour la supervision intensive. Lorsqu'un module de supervision intensive atteint la valeur configurée dans ce paramètre, il doit notifier dans le intervalle intensif définie.

module_end

Étiquette de finale d’un module. Il est obligatoire.

Directives spécifiques pour l'obtention d'informations

Les directives spécifiques qui peuvent être spécifiées dans chaque module pour obtenir des informations en tant que telles sont décrites ci-dessous. Dans chaque module *un seul de ces types peut être utilisé* dans chaque module.

module_exec

```
module_exec
```

Ligne générale d'exécution des commandes. L'exécution souhaitée doit être précisée pour obtenir les informations sur une seule ligne.

Sous GNU/Linux, la commande sera exécutée via le shell par défaut. L'interpréteur par défaut est déterminé par le lien symbolique de `/bin/sh`. Normalement, le lien pointe vers bash, mais sur des systèmes comme Ubuntu, ce n'est pas le cas (dans ce cas, il pointe vers dash). Pour cette raison, il peut arriver qu'une commande soit essayée dans le terminal puis échoue lorsque l'agent logiciel l'exécute. Une solution qui fonctionnera dans la plupart des cas sera de forcer l'exécution de la commande bash de la manière suivante :

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

Si l'exécution de la commande renvoie un code d'erreur (return code) différent de 0, la commande sera interprétée comme donnant une erreur et les données obtenues seront rejetées.

Pour un agent sous Windows®, il existe d'autres directives pour obtenir des données, celles-ci sont décrites ci-dessous.

module_service

```
module_service <service>
```

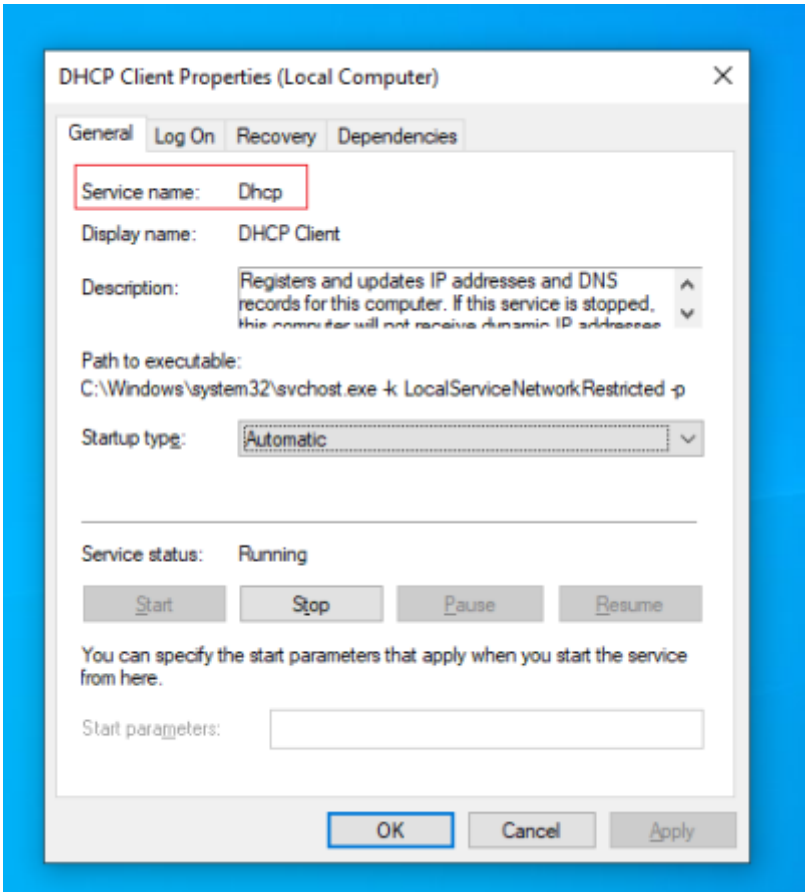
Il vérifie si un certain service est en cours d'exécution sur la machine.

Sur MS Windows

Si le nom du service contient des espaces, vous devez utiliser des guillemets “ ”.

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

Le service est identifié par le nom abrégé du service (`Service name`), tel qu'il apparaît dans le gestionnaire de services Windows.



Mode asynchrone

Pandora FMS, en général, exécute une batterie de tests (chacun étant défini dans un Module) toutes les X secondes (par défaut 300 secondes = 5 minutes) de sorte que si un service tombe en panne juste après une exécution de Pandora FMS, il faudra au moins 300 secondes supplémentaires pour savoir qu'il est en panne. Les modules asynchrones permettent à Pandora FMS de notifier *instantanément* l'échec de ce service. Nous appelons ce mode de fonctionnement *asynchrone*. Pour ce faire, ajoutez simplement la directive.

```
module_async yes
```

Cette fonctionnalité n'est pas prise en charge sur les agents Broker.

Dans les versions Windows Home Edition®, cette fonctionnalité asynchrone n'est pas prise en charge et, uniquement dans ces versions, l'Agent Pandora FMS effectue une requête périodique pour savoir si le service est en cours d'exécution ou non. Cela peut consommer beaucoup de ressources, il est donc recommandé d'utiliser la version synchrone si vous surveillez un grand nombre de services.

Watchdog de services

Il existe un mode de vigilance ou *watchdog* pour les services, afin que l'agent puisse les redémarrer s'ils s'arrêtent. Dans ce cas, le service redémarré ne nécessite aucun paramètre car Windows® sait déjà le faire. Dans ce cas la configuration est plus simple et cela pourrait être un exemple :

```
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
```

Sur Unix

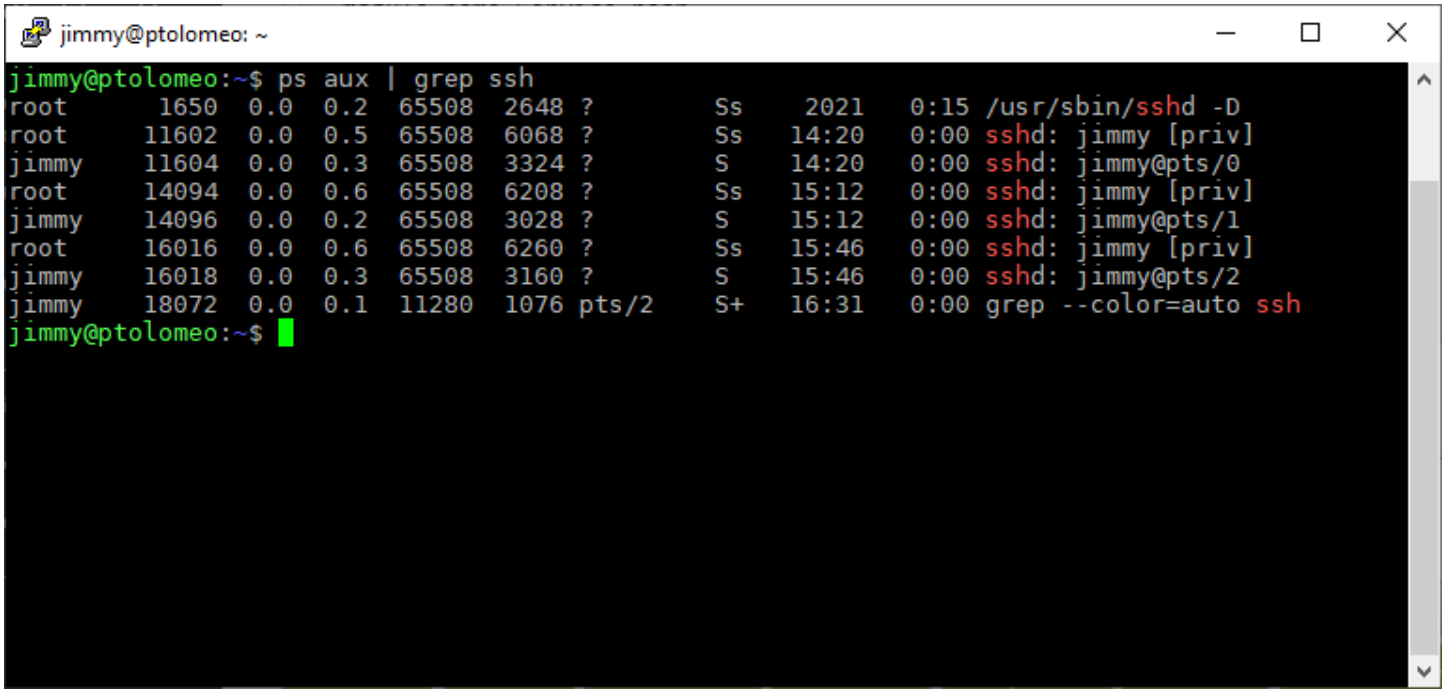
Sous Unix, cela fonctionne de la même manière que dans MS Windows®, sauf que pour le processus et le service Unix, c'est le même concept, par exemple, pour voir si le processus bash est actif dans le système, il suffira d'exécuter :

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

Le mode watchdog et la détection asynchrone ne sont pas
possibles dans l'agent Unix.

Pour `module_service`, vous devez saisir le chemin complet tel que le service apparaît avec la commande `ps aux` . Par exemple, pour trouver le service SSH :

```
ps aux | grep ssh
```



Il doit être configuré comme suit :

```
module_begin
module_name MY_SSHD
module_type generic_proc
module_service /usr/sbin/sshd -D
module_description Is sshd running?
module_end
```

module_proc

```
module_proc <processus>
```

Vérifiez si un certain nom de processus est en cours d'exécution sur cette machine.

Sur MS Windows®

Les guillemets pour le nom du processus sont inutiles.
Notez que le nom du processus doit avoir l'extension `.exe`.
Le module renverra le nombre de processus en cours
d'exécution avec ce nom.

Ce serait un exemple de la supervision du processus `cmd.exe` >

```
module_begin
module_name CMDProcess
module_type generic_proc
module_proc cmd.exe
```

```
module_description Process Command line
module_end
```

Mode asynchrone

Comme pour les services, les processus de supervision peuvent être critiques dans certains cas. L'agent logiciel pour Windows® prend désormais en charge les vérifications asynchrones du paramètre `module_proc`. Dans ce cas, l'Agent avertit immédiatement lorsque le processus change d'état, sans attendre l'expiration de l'intervalle d'exécution de l'Agent. De cette façon, vous pouvez être au courant des temps d'arrêt des processus critiques presque dès qu'ils se produisent. Exemple de supervision de processus asynchrone :

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

La différence réside dans le *token* de configuration `module_async yes`.

Cette fonctionnalité n'est pas prise en charge sur les agents Broker.

Watchdog de processus

Un *Watchdog* est un système qui vous permet d'agir immédiatement lorsqu'un processus plante, généralement en levant le processus qui a planté. L'agent logiciel Pandora FMS pour Windows® peut agir en tant que *watchdog* lorsqu'un processus se bloque.

Étant donné que l'exécution d'un processus peut nécessiter certains paramètres, il existe des options de configuration supplémentaires pour ces types de modules.

Il est important de noter que le mode *Watchdog* ne
fonctionne que lorsque le type de module est *asynchrone*.

Exemple de configuration d'un `module_proc` avec watchdog :

```
module_begin
module_name Notepad
module_type generic_proc
module_proc notepad.exe
module_description Notepad
module_async yes
module_watchdog yes
module_start_command c:\windows\notepad.exe
```

```
module_startdelay 3000
module_retrydelay 2000
module_retries 5
module_end
```

Voici la définition des paramètres supplémentaires pour `module_proc` avec *Watchdog* :

module_retries

Nombre de tentatives consécutives que le Module essaiera de lancer le processus avant de désactiver le *Watchdog* . Si la limite est atteinte, le mécanisme *Watchdog* de ce Module sera désactivé et ne tentera plus de relancer le processus tant que l'Agent n'aura pas redémarré. Valeur par défaut : Illimitée.

module_startdelay

Nombre de millisecondes que le module attendra avant de lancer le processus pour la première fois.

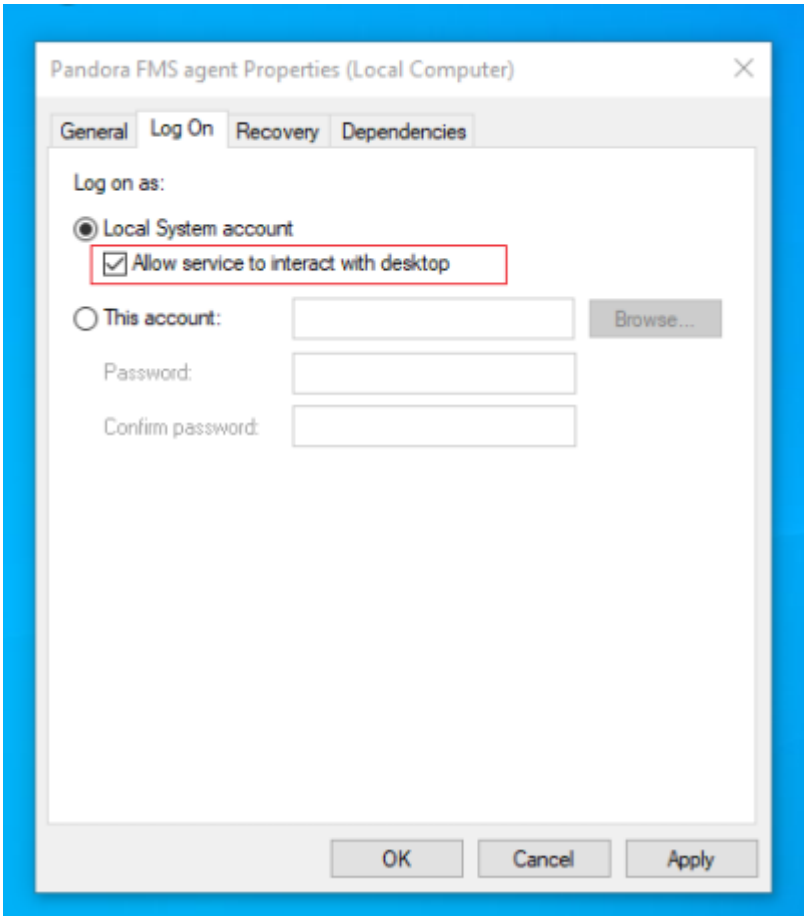
module_retrydelay

Nombre de millisecondes que le module attendra avant de lancer le processus pour la première fois.

module_user_session

Contrôle dans quelle session vous voulez que le processus se lance. S'il est défini sur no, le processus démarrera dans la session de services et, par conséquent, restera en arrière-plan (option par défaut). Sinon, s'il est défini sur `yes`, le processus sera lancé dans la session de l'utilisateur et sera visible depuis le bureau du PC.

Pour les versions antérieures à Windows Vista®, le *jeton* `module_user_session` peut être configuré de manière générale en activant la case « Accès interactif avec le bureau » dans les propriétés du service Pandora FMS :



Pandora FMS, en tant que service, s'exécute sous le compte SYSTEM et que le processus exécuté le fera sous cet utilisateur et avec cet environnement, de sorte que si vous devez exécuter un processus spécifique qui doit être utilisé avec un utilisateur spécifique, vous devez encapsuler dans un *script* (`.bat` ou similaire) les processus précédents pour initialiser l'environnement, les variables d'environnement, etc.), et exécuter ce *script* en tant qu'action *Watchdog*.

Sur Unix

Sous Unix, cela fonctionne exactement de la même manière que module_service . Il ne prend pas non plus en charge le mode asynchrone ou *watchdog*.

module_cpuproc

Unix uniquement

```
module_cpuproc
```

Il renvoie l'utilisation spécifique du processeur d'un processus. Exemple.

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

module_memproc

```
module_memproc
```

Unix uniquement. Il renvoie la consommation de mémoire spécifique d'un processus.

```
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>
```

Il vérifie l'espace libre sur l'unité.

Sous Windows®

Vous devez mettre `:` après la lettre du lecteur (`<disk_letter:>`).

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

Sous Unix®

Le volume à vérifier, tel que `/var` .

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

```
module_end
```

module_freepercentdisk

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

Ce module renvoie le pourcentage de disque libre dans un lecteur logique.

Sous Windows

Vous devez mettre `:` après la lettre du lecteur (`<disk_letter:>`).

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

Sur Unix

Le volume à vérifier, tel que `/var` .

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

module_occupiedpercentdisk

```
module_occupiedpercentdisk <vol>
```

Pour Unix uniquement. Ce module renvoie le pourcentage de disque occupé, par exemple :

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

module_cpuusage

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

Il renvoie l'utilisation du processeur dans un certain numéro d'UCT. S'il n'y a qu'un seul

processeur, ne définissez aucune valeur ou utilisez la valeur `all`. Pour Windows® et Unix.

Il est également possible d'obtenir l'utilisation moyenne de tous les processeurs d'un système multiprocesseur :

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

Pour vérifier l'utilisation de l'UCT #1 :

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

module_freememory

Il fonctionne à la fois sur Unix et Windows®. Il renvoie la mémoire libre sur l'ensemble du système.

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

module_freepercentmemory

Il fonctionne à la fois sur Unix et Windows®. Ce module renvoie le pourcentage de mémoire libre sur un système :

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

module_tcpcheck

MS Windows® uniquement. Ce module initie la connexion avec l'adresse IP et le port spécifiés. Il renvoie 1 en cas de succès et 0 sinon. Un délai d'expiration doit être spécifié avec `module_timeout`. Exemple :

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

module_regexp

Pour MS Windows® uniquement. Ce module surveille un fichier journal *log* à la recherche de correspondances à l'aide d'expressions régulières, en supprimant les lignes existantes lorsque la supervision démarre. Les données renvoyées par le Module dépendent du type de Module :

- generic_data_string, async_string: Il renvoie toutes les lignes qui correspondent à l'expression régulière.
- generic_data : Il renvoie toutes les lignes qui correspondent à l'expression régulière.
- generic_proc : Il renvoie 1 s'il y a correspondance, 0 sinon.
- module_noseekeof : Par défaut inactif 0. Avec ce *token* de configuration 1 actif, à chaque exécution, quelles que soient les modifications dans le fichier *log*, le module relance sa vérification sans rechercher la fin du fichier (*flag*EOF). De cette façon, il affichera toujours dans le XML toutes les lignes qui correspondent au modèle de recherche. Exemple :

```
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

Windows® uniquement. Les modules WMI vous permettent d'exécuter n'importe quelle *query* WMI localement sans utiliser d'outil externe. Il est configuré au moyen de deux paramètres :

- module_wmiquery : WQL *query* employée. Plusieurs lignes peuvent ainsi être obtenues, qui seront insérées comme plusieurs données.
- module_wmicolumn : Nom de la colonne à utiliser comme source de données.

Par exemple, pour une liste des services installés :

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

Pour obtenir la charge de l'UCT actuelle :

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

module_perfcounter

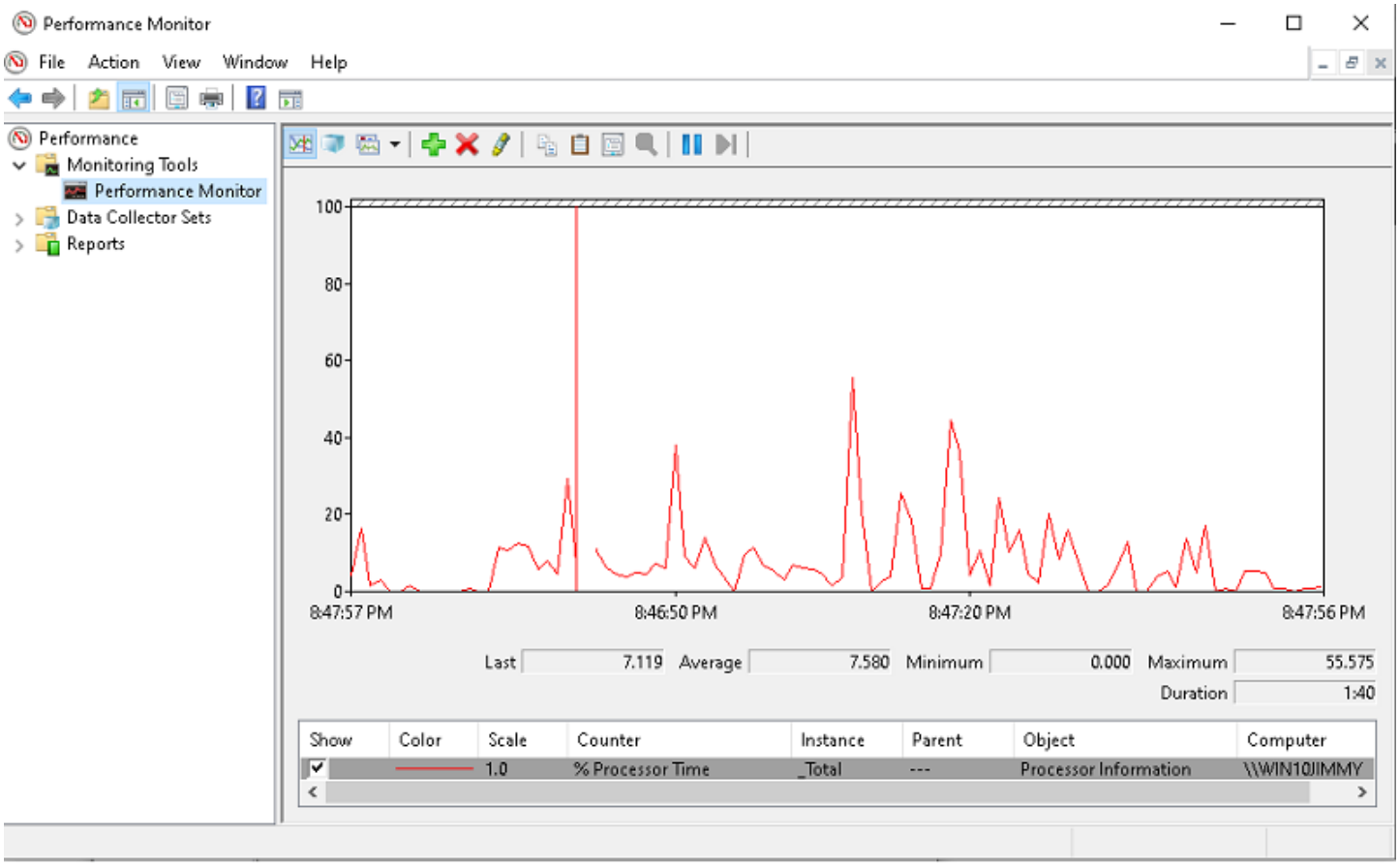
Uniquement pour MS Windows®.

Obtenez des données de compteur de performances via l'interface PDH. La bibliothèque `pdh.dll` doit être installée sur le système. PDH.DLL est une bibliothèque Windows, si elle n'est pas disponible, vous devez installer l'outil d'analyse des performances de Windows®, qui est généralement fourni par défaut.

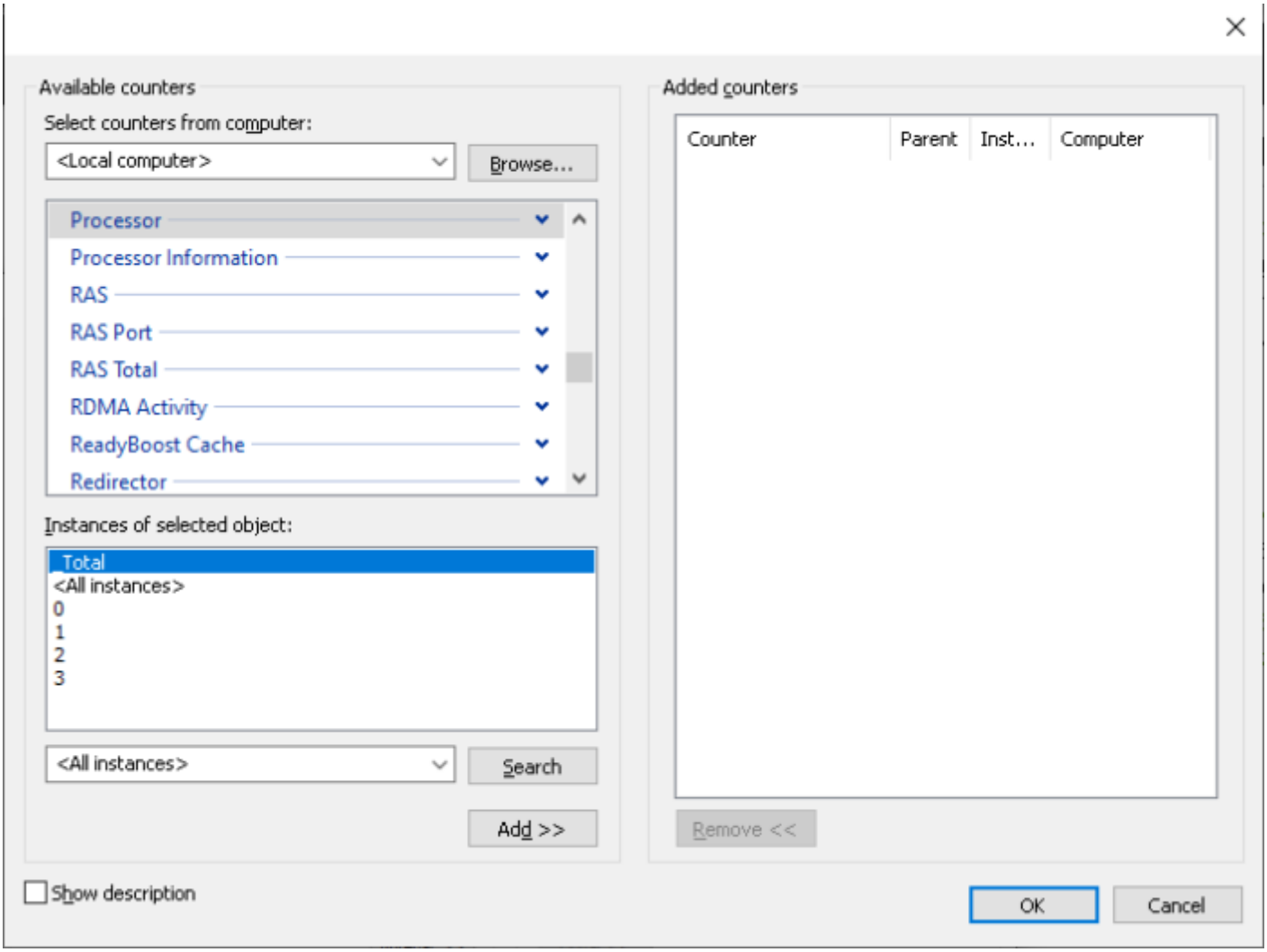
```
module_begin
module_name perfcounter
module_type generic_data
module_perfcounter \Memory\Pages/sec
module_end
```

Le moniteur de performance Windows® est un outil très puissant avec des centaines de paramètres qui peuvent être utilisés pour le monitorage. De plus, chaque constructeur intègre ses propres compteurs.

Vous pouvez observer les compteurs de performance à l'aide de l'outil Performance :



De nouveaux compteurs de performances peuvent être ajoutés à l'aide de l'outil système. Sa configuration a une structure hiérarchique avec des éléments et des sous-éléments. Dans ce cas *Processeur* , *% de temps de processeur* et *_Total* :



La configuration du module pour ce contrôle particulier est la suivante :

```
module_begin
module_name Processor_Time
module_type generic_data_inc
module_perfcounter \Processeur(_Total)\% Temps processeur
module_end
```

Par défaut la valeur brute du compteur est affichée, pour obtenir la valeur *cuite* vous pouvez ajouter le paramètre module_cooked 1 :

```
module_begin
module_name Disk_E/S_Seg
module_type generic_data
module_cooked 1
module_perfcounter \PhysicalDisk(_Total)\I/O divisé par sec.
module_end
```

Une grande partie des données qu'il renvoie sont des compteurs, vous devrez donc utiliser generic_data_inc comme type de données. Il peut également renvoyer des valeurs à des échelles de données très élevées (plusieurs millions), vous pouvez donc réduire ces valeurs à l'aide du post-traitement du module, avec des valeurs telles que 0,000001 ou similaire.

module_inventory

Actuellement, cette fonctionnalité a été remplacée par l'
inventaire des plugins Agent sur les systèmes Windows®
et Linux/Unix®.

module_logevent

Pour MS Windows® uniquement : Il permet d'obtenir des informations du *log* des événements Windows® en fonction des modèles indiqués, permettant un filtrage basé sur la source et le type d'événement.

Le format général de ce module est le suivant :

```
module_begin
module_name MyEvent
module_type async_string
module_logevent
module_source
module_eventtype
module_eventcode
module_application
module_pattern
module_description
module_end
```

Pour éviter d'afficher des informations répétées, seuls les événements survenus depuis la dernière exécution de l'Agent sont pris en compte.

`module_logevent` accepte les paramètres suivants, qui nécessitent tous une entrée correcte sensible à la casse:

- module_source : Source d'événement (System, Application, Security). Champ requis.
- module_eventtype : Type d'événement (error, information...). Champ facultatif.
- module_pattern : Modèle à rechercher (sous-chaîne). Champ facultatif.
- module_eventcode : ID numérique de l'événement. Champ facultatif.
- module_application : Application à l'origine de l'événement enregistré dans le *log*. Distinguez-le bien du `module_source` qui indique le nom de la source ou du fichier *log* à partir duquel les événements sont recherchés.

Par exemple, pour afficher tous les événements système de type erreur :

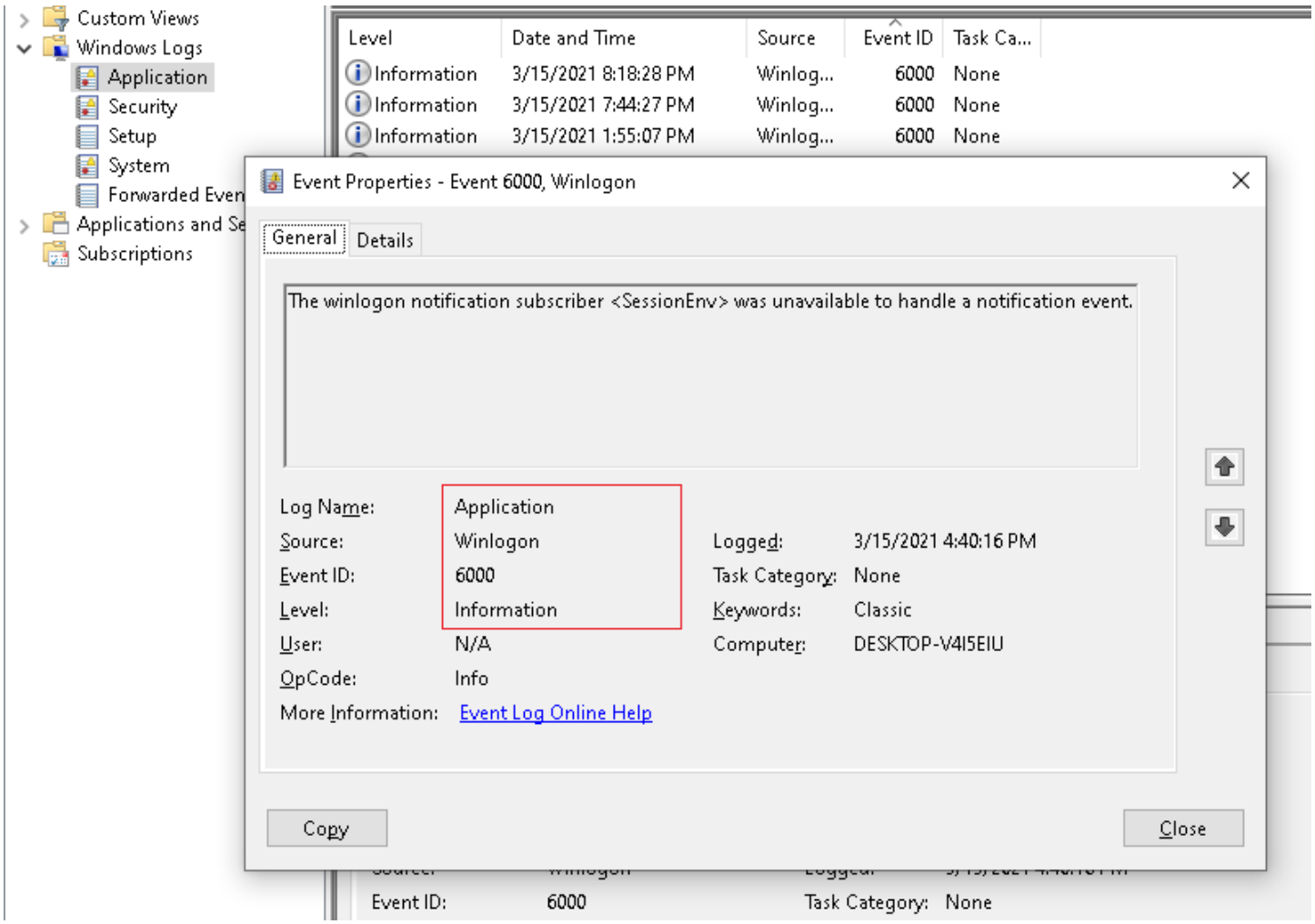
```
module_begin
module_name log_events
module_type generic_data_string
module_description System errors
```

```
module_logevent
module_source System
module_eventtype error
module_end
```

Pour afficher tous les événements contenant le mot `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
```

Exemple pour filtrer l'événement suivant :




```
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

Version NG 715 ou supérieure, uniquement pour MS Windows®.

Type de module permettant d'obtenir des informations à partir des canaux de *logs* de Windows®. Bien que `module_logevent` n'ait accès qu'aux *logs* du registre Windows®, ce type de module vous permet d'extraire des données d'autres fichiers *logs* configurés en tant que canaux. De cette manière, il est possible d'obtenir les *logs* inclus dans les journaux d'application et de service.

Le format général de ce module est le suivant :

```
module_begin
module_name MyEvent
module_type async_string
module_logchannel
module_source
module_eventtype
module_eventcode
module_application
module_pattern
module_description
module_end
```

Pour éviter d'afficher des informations répétées, seuls les événements survenus depuis la dernière exécution de l'Agent sont pris en compte.

`module_logchannel` aaccepte les paramètres suivants, qui nécessitent tous une entrée correcte de majuscules y minuscules (case-sensitive) :

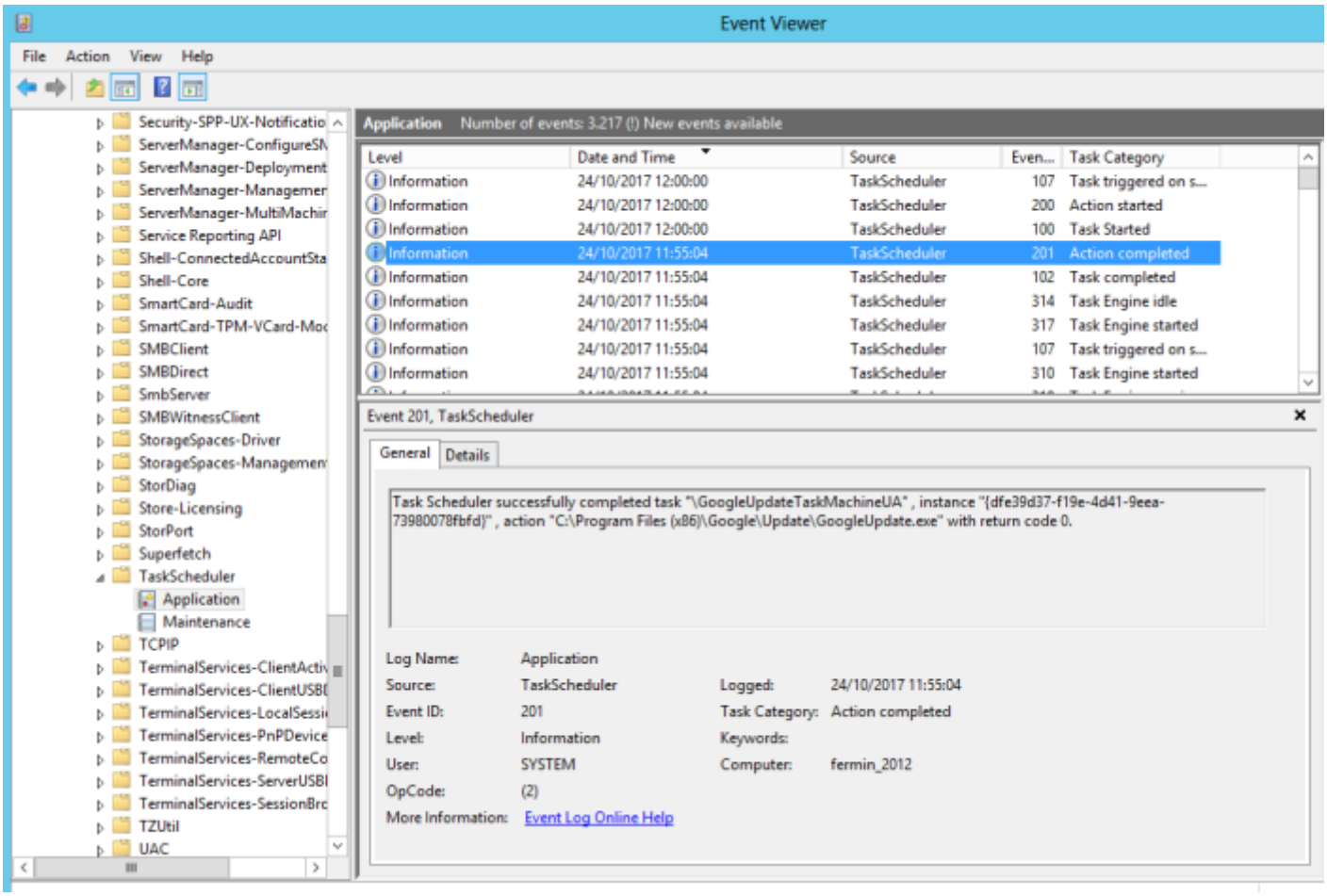
- module_source : Canal d'événement. Avec la commande `wevtutil.exe enum-logs`, une liste de tous les canaux de *logs* locaux de la machine est obtenue. Champ requis.
- module_eventtype : Type d'événement (`critical` , `error` , `warning` , `info` ou `verbose`). Champ facultatif.
- module_pattern : Modèle à rechercher (sous-chaîne). Champ facultatif.
- module_eventcode : Identifiant numérique de l'événement. Champ facultatif.
- module_application : Application de source d'événement. À distinguer de `module_source` qui indique le nom de la source ou du fichier *journal* à partir duquel les événements sont recherchés.

Par exemple, le module suivant permet d'afficher tous les événements du canal *Microsoft-Windows-TaskScheduler/Operational*, de type *information*, avec le code *201* et dont le texte *code 0*

apparaît dans le texte du journal :

```
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
code module_pattern 0
module_end
```

Avec cette configuration du Module, l'Agent Pandora FMS peut collecter le *log* suivant :



Pour obtenir le nom du canal d'événement, il faudra faire un clic droit dessus, sélectionner Propriétés et copier le paramètre Full name, nécessaire pour `module_source`.

module_plugin

Pour exécuter les *plugins* de l'Agent . C'est un cas particulier puisqu'il ne nécessite aucune autre

étiquette de type `module_begin` ou `module_end`, ni d'indiquer le type de Module.

Syntaxe avec leurs paramètres respectifs :

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

Cependant, il est également possible de l'utiliser entre les étiquettes des Modules habituelles pour ajouter des options supplémentaires telles que des conditions ou un intervalle :

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

Les paramètres à utiliser seront différents pour chaque *plugin*, il faudra donc se référer à sa documentation particulière. Pour décrire le fonctionnement d'un des *plugins* livrés par défaut avec l'Agent, le *plugin* `grep_log` sert d'exemple pour rechercher des correspondances dans un fichier :

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

Dans cet exemple, le nom du plugin s'appelle « grep_log » et il cherchera dans le fichier « /var/log/syslog » l'expression régulière « ssh » et l'enregistrera dans un module appelé « Syslog ».

Exemple d'appel de *plugin* dans un Agent Windows :

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

module_ping

Pour Windows® uniquement .

```
module_ping
```

Ce module ping le ou les *hôtes* spécifiés et renvoie 1 s'il est en ligne.

Réglages :

- module_ping_count x : Nombre de paquets ECHO_REQUEST à envoyer (1 par défaut).
- module_ping_timeout x : Délai d'attente en millisecondes pour chaque réponse (1 000 par défaut).
- module_advanced_options : Options avancées pour `ping.exe`.

Exemple :

```
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

Pour MS Windows® uniquement.

```
module_snmpget
```

Ce module exécute une requête SNMP `get` et renvoie la valeur demandée. Les paramètres de configuration doivent être spécifiés sur les lignes suivantes comme ceci :

- module_snmpversion [1_2c_3] : Version SNMP (1 par défaut).
- module_snmp_community : Communauté SNMP (*public* par défaut).
- module_snmp_agent : Agent SNMP cible.
- module_snmp_oid : OID cible.
- module_advanced_options : Options avancées pour `snmpget.exe`.

Exemple :

```
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

Pour MS Windows® uniquement.

```
module_wait_timeout X
```

Timeout utilisé lors de la vérification de la sortie des modules module_exec et module_plugin. Valeur par défaut 500 millisecondes. Modifiez à 5 si l'exécution d'un module qui génère beaucoup de sortie est lente. Il est recommandé de ne pas l'utiliser dans d'autres cas.

module_advanced_options

Seulement pour MS Windows®.

```
module_advanced_options <parameter>
```

Pour module_ping et module_snmpgetlet, utilisez des paramètres supplémentaires. Par exemple, pour ping, il peut spécifier une taille de paquet de 512 octets :

```
module_advanced_options -l 512
```

Configuration automatique des agents

Introduction

Version NG 725 ou supérieure.

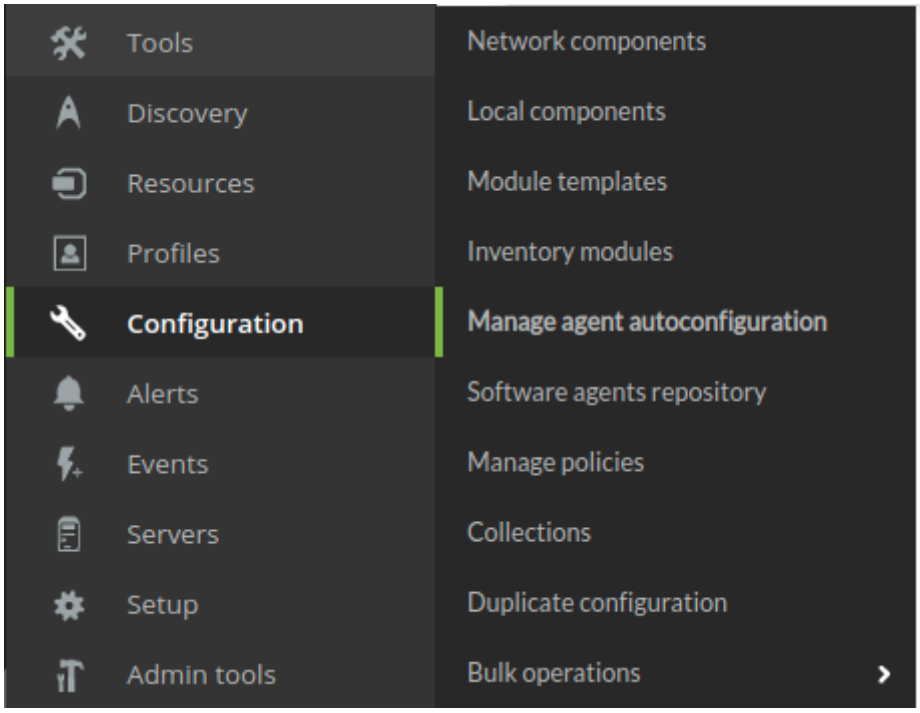
Dans le processus de configuration automatique de l'agent, vous pouvez établir une série de règles afin qu'elles soient configurées automatiquement et cela fonctionne comme suit :

1. Préparez les configurations automatiques dans votre Console Pandora FMS ou Métaconsole Pandora FMS.
2. Installez les agents qui rapportent à votre Pandora FMS (si vous avez une Métaconsole avec le système de provisionnement automatique configuré, définissez la Métaconsole elle-même comme serveur).
3. Le serveur Pandora FMS recevra un XML (`.data`) avec les données de l'agent pour la première fois.
4. Les règles seront évaluées pour déterminer la configuration automatique à appliquer.
5. L'agent récupérera la nouvelle configuration et signalera au cycle suivant la configuration mise à jour.

Création/modification de la configuration automatique

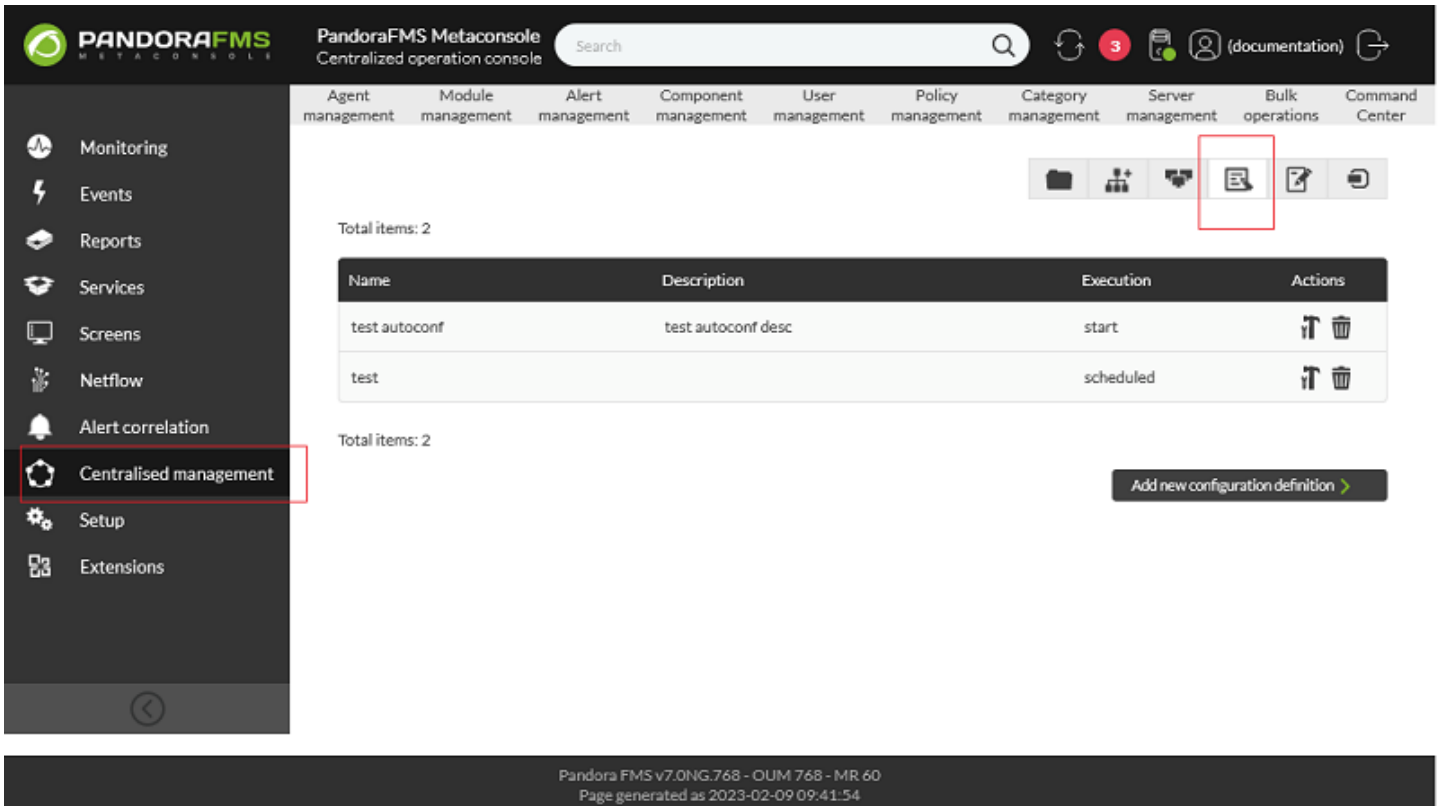
Console

Accédez à la gestion des configurations automatiques via Configuration → Manage agent autoconfiguration:

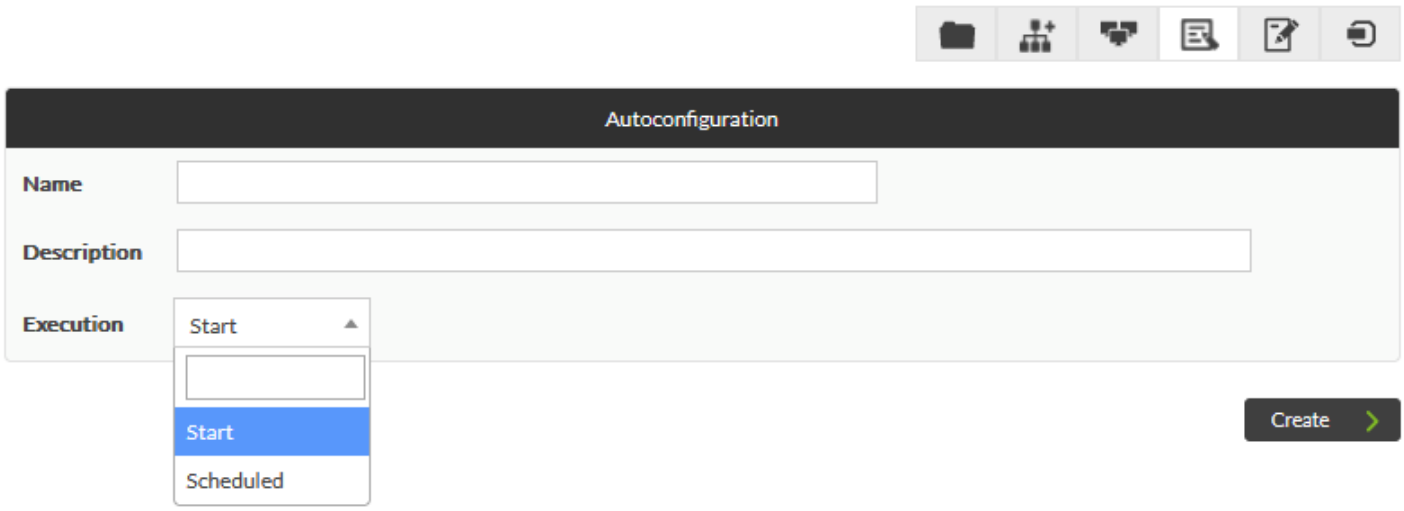


Métaconsole

Accédez à Centralised management → Agent management → icône de configuration automatique des agents:

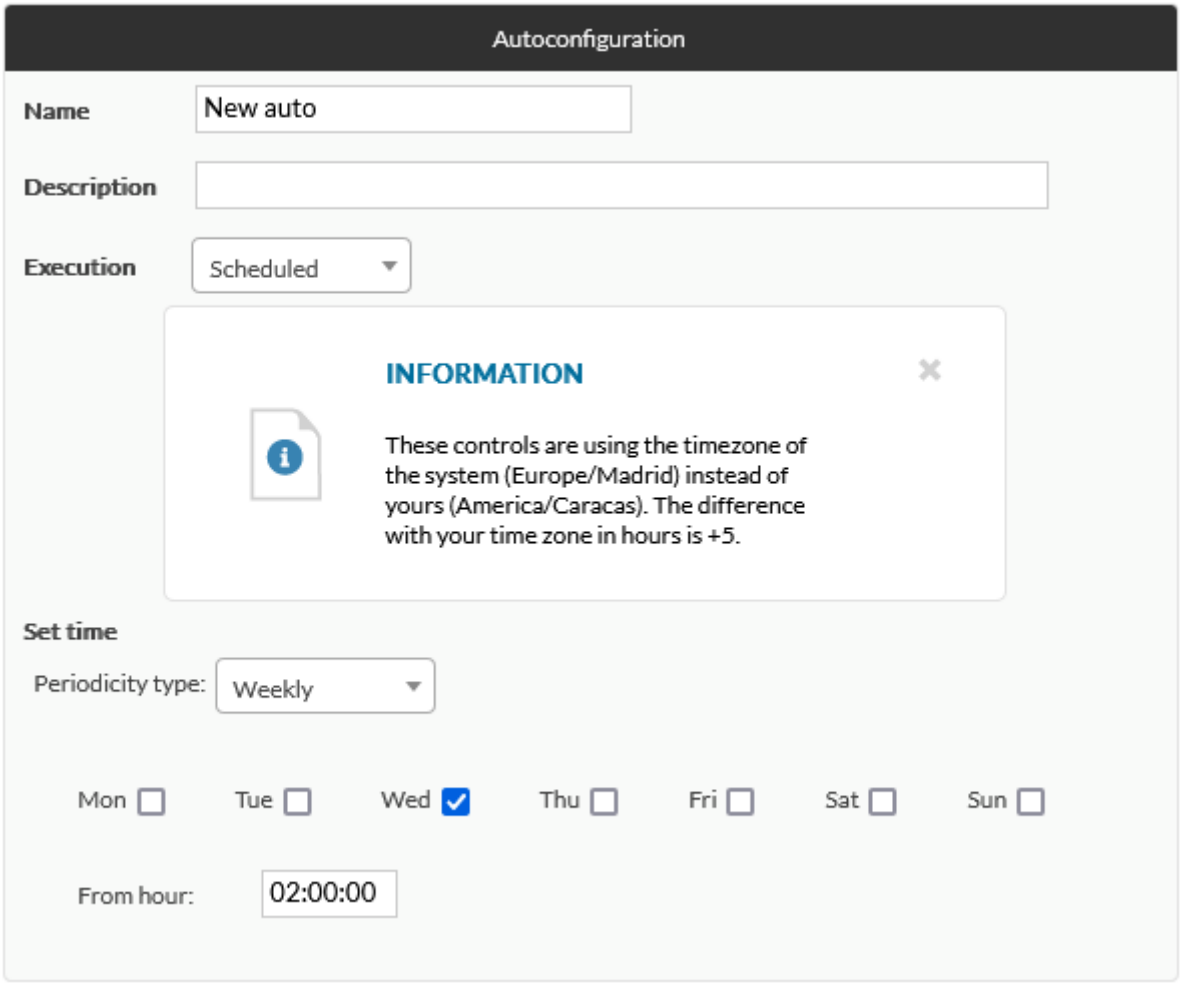


Une fois que vous accédez à la page d'administration, vous pouvez créer de nouvelles configurations automatiques en cliquant sur « Créer une configuration automatique » (Add new configuration definition). Vous devrez choisir un nom et une description pour votre configuration automatique.



Version 764 ou supérieure:

Elle permet d'exécuter périodiquement la tâche d'autoconfiguration en fonction du fuseau horaire du serveur PFMS.

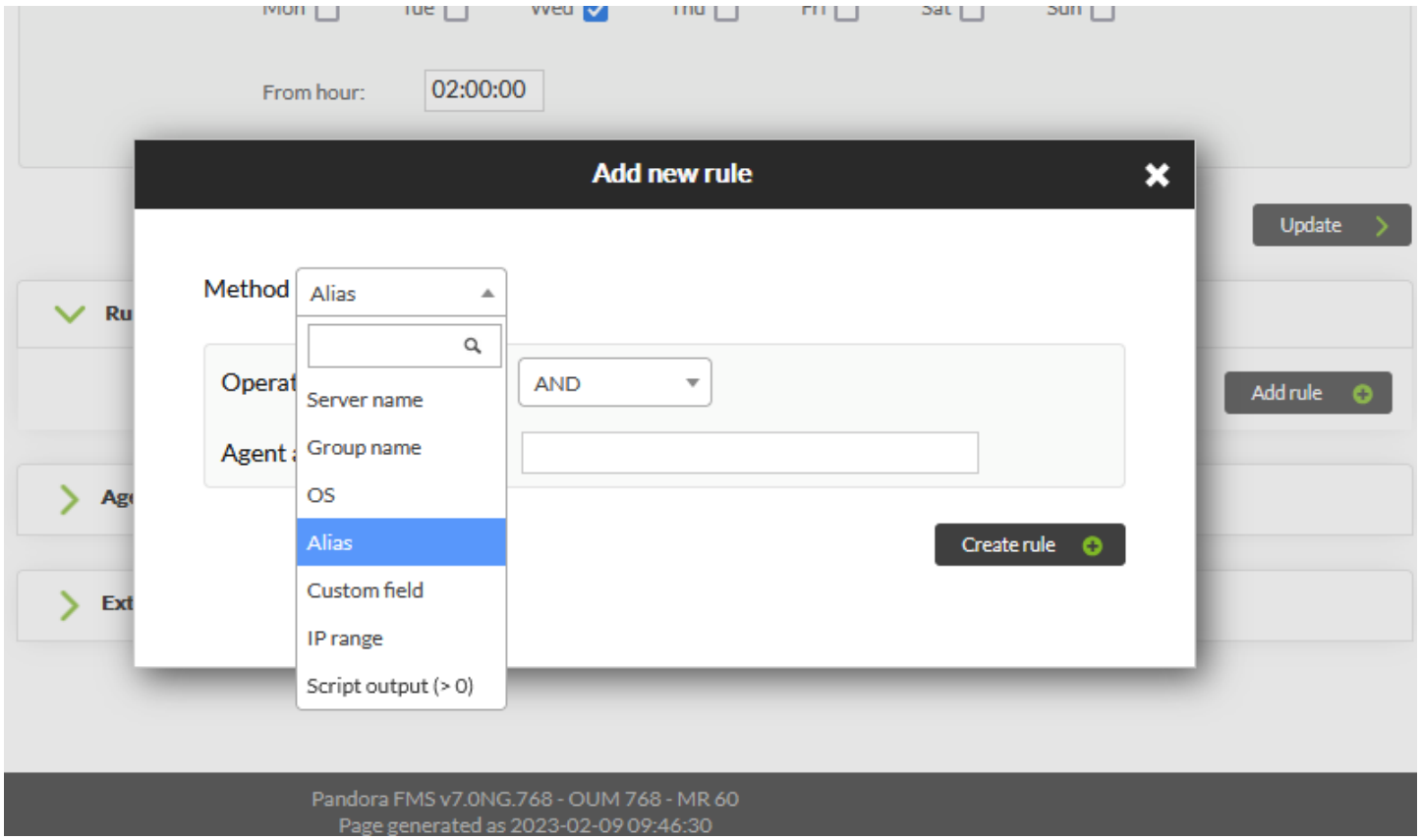


Une fois la nouvelle configuration automatique créée, vous pouvez afficher les formulaires de configuration en cliquant sur la section dont vous avez besoin:

Règles

Pour définir les Agents sur lesquels la configuration automatique sera appliquée, vous pouvez d'abord ajouter des règles pour les identifier.

Déployez la section des règles dans votre configuration automatique et sélectionnez « Ajouter une nouvelle règle » (Add new rule). Vous pouvez choisir une série d'options dans le sélecteur de règles pour identifier les agents qui vont être configurés.



Server name

Correspondance sur le nom du serveur.

Group name

Correspondance sur le nom du groupe.

OS

Correspondance du nom du système d'exploitation à l'aide d'expressions régulières.

Custom field

Correspondance par clé/valeur basée sur un champ personnalisé signalé par l'agent. Indiquez le nom du champ personnalisé et la valeur qu'il doit avoir.

IP range

Correspondance par plage d'adresses IP (réseau), utilisez la notation IP/masque, par exemple :

```
192.168.1.0/24
```

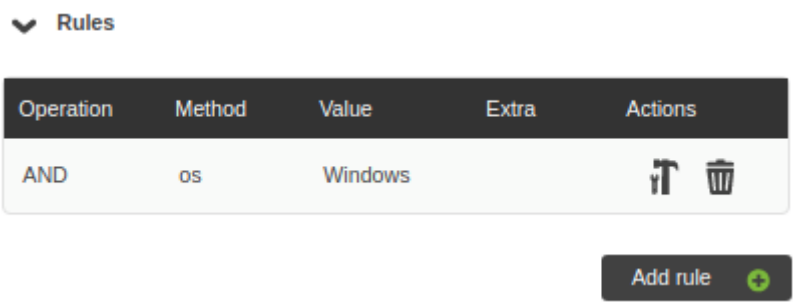
Script output (> 0)

Destiné à exécuter un *script* dont le résultat d'exécution est évalué comme valide lorsque la sortie standard est supérieure à 0.

Appelle au *script* de règles

Il prend en charge les macros suivantes dans le champ « arguments » (vous pouvez choisir entre les opérateurs AND et OR pour modifier la logique des règles) :

- _agent_ : Il sera remplacé par le nom de l'Agent.
- _agentalias_ : Il sera remplacé par l'alias de l'Agent.
- _address_ : Il sera remplacée par l'adresse IP principale communiquée par l'Agent.
- _agentgroup_ : Il sera remplacé par le nom du groupe rapporté par l'Agent.
- _agentos_ : Il sera remplacé par le système d'exploitation de l'Agent.



Si vous n'ajoutez aucune règle, les paramètres automatiques ne seront pas appliqués. Si vous avez besoin d'une configuration unique pour tous les agents, vous pouvez utiliser l'expression régulière suivante pour faire correspondre n'importe quel *alias* : `.*`

Configurations

À partir de cette section, vous pouvez configurer :

Groupe de l'agent

Vous pouvez le garder sans aucune modification ou le forcer à être spécifique.

Groupes secondaires

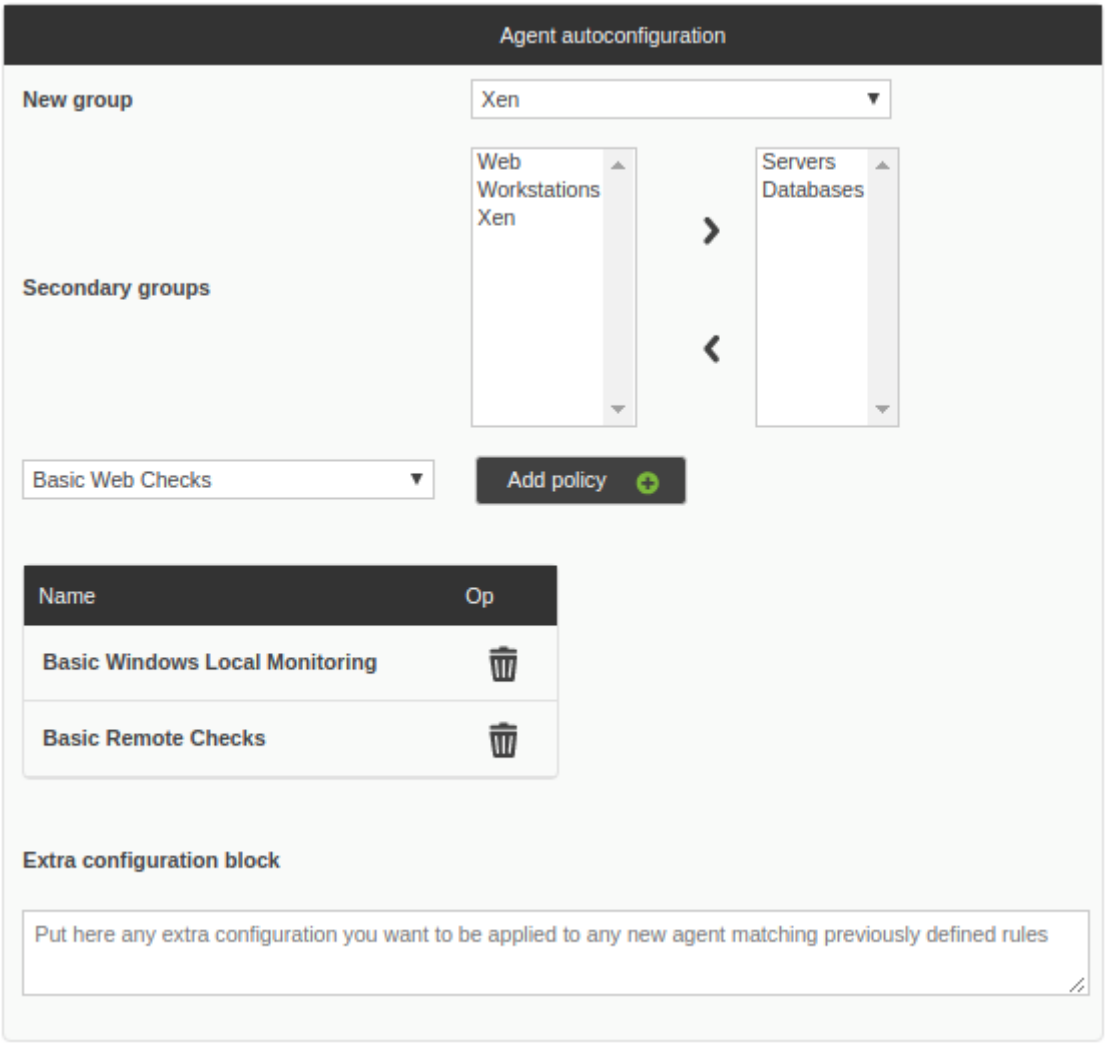
Les groupes sélectionnés ici seront ajoutés en tant que groupes secondaires à l'Agent.

Politiques

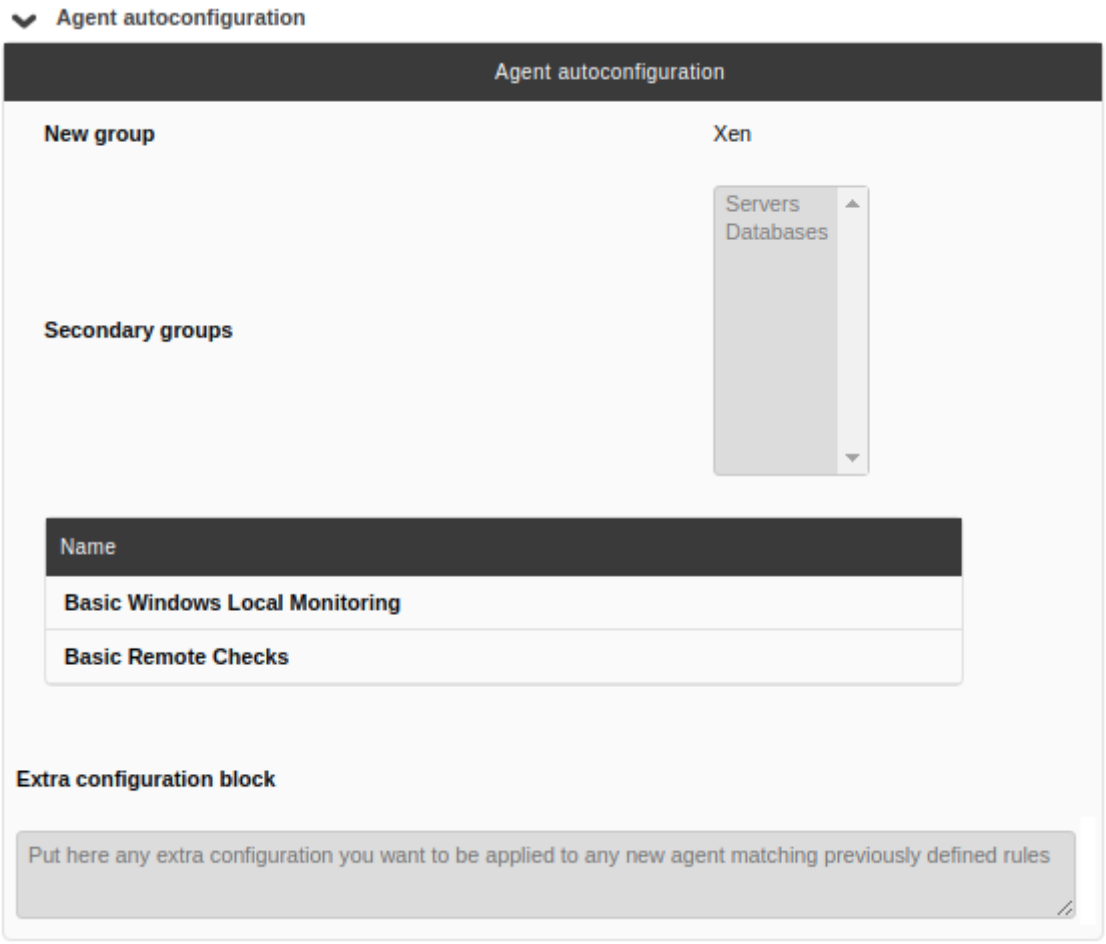
Vous pouvez sélectionner des politiques à appliquer automatiquement lorsque l'agent atteint le serveur.

Bloc de configuration

Il ajoute la configuration brute supplémentaire au fichier de configuration de l'agent.



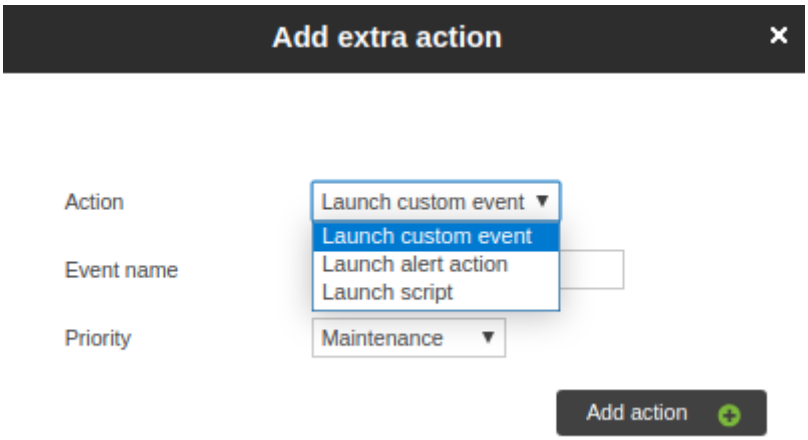
Remarque : Si vous tentez d'accéder à la gestion automatique de la configuration depuis un nœud appartenant à une Métaconsole, avec la gestion centralisée active, la vue sera en lecture seule :



Actions supplémentaires

À partir de cette section, vous pouvez associer d'autres actions à l'autoconfiguration, telles que :

1. Lancer un événement personnalisé (Launch custom event).
2. Exécuter une action d'alerte (Launch alert action).
3. Exécuter un *script* (Launch script).



Le système prend en charge les macros suivantes :

agent

Il sera remplacé par le nom de l'Agent.

agentalias

Il sera remplacé par l'alias de l'Agent.

address

Il sera remplacée par l'adresse IP principale communiquée par l'agent.

agentgroup

Il sera remplacé par le nom du groupe rapporté par l'Agent.

agentos

Il sera remplacé par le système d'exploitation de l'Agent.

agentid

Il est remplacé par l'ID de l'agent.

Agents Unix/Linux

Configuration des agents Unix de Pandora FMS

Les chemins et répertoires fondamentaux à prendre en compte sont :

- `/usr/share/pandora_agent` : Où l'agent Pandora FMS est installé. Sur les systèmes où la stratégie ne le permet pas, il est recommandé de créer un lien vers ce chemin à partir du chemin d'installation réel, par exemple `/opt/pandora` → `/usr/share/pandora_agent` .
- `/etc/pandora/pandora_agent.conf` : Fichier de configuration principal de l'agent. Les modules d'exécution locaux et les *plugins* d'agent sont configurés ici.
- `/usr/local/bin/pandora_agent`> Binaire exécutable de l'agent. Il a généralement un lien vers `/usr/bin/pandora_agent` .
- `/usr/local/bin/tentacle_client`> Binaire exécutable Tentacle, pour transférer des fichiers vers le serveur. Il a généralement un lien vers `/usr/bin/tentacle_client`.
- `/etc/init.d/pandora_agent_daemon` > *Script* de démarrage/arrêt/redémarrage. Sur les systèmes AIX, le démon est`/etc/rc.pandora_agent_daemon`.
- `/var/log/pandora/pandora_agent.log`> Fichier texte dans lequel l'activité de l'Agent Pandora FMS est enregistrée, lorsque l'Agent est exécuté en mode débogage.
- `/etc/pandora/plugins`> Répertoire contenant les *plugins* de l'agent. Il est lié au répertoire `/usr/share/pandora_agent/plugins`.

- `/etc/pandora/collections>` Répertoire contenant les collections déployées sur l'Agent. Il est lié au répertoire `/usr/share/pandora_agent/collections`.

Lancement initial de l'agent Unix

Pour démarrer l'Agent, il suffit d'exécuter :

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

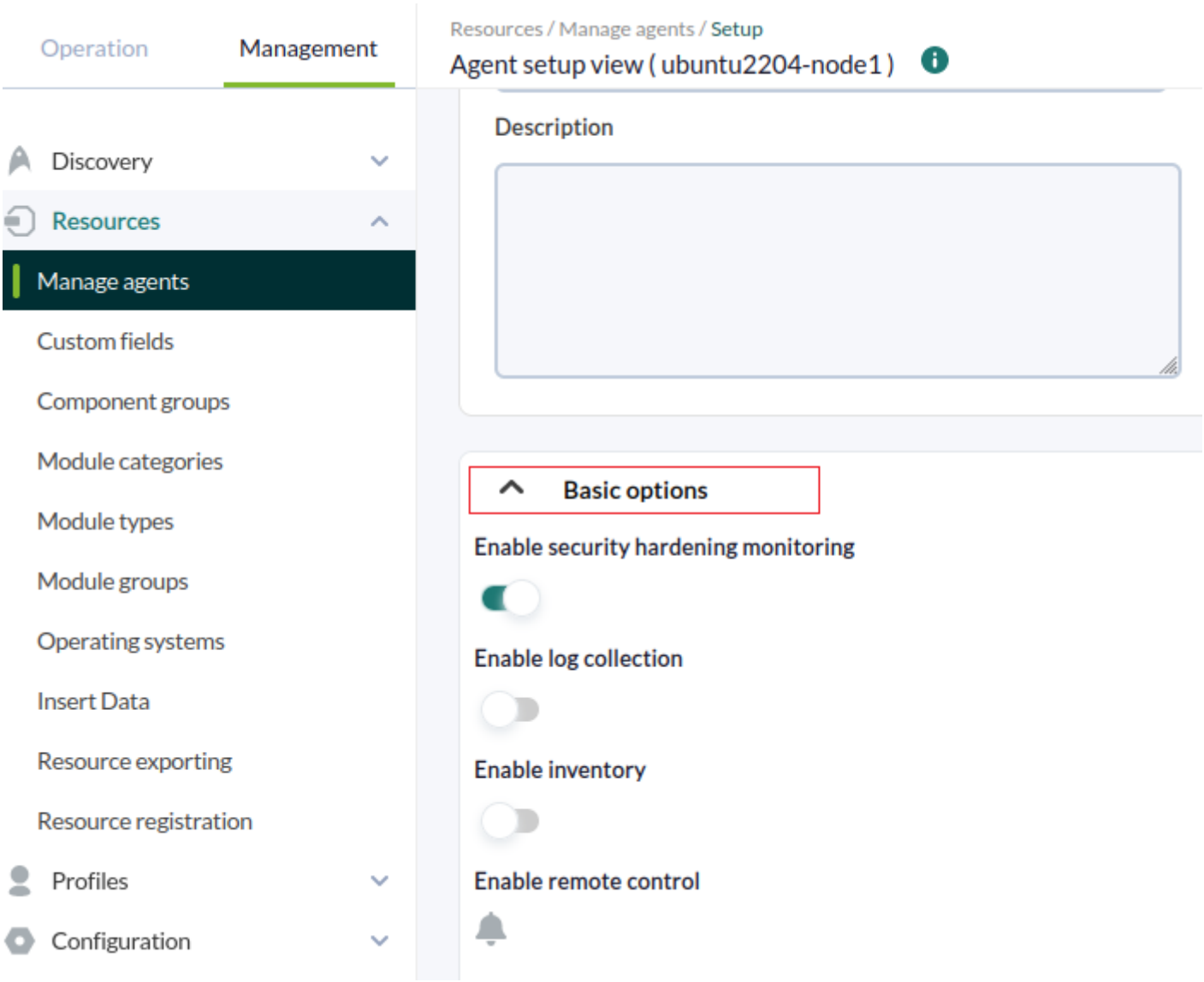
Pour arrêter l'agent, exécutez :

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

Ce *script* de démarrage sera en mesure de démarrer ou d'arrêter l'agent Pandora FMS, qui, une fois démarré, s'exécutera par défaut dans le système comme un démon.

Options de base de l'agent

Si l'agent logiciel a la configuration à distance activée et correspond à une version 774 ou ultérieure, les options suivantes peuvent être activées dans la section Basic options de la configuration de l'agent dans la Console Web.



- Enable security hardening monitoring : Il permet au *plugin* de renforcer la sécurité sur l'appareil supervisé. Les options suivantes seront activées dans le fichier de configuration :

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

Les paramètres sont fixés à un délai d'exécution de 150 secondes (`-t 150`) à un intervalle de 7 jours (7d).

- Enable log collection : Cela permettra de collecter les fichiers journaux pour une analyse judiciaire et de stocker tous les journaux. Les options suivantes doivent être activées dans le fichier de configuration :

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

- Enable inventory : Il active l'option de supervision de l'inventaire. Les options suivantes seront activées dans le fichier de configuration :

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

Modifier la façon dont les agents Unix obtiennent des informations système

Certains modules obtiennent les informations de manière prédéfinie sans qu'il soit nécessaire d'indiquer une commande avec `module_exec`. Ces modules sont les suivants :

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

Il est possible de modifier le fonctionnement de ces Modules par défaut en éditant directement l'exécutable de l'Agent (par défaut `/usr/bin/pandora_agent`). L'agent Pandora FMS est généralement situé à `/usr/bin/pandora_agent`.

Recherchez la chaîne `Commands to retrieve` qui contient le code qui contient les commandes internes. Vous pouvez bien apporter les modifications dont vous avez besoin pour les adapter au système.

```
# 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}\
};
```

Pour modifier l'une des commandes prédéfinies, modifiez simplement le code pour modifier la commande, mais soyez prudent avec les aspects suivants :

1. Vérifiez que les blocs `{ };` se terminent toujours par des points-virgules.

2. Vérifiez que les commandes sont placées entre guillemets simples : ' ' .
3. À votre tour, dans ces guillemets, vous aurez peut-être besoin d'un autre guillemet supplémentaire avec ` ` (notez bien l'exemple ci-dessus).
4. Vérifiez que tous les guillemets simples que vous souhaitez utiliser dans la commande sont précédemment échappés avec le caractère \, c'est-à-dire \ '. Par exemple, cette commande serait normalement :

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

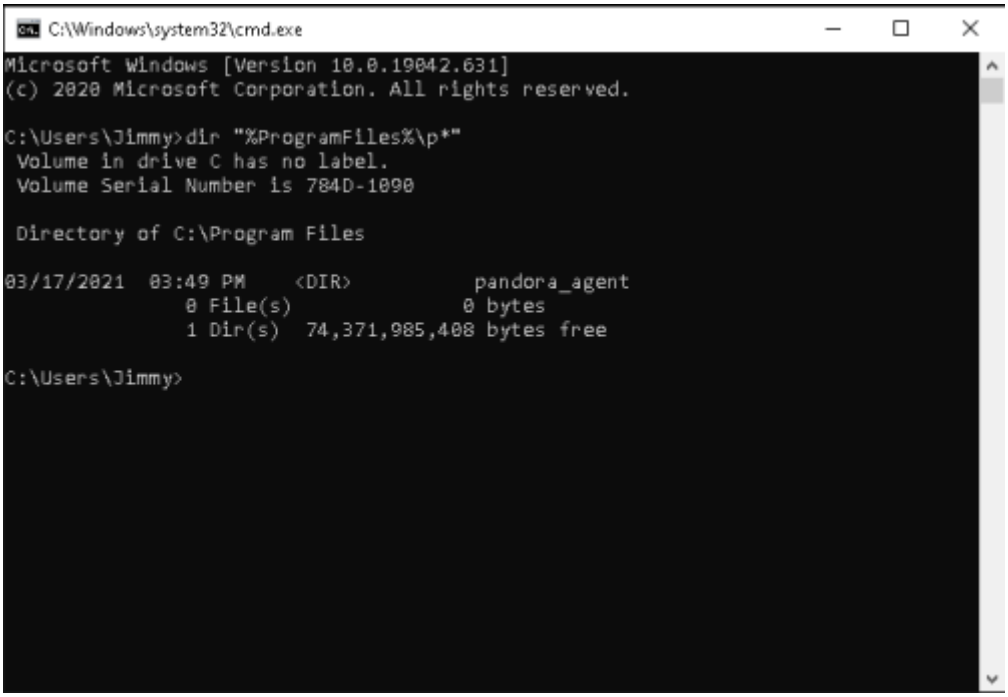
Vous devez l'écrire comme suit :

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

Agents Windows de Pandora FMS

Configuration de l'agent Pandora FMS pour Windows

Les chemins d'accès et répertoires fondamentaux dans les installations de l'Agent pour MS Windows® se trouvent dans le répertoire où l'Agent est installé, par défaut `%ProgramFiles%`.



Les plus importants à garder à l'esprit sont :

`%ProgramFiles%\pandora_agent`

Où l'agent Pandora FMS, son exécutable et ses répertoires sont installés.

`%ProgramFiles%\pandora_agent\pandora_agent.conf`

Fichier de configuration principal de l'agent. Les modules d'exécution locaux et les *plugins* d'agent sont configurés ici.

`%ProgramFiles%\pandora_agent\PandoraAgent.exe`

Binaire exécutable de l’agent.

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

Exécutable binaire Tentacle, pour le transfert de fichiers vers le serveur.

`%ProgramFiles%\pandora_agent\scripts`

Les scripts de démarrage/arrête/redémarrage de l'agent Pandora FMS.

`%ProgramFiles%\pandora_agent\pandora_agent.log`

Fichier texte où l’activité de l’agent Pandora FMS est enregistrée, lorsque l’agent est exécuté en mode débogage.

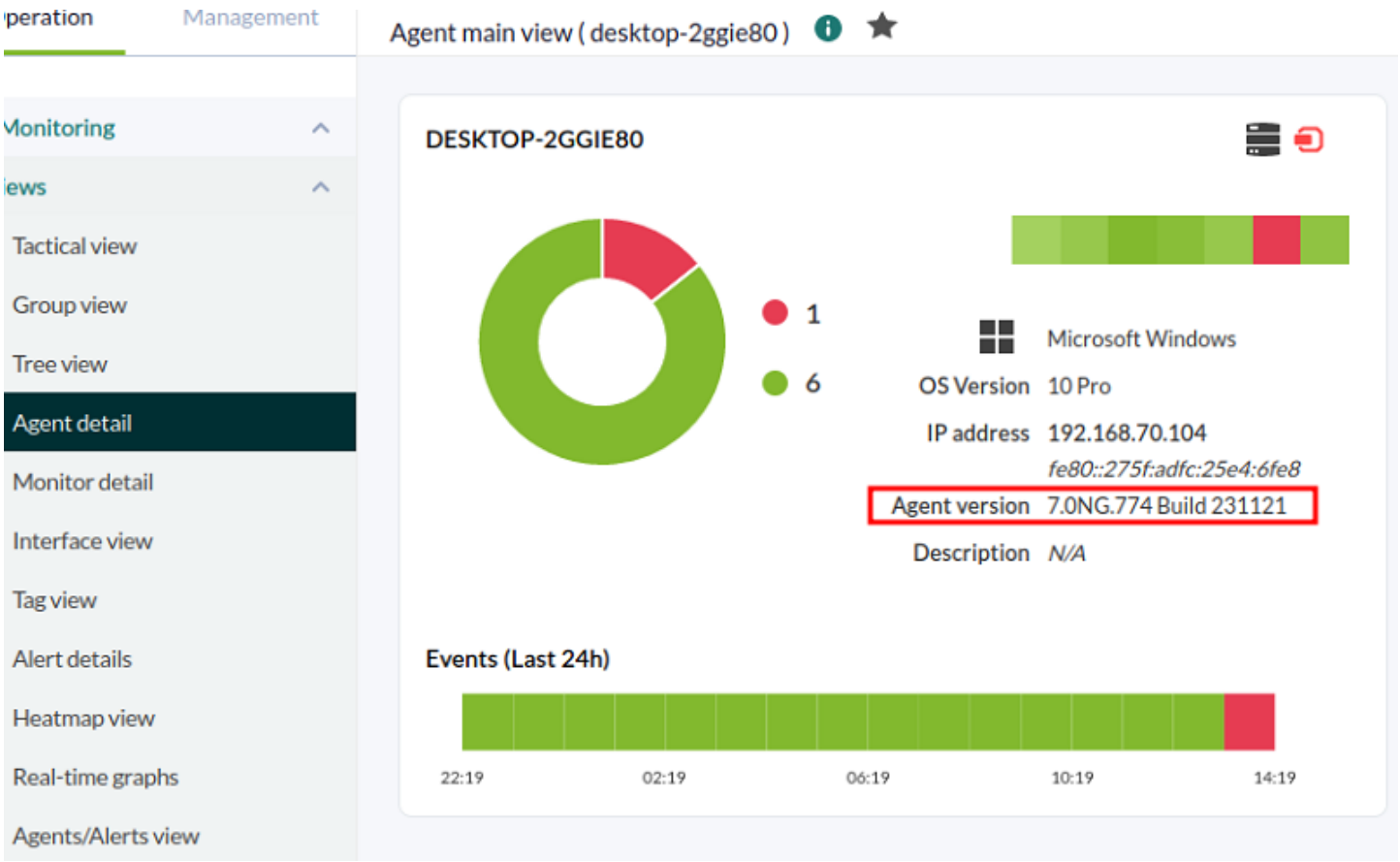
`%ProgramFiles%\pandora_agent\util`

Répertoire qui contient les *plug-ins* de l’agent.

`%ProgramFiles%\pandora_agent\collections`

Répertoire qui contient les collections de l’agent.

Options de base de l'agent pour MS Windows



Lorsque l'agent logiciel a Configuration à distance activée et que la version installée est 774 ou ultérieure, les options suivantes peuvent être activées dans la section Basic options de la configuration de l'agent dans la console Web.

- Enable inventory: Il active l'option de supervision de l'inventaire. Les paramètres suivants doivent être activés dans le fichier de configuration :

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

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

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

module_begin
module_plugin cscript.exe //B //t:20
```

```
"%PROGRAMFILES%\Pandora_Agent\util\cdromdrives.vbs"
module_crontab * 12-15 * * 1
module_end

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

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

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

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

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

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

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

module_begin
```

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

- Enable security hardening monitoring : Il active le *plugin* pour renforcer la sécurité sur l'appareil supervisé. Les paramètres suivants seront activés dans le fichier de configuration :

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

Lorsque les options sont réglées sur un délai d'exécution de 150 secondes (`-t 150`) à un intervalle de 7 jours (`7d`) en tant que période de l'agent.

- Enable log collection : Cette fonction permet de collecter les fichiers journaux à des fins d'analyse judiciaire et de stocker tous les journaux. Les paramètres suivants doivent être activés dans le fichier de configuration :

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

Options de sécurité des agents pour MS Windows

Pour l'agent logiciel PFMS version 775 ou ultérieure, un *plugin* est inclus et est désactivé par défaut. Pour l'activer, vous devez uncomment suivre les instructions suivantes dans le fichier de configuration :

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

Une fois que l'agent logiciel a été redémarré, les modules suivants de l'agent sont collectés :

- Antivirus installé et en cours d'exécution, qu'il s'agisse d'un antivirus Microsoft ou d'un antivirus tiers, et si ses définitions de virus sont à jour (deux modules). Sur MS Windows Server®, cette fonction n'est pas disponible et les modules ne sont donc pas créés.
- Vérifier si le verrouillage automatique de l'écran est actif (État du verrouillage de l'écran), ce qui protège le compte de l'utilisateur lorsqu'il laisse l'ordinateur sans surveillance, sans activité de la souris et du clavier (un module).
- Vérifiez si MS Windows® a été mis à jour (Windows updated®) il y a une semaine ou moins (un

module).
- État du pare-feu, activé ou non (trois modules, un pour chaque profil *firewall* : Réseau de domaine, Réseau privé et Réseau public).
- Vérification que tous les comptes locaux ont un mot de passe (un module).
- Un module est dédié au contrôle de l'activation ou non du journal des tentatives de sessions échouées (uniquement pour les systèmes d'exploitation installés en anglais et en espagnol).

Déploiement automatique d'agents logiciels

Vous pouvez déployer des agents logiciels à l'aide du centre de déploiement via le système Discovery, plus d'informations dans ce lien.

Mise à jour automatique des agents logiciels

L'utilisation des collections de fichiers et de l'outil `pandora_update` peut fournir un moyen de « mettre à jour automatiquement » les agents logiciels.

L'outil `pandora_update` a besoin du module Perl `Digest::MD5` pour fonctionner. À partir de la version 5.14 de Perl, ce module est intégré par défaut, mais dans les versions précédentes, vous devrez l'installer manuellement.

Il fonctionne comme suit :

1. Les agents reçoivent de nouveaux fichiers binaires dans le répertoire d'entrée des collections.

Exemple dans Windows® :

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

Exemple sous Linux® :

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

2. L'agent exécute le *plugin*`pandora_update`. Ce plugin reçoit un seul paramètre : le nom court de la collection (dans cet exemple, `fc_1`). Il analysera le répertoire de la collection à la recherche du binaire de l'agent (et non de l'ensemble du programme d'installation), comparera le binaire situé dans la collection avec celui en cours d'exécution et, s'il est différent, `pandora_update` arrêtera l'agent, remplacera le binaire et redémarrera l'agent à l'aide du nouveau binaire.

Pour mettre à jour différentes architectures, vous devez établir une collection différente pour chaque architecture. Par exemple, si vous souhaitez mettre à jour des agents Windows® 32 bits et 64 bits, vous devez créer deux collections et chacune d'elles inclut le fichier binaire `PandoraAgent.exe` correspondant.

3. `Pandora_update` il écrit également dans un petit *log* l'événement mis à jour, pour pouvoir récupérer lors de la prochaine exécution et avertir l'utilisateur (à l'aide d'un module `async_string`) du processus de mise à jour de l'agent.

Cela implique que les modules utilisés pour terminer le processus de mise à jour peuvent être configurés pour avoir un intervalle élevé.

Installation standard d'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
```

Installation personnalisée 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
```

La commande `pandora_update` accepte comme deuxième paramètre le chemin du répertoire d'installation pandora FMS, il n'est pas nécessaire de le spécifier si l'installation a été effectuée dans le chemin par défaut.

Windows®

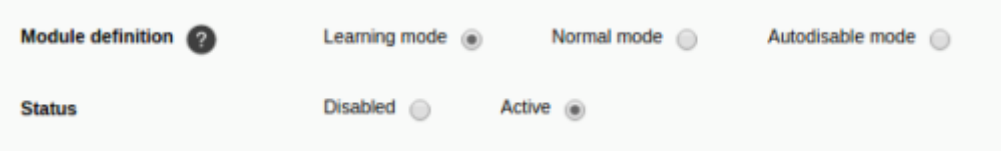
```
 module_begin
 module_name Pandora_Update
 module_type async_string
```

```
 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
```

Auto-création d'agents et de modules à partir de XML

Les agents peuvent être configurés à partir de la console en trois modes de fonctionnement :

- Mode d'apprentissage :
- Si le code XML reçu de l'agent logiciel contient de nouveaux modules, ceux-ci seront automatiquement créés. Il s'agit du comportement par défaut.
- Mode normal : Aucun nouveau module n'arrivera dans le code XML s'il n'a pas été précédemment déclaré dans la console.
- Mode désactivation automatique :
- Semblable au *mode d'apprentissage*, dans ce mode, en outre, si tous les modules passent à un état inconnu, l'agent sera désactivé automatiquement, redevenant activé s'il reçoit de nouvelles informations.



Données incorporées dans la création de l'agent

Les données qui sont incorporées dans l'Agent au moment de sa création automatiquement à la réception d'un XML sont les suivantes :

- Nom de l'agent.
- Adresse IP de l'agent.
- Description de l'agent.
- Agent parent.
- *Timezone offset*.
- Groupe.
- Système d'exploitation.
- Intervalle.
- *Agent version*.
- *Custom fields*.
- *Custom ID*.
- Adresse URL.
- Mode Agent : Apprentissage, Normal, désactivation automatique.

Données mises à jour à partir de l'agent à la réception d'un XML (mode d'apprentissage activé)

- Adresse IP de l'agent.
- Agent parent.
- Version du système d'exploitation.
- Version de l'agent.
- Fuseau horaire.
- *Custom fields*.

Les données SIG sont toujours mises à jour (si elles sont activées), que le *mode d'apprentissage* soit désactivé ou non.

De plus, avec le *mode d'apprentissage* activé, de nouveaux modules seront créés dans Pandora FMS lorsqu'ils seront reçus via le XML.

Données incorporées dans la création d'un module

Les données incorporées dans le module la première fois qu'un code XML est reçu sont les suivantes :

- Nom.
- Type.
- Description.
- Maximum, Minimum.
- Post-traitement.
- Intervalle de module.
- *Min/max Critical*.
- *Min/max Warning*.
- Désactivé.
- Unités.
- *Module group*.
- *Custom ID*.
- *Str. Warning/Critical*.
- Instructions pour critique.
- Instructions d'avertissement.
- Instructions pour étranger.
- *Tags*.
- Investissement critique.
- Investissement d'avertissement.
- Mode « Silencieux ».
- *Min. FF Threshold*.
- *Modèle d'alerte*.
- Crontab.

Données mises à jour à partir d'un module existant à la réception d'un code XML

Lorsque vous recevez un XML qui contient des informations provenant d'un module existant,

seules la description et les informations étendues sont mises à jour, en plus des données du module.

Retour à l'index de documentation Pandora FMS