# ソフトウエアエージェントを使った監視

# ソフトウエアエージェントを使った監視

Pandora FMS ドキュメント一覧に戻る

## ソフトウエアエージェントを使った監視

ソフトウェアエージェントは、情報を収集するオペレーティングシステム上で実行され、モジュールごとにチェックを実行します。

ソフトウェアエージェント独自のディレクティブは、オペレーティングシステムから特定のデータ (CPU 使用率、メモリ、イベントなど) を直接収集するために使用され、事前定義されたスクリプトの指示に従って オペレーティングシステム独自のコマンド を実行します。

Pandora FMS データサーバは、ソフトウェア エージェントによって生成され XML ファイルで送信されたすべての情報を処理し、データベースに保存します。

### エージェント設定

すべての設定とパラメータは pandora_agent.conf ファイルに保存され、このファイルはソフトウェアエージェントと一緒にローカルにインストールされます。 基本的な設定については "Pandora FMS エージェントの設定" で説明しています。高度な設定については以下で説明しています。

#### ローカル設定

ソフトウエアエージェントの設定ファイルでは、以下のテキストの基本構造でモジュールが定義されています。

```
module_begin
module_name <your module name>
module_type generic_data
module_exec <your command>
module_description <your description>
module_end
```

- module_name: モジュールの名前。
- module_exec: 実行されるコマンド。
- module_description: 監視タスクの説明。

例 1

```
module_begin
module_name Files in var spool
module_type generic_data
```

```
module_exec ls /var/spool | wc -l
module_description Number of files incoming dir
module_end
```

*nix 環境 では、コマンド ls はディレクトリファイルを一覧表示します。行 *module_exec* でそれを実行し、行数をカウントする wc コマンドに値を渡します。 この実行によって返される値は、モジュールが取得するデータであり、監視データとして表示されます。

- MS Windows® のソフトウェアエージェントで、module_name に拡張 ASCII 文字 (áéíóú など) を使用したい、または使用する必要がある場合は、外部プラグインまたはスクリプトを使用する必要があります ソフトウェアエージェントプラグインの章 を参照してください。
- MS Windows® 上のソフトウェアエージェントの場合、PowerShell® によるネイティブチェックの実行に module_exec_powershell も使用できます。

例 2

```
module_exec vmstat 1 2 | tail -1 | awk '{ print $13 }'
```

vmstat コマンドは、仮想メモリの統計をレポートします。この例では、必要な情報を “ 絞り込む ” ための 2つの追加コマンドがあります。 最初にコマンドを手動で起動し、出力を分析することをお勧めします。

```
$> vmstat 1 2 | tail -1 | awk '{ print $13 }'
```

結果が要件を満たしていたら、設定ファイルに追加します。ソフトウェアエージェントを介した実行によって返される値は、モジュールデータとして XML に格納されます。

例3

出力が Pandora FMS で受け入れられる値(数値、英数字、または、ブール値)をサポートしていれば、*module_exec* を介して任意のコマンドまたはソフトウェアを実行できます。そのため、カスタムスクリプトを指定することができます。

```
module_exec myScript.pl --h 127.0.0.1 -v cpu
```

この場合も、エージェントはシェルを実行し、オペレーターによって実行されたかのように結果を取得します。

```
$> myScript.pl --h 127.0.0.1 -v cpu
```

### リモート設定

リモート設定を有効にするには、パラメータ remote_config 1 (有効) を設定し、ソフトウェアエージェントを再起動する必要があります。

Pandora FMS Web コンソールからソフトウェアエージェントのファイルをリモートで管理できます。各エージェントの設定は、Pandora FMS サーバの 2 つのファイル (< md5 >.conf と < md5 >.md5) に保存されます。ここで、< md5 > はソフトウェアエージェント名のハッシュです。これらのファイルはそれぞれ次の場所に保存されます。

```
/var/spool/pandora/data_in/conf
```

および

```
/var/spool/pandora/data_in/md5
```

エージェントのリモート設定を有効にすると、設定ファイルにローカルで加えられた変更は、コンソールに保存されている設定によって上書きされます。ソフトウェアエージェントのローカル管理に戻るには、サービスを停止し、remote_config をゼロにリセットして、サービスを再度開始します。

### カスタムフィールド

カスタムフィールドを使用すると、エージェントに追加情報を追加できます。 カスタムフィールドは、PFMS 1.0 API とコマンド set create_custom_field を使用して作成するか、ウェブコンソールのメニュー 管理(Management) → リソース(Resources) → カスタムフィールド(Custom fields) → フィールドの作成(Create field) を使用して作成できます。

- オプション 選択肢の有効化(Enabled combo)、パスワードタイプ(Password type)、リンクタイプ(Link type) は相互に排他的です。つまり、いずれか 1 つだけを使用できます (または、デフォルト値のなし)。
- 前面に表示(Display up front) フィールドを有効にすると、値が設定されている場合、カスタムフィールドの情報がエージェントの概要に表示されます。 さらに、カスタムフィールド情報を コマンドセンター (メタコンソール) に送信するには、このトークンを有効にする必要があります。
- 選択肢の有効化(Enabled combo): このパラメーターを使用すると、ドロップダウンリストからのパラメータ選択を有効化できます。 有効化した場合、対応するカスタムフィールドの構成ウィンドウに新しいフィールドが表示され、カンマで区切で値を入力します。
- パスワードタイプ(Password type): ウェブコンソールでは、フィールド (パスワード) の値がアスタリスクを使用して表示されます。
- リンクタイプ(Link type): ウェブコンソールまたは エージェントから受信した XML によって入力されるウェブリンクをホストするカスタムフィールドを追加できます、JSON 形式の CDATA 命令<![CDATA[…]]> が埋め込まれた XML にリンクを含めることができます。 たとえば、リンクの JSON 形式が次の場合、

```
["Web name","https://example.com"]
```

XML は次のような書式になります。

```
<custom_fields>
   <name>![CDATA[web]]</name>
   <value>![CDATA[["Web name","https://example.com"|]]]</value>
</custom_fields>
```

XML の確認Tentacle プロトコルのセキュリティアーキテクチャ(XML フォーマットでの Pandora FMS データサーバへのデータ送信の仕組)、Pandora FMS データサーバのセキュリティアーキテクチャ(エージェントの自動作成の制限、各エージェントが属するエージェントグループのパスワード設定) を参照してください。

カスタムフィールドは、custom_fieldx_name および custom_fieldx_value トークンを用いて、エージェント設定ファイルから渡すこともできます。 例:

```
custom_field1_name Serial Number
custom_field1_value 56446456KS7000
```

シリアル番号(Serial Number) と呼ばれるカスタムフィールドは、Pandora FMS のインストール時にデフォルトで作成されます。カスタムフィールドは必要に応じて必要なだけ作成でき、それぞれのタイプ (単一値、ウェブリンク、パスワードタイプ、およびオプションリストタイプ) を作成できます。 各カスタムフィールドの数値識別子の順序は関係ありません。名前がまったく同じであることを確認する必要があるだけです。

```
custom_field11_name Simple custom field name
custom_field11_value Simple custom field value

custom_field12_name Custom field Link type
custom_field12_value ["Pandora FMS web site","https://pandorafms.com"]

custom_field13_name Custom field Password type
custom_field13_value My;Password;

custom_field14_name Custom field Combo type
custom_field14_value Two
```

カスタムフィールド Combo type では、ソフトウェア エージェントによって送信される値がその項目の 1 つに正確に一致する必要があります。そうでない場合、値は変更されません。

#### 共通設定パラメータ

基本的なエージェント設定における、最も重要なパラメータ (より詳細は、Pandora FMS ソフトウエアエージェント を参照してください):

- server_ip: Pandora FMS サーバの IP アドレスです。
- server_path: Pandora FMS サーバの 'incoming' フォルダのパスです。デフォルトは、`/var/spool/pandora/data_in` です。
- temporal: ソフトウエアエージェントのテンポラリフォルダです。デフォルトは、`/var/spool/pandora/data_out` です。
- logfile: ソフトウエアエージェントのログファイルです。デフォルトは、`/var/log/pandora/pandora_agent.log` です。
- interval: エージェントの実行間隔です。デフォルトは、`300` です。

#### パスワード保護グループ

デフォルトでは、エージェントが初めて Pandora FMS サーバにデータを送信すると、そのデータはエージェント設定ファイルで定義されているグループに自動的に追加されます。

オプションで、Pandora FMS コンソールからグループのパスワードを設定できます。この場合、エージェント設定ファイルで正しいパスワードが指定されていない限り、エージェントはグループに追加されません。

グループパスワードを編集して追加するには、管理(Management) メニュー → プロファイル(Profiles) → エージェントグループ管理(Manage agent groups) に移動し、グループ名をクリックします。

このグループに新しいエージェントを追加するには、その設定ファイルを編集し、設定オプション `group_password` を追加し、エージェントソフトウェアを再起動します。

### エージェントおよびソフトウエアエージェントのモジュール

#### モジュールの種類

データのタイプに応じた、ソフトウエアエージェントのモジュールのタイプは次の通りです。

- generic_data: 数値および浮動小数点データです。
- generic_data_inc: インクリメンタルな数値データです。前の値と現在の値の差分を間隔で割った値を保存します。このデータタイプは、'秒間の回数' をカウントするのに利用します。たとえば、秒間のログエントリー数、秒間受信バイト数、秒間のコネクション数などです。
- generic_data_inc_abs: 絶対増加する数値データのタイプです。 これは、経過した秒数で割ることなく、前と現在のデータの差分を格納します。したがって、値は 2回の実行の間の差となり、1秒あたりの値にはなりません。このタイプのデータは、ログエントリ、合計受信バイト数、接続数など何かが発生した回数をカウントするために使用されます。
- generic_proc: ブール型のデータです。値が 0 の場合は障害を意味し、0 より大きい場合は正常であることを意味します。 *generic_proc* 型には、あらかじめ設定された障害(0)および正常(1以上)の状態があります。
- generic_data_string: 文字列(テキスト)データです。
- async_data: 非同期の数値データです'generic_data' と同じですが非同期のデータで、変化したときのみ更新されます。非同期データは、収集間隔の定義がありません。
- async_string: 非同期の文字データですgeneric_string と同じですが非同期のデータで、変化したとき

のみ更新されます。データを取得したあとに、次の新たなデータの取得がいつになるかわからないような、ログやイベントをモニタするのに利用します。新しいデータを数秒間隔で受信したり、逆に数日間何もデータがない場合もあります。
- async_proc: 非同期のブーリアンデータです。generic_proc と同じですが非同期のデータで、変化したときのみ更新されます。
- Image module: テキスト文字列型モジュール(generic_data_string または async_string)をもとにしたものです。モジュール内のデータが base64 イメージの場合、つまり文字列の一部に “data:image” が含まれている場合、それは画像データとして識別され、表示画面では画像を表示するウィンドウを開くリンクが有効になります。 また、ヒストリデータとしても文字列として保存され、画像を構築/生成し表示されます。

**ローカルモジュールの実行間隔**

ローカルモジュール(ソフトウエアエージェントのモジュール)には、ベースとしてエージェントの実行間隔があります。ただし、module_interval パラメータにゼロより大きい整数を設定することによって、エージェントの実行間隔にそれを乗じた値をモジュールの実行間隔とすることができます。例:

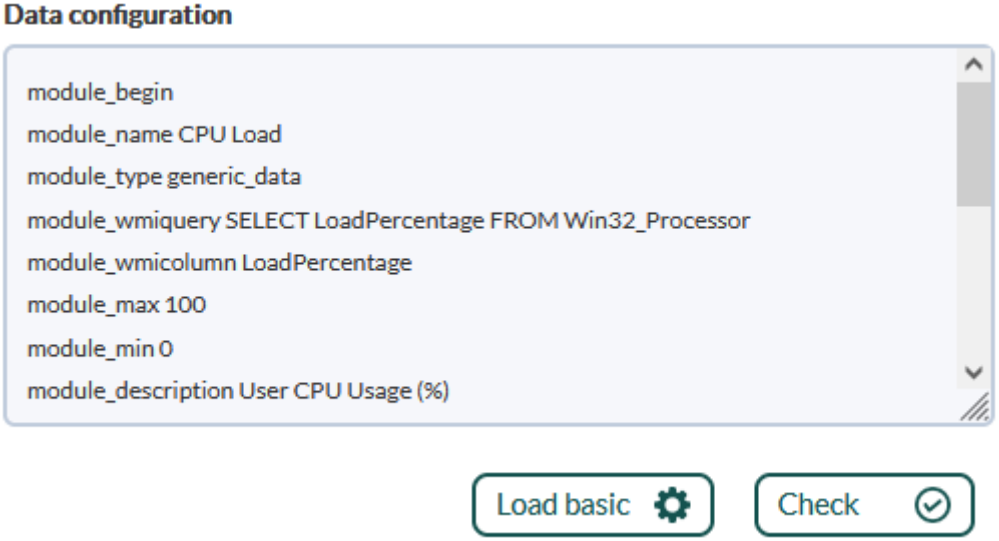
```
module_interval 2
```

この場合、エージェントの実行間隔が 300 のとき、モジュールの実行間隔は 300×2 (600) になります。

**モジュール作成インタフェース**

ソフトウエアエージェントのリモート設定を有効化する必要があります。

コンソールからのローカルモジュールの作成は、リモートモジュール(の閾値、タイプ、グループなど)と同様に、フォームを利用して行います。設定データを指定するテキストボックスがあり、ソフトウエアエージェントの設定ファイルが表示されています。



- 基本(テンプレート)のロード(Load basic (template) をクリックすると、データ設定(Data configuration) のコンテンツが監視のニーズに応じて変更する必要のある基本テンプレートに置き換えられます。
- 変更後、チェック(構文)(Check (syntax)) をクリックすると、テンプレートの構文が正しいことを確認しますが、残りのコマンドはチェックされません。

モジュールがローカルコンポーネントからロードされる場合、マクロが含まれる場合があります。マクロがある場合、設定ボックスは非表示のままになり、マクロごとにフィールドが表示されます。詳細については、テンプレートとコンポーネント を参照してください。

## 状態にもとづくモニタリング

### 事後処理

Pandora FMS ソフトウエアエージェントは、モジュールの実行結果の値に応じたスクリプトの実行をサポートしています。これは、モジュールの実行で得られた値に応じてアクションを行えることを意味します。

例 1

値または値の範囲を示す *module_condition* パラメータを使用して、取得したデータが条件(CPU 使用率が 200%未満)を満たす場合に、実行する内容を設定します。

*module_condition* パラメータにて、モジュールの実行の事後処理を定義します。モジュールから返される値にもとづいて実行されるコマンドを定義します。設定ファイルの例を次に示します。

```
module_begin
module_name CPU_Usage_Condition
module_type generic_data
module_exec get_cpu_usage.pl
module_condition <20 add_processes.sh
module_end
```

例 2

同じモジュールに対して、範囲内や最小しきい値を指定した複数の条件を指定できます(数学的には、両方のオプションのいずれかが実行されるか、まったく実行されません)。

```
module_begin
module_name CPU_Usage_Condition
module_type generic_data
module_exec get_cpu_usage.pl
module_condition (90, 100) remove_processes.sh
module_condition <20 add_processes.sh
module_end
```

例 3

前の例に似ていますが、両方の条件を実行することも、1つまたはまったく実行しないこともできます (選択した値で試してください: 5、15、または 30 の場合)。

```
module_begin
module_name CPU_Usage_Condition
module_type generic_data
module_exec get_cpu_usage.pl
module_condition < 10 start_new_server.sh
module_condition < 20 add_processes.sh
module_end
```

**事前処理**

module_precondition パラメータにて、モジュール実行前の事前処理を定義します。この事前処理により、ソフトウエアエージェントはモジュールを実行するかどうか決めます。

module_precondition に想定する値を割り当て、条件と比較
する必要があります。

例 1

アクティブなプロセスが 10 を超える場合CPU 使用率のパーセンテージを取得し、Pandora FMS サーバにレポートします。

```
module_begin
module_name CPU_Usage
module_type generic_data
module_precondition> 10 number_active_processes.sh
module_exec get_cpu_usage.pl
module_end
```

例 2

同じモジュールに対して複数の前提条件を定義でき、それらすべてが満たされている必要があります。

```
module_begin
module_name CPU_Usage
module_type generic_data
module_precondition> 10 number_active_processes.sh
module_precondition> 1 important_service_enabled.sh
module_exec get_cpu_usage.pl
module_end
```

この場合、アクティブなプロセスが 10 を超えており、そのうちの少なくとも 1つが重要なプロセスである場合にのみモジュールが実行されます。

### 高頻度モニタリング

重要な実行プロセスやサービスなど、特に重要なモジュールが存在する場合があります。 これらのケース向けに、より制御された監視を可能にする集中監視が利用できます。

エージェントの通常の実行間隔を短くすることなく、問題が発生した場合にのみ短い間隔で監視をします。

ソフトウエアエージェントの設定:

- Interval:エージェントの実行間隔を秒で指定します。これは、全ローカルモジュールの一般的な間隔で、必須のパラメータです。
- Intensive_interval: 障害状態のモジュールに対して監視を行う間隔で、オプションパラメータです。

モジュール設定:

- module_intensive_condition = 0: モジュールの値がこのパラメータで指定した値(この場合 0)の場合、エージェントで定義された高頻度の間隔で監視します。

#### 例

sshd サービスは、シェルでリモート接続するために使用されるため非常に重要です。その動作を監視する必要があります。 次の例は、sshd プロセスの監視を通常は 5分ごとに実施しますが、障害時は 10秒ごとに監視したい場合の例です。

```
intensive_interval 10
interval 300
```

```
module_begin
module_name SSH Daemon
module_type generic_data
module exec ps aux | grep sshd | grep -v grep | wc -l
module_intensive_condition = 0
module_end
```

サービスがダウンすると、次の 10秒で再度チェックされます。サービスが起動していれば、次のチェックは通常通り 5分後です。

### 指定時間モニタリング

ソフトウエアエージェントは、モジュールを指定時間に実行する設定ができます。書式は crontab と同じです。モジュールの設定例は次の通りです。以下は、モジュールを毎月曜の 12 から 15時の

間に実行する例です。

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

モジュールを毎時 10分に実行したい場合は、次のようにします。

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

モジュールがデータを出力しない期間を設定した場合、モジュールは“ 不明 ”状態になります。このような場合は非同期モジュールを利用します。

## ソフトウエアエージェントでのリモートチェック

Pandora FMS サーバが(セキュリティ上の理由などから)直接アクセスできないシステムに対してリモートチェックを行う場合に使用します。ソフトウェアエージェントをインストールし、そこからリモートチェックを実行し、ブローカエージェントで配布することができます。

### ICMP チェック

ICMP もしくは ping の監視は、マシンがネットワークにつながっているかどうかを確認するのに便利です。単一のソフトウェアエージェントがすべてのマシンのステータスを簡単に監視できます。

Unix

システムコマンドを利用して ping チェックを実行するモジュールを作成することができます。 (すべてのパラメータは module_exec の“ コマンドライン ”にあります)

```
module_begin
module_name Ping
module_type generic_proc
module_exec ping -c 1 192.168.100.54>/dev/null 2>&1; if [ $? -eq 0 ]; then echo
1; else echo 0; fi
module_end
```

このモジュールの例では、192.168.100.54 宛の ping チェックを実行します。他のホストをチェックしたければIP アドレスを変更するだけです。

Windows

パラメータは、module_ping_count (パケット数、デフォルトは 1) および module_ping_timeout (秒単位のタイムアウト値、デフォルトは 1) で指定する必要があります。以下に例を示します。

```
module_begin
module_name Ping
module_type generic_proc
module_ping 192.168.100.54
module_ping_count 2
module_ping_timeout 5
module_end
```

注意: module_advanced_options で、ping.exe の高度なオプションを指定できます。

### TCP チェック

TCP チェックは、ホストのポートが開いているかを確認するのに便利です。アプリケーションがネットワーク上で応答するかどうかを確認することができます。

Unix

Unix のソフトウエアエージェントでは、次のようなモジュール設定にて TCP チェックを実行します。

```
module_begin
module_name PortOpen
module_type generic_proc
module_exec nmap 192.168.100.54 -p 80 | grep open> /dev/null 2>&1; echo $?; if [
$? == 0 ]; then echo 1; else echo 0; fi
module_timeout 5
module_end
```

このモジュールでは、ホスト 192.168.100.54 の 80番ポートが開いているかどうかをチェックします。

Windows

Windows のソフトウエアエージェントを使っている場合は、TCP チェックを設定するためのパラメータがあります。パラメータは次の通りです。

- module_tcpcheck: チェックするホスト
- module_port: チェックするポート番号
- module_timeout: チェックのタイムアウト値

モジュールの設定例を以下に示します。

```
module_begin
module_name TcpCheck
module_type generic_proc
module_tcpcheck 192.168.100.54
module_port 80
module_timeout 5
module_end
```

このモジュールも同様に、192.168.100.54 の 80番ポートに対してチェックを行います。

### SNMP チェック

SNMPチェックは、一般的にネットワークデバイスのインタフェースのステータス、送受信トラフィックなどをチェックするのに利用します。

Unix

Unix のソフトウエアエージェントを利用している場合は、次のように snmpget コマンドを使ったモジュールを作成します。

```
module_begin
module_name SNMP get
module_type generic_data
module_exec snmpget 192.168.100.54 -v 1 -c public .1.3.6.1.2.1.2.2.1.1.148 | awk
'{print $4}'
module_end
```

このモジュールは、ホスト 192.168.100.54 の OID .1.3.6.1.2.1.2.2.1.1.148 の値を返します。

Windows

Windows のソフトウエアエージェントでは、モジュールを設定するための次のパラメータがあります。

- module_snmpversion [1,2c,3]: SNMP バージョン (デフォルトは 1)
- module_snmp_community <community>: SNMP コミュニティ(デフォルトは public)
- module_snmp_agent <host>: モニタ対象のホスト
- module_snmp_oid <oid>: OID.
- module_advanced_options: snmpget.exe の拡張オプション

モジュールの設定例を以下に示します。

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

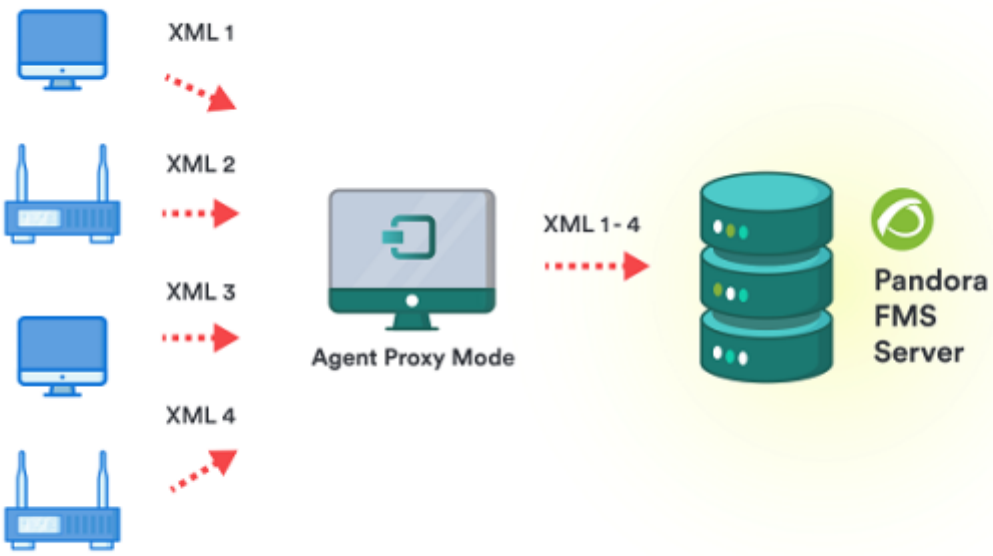
```
module_snmpget
module_snmpversion 1
module_snmp_community public
module_snmp_agent 192.168.100.54
module_snmp_oid .1.3.6.1.2.1.2.2.1.1.148
module_end
```

このモジュールは、Unix のソフトウエアエージェントと同じ動作をします。

## プロキシモード

Linux/Unix システムで、エージェントのプロキシモードを利用するためには、エージェントを root 以外のユーザで実行 する必要があります。そのためPandora FMS エージェントのカスタムインストールが必要です。カスタムインストールの詳細については、エージェントのカスタムインストールを参照してください。

Pandora FMS ソフトウエアエージェントには、エージェントから Pandora FMS サーバへの通信をプロキシするする、プロキシモードがあります。プロキシモードを有効にしたソフトウエアエージェントは、モニタリング処理も実行可能です。




プロキシモードは、1台のマシンのみ Pandora FMS サーバと通信できないといったネットワークの場合にとても便利です。この場合、すべてのマシンにインストールされたソフトウエアエージェン

トは直接 Pandora FMS サーバへは通信できずXML ファイルをプロキシとして動作しているエージェントに送ります。プロキシはそれを Pandora FMS サーバへ送信します。

プロキシモードではXML ファイル送信をプロキシすることに加えて、*リモート設定* および *ファイルコレクション* の機能もサポートしています。

これらの機能により、プロキシモードは接続が制限されたネットワークでのソフトウエアエージェントの「透過的な操作」を提供します。

プロキシモードを有効にするには、ソフトウエアエージェントにて次のパラメータを設定します。

- server_ip: Pandora FMS サーバの IP アドレスです。
- proxy_mode: 有効(1) または 無効(0)。
- proxy_max_connection: プロキシへの最大接続数です。デフォルトは 10です。
- proxy_timeout: プロキシのタイムアウトです。デフォルトは 1秒です。
- proxy_address: プロキシのアドレスです。
- proxy_port: プロキシのポート番号です。

設定例を以下に示します。

```
server_ip 192.168.100.230
proxy_mode 1
proxy_max_connection 20
proxy_timeout 3
```

ソフトウエアエージェントの接続を中継するには、ソフトウエアエージェントの設定で接続先の Pandora FMS サーバのアドレスとしてプロキシモードのエージェントのアドレスを指定します。例えば次の通りです。

プロキシモードが有効なエージェントの IP アドレスが、192.168.100.24 とします。

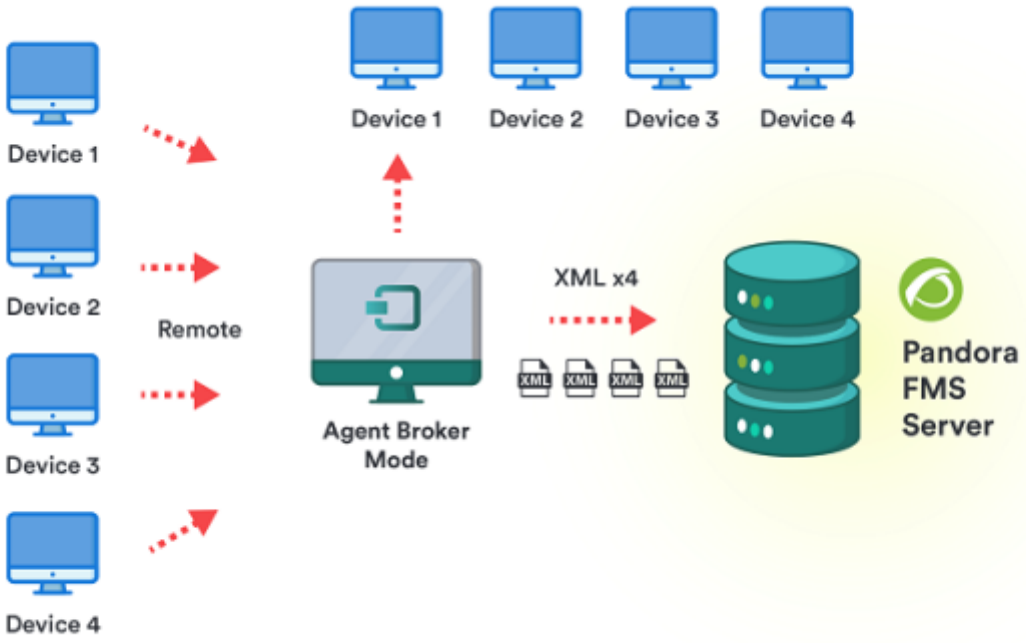
直接 Pandora FMS サーバへ接続できないソフトウエアエージェントでは、server_ip パラメータを次のように設定します。

```
server_ip 192.168.100.24
```

この設定によりPandora FMS サーバと直接通信ができないソフトウエアエージェントが、プロキシモードの他のソフトウエアエージェントを経由して通信できるようになります。リモート設定、ポリシー、ファイルコレクションといったすべての機能が利用できます。

## ブローカーモード

ソフトウエアエージェントには、複数のソフトウエアエージェントがインストールされているかのように一つのエージェントを設定し、モニターするブローカーモードがあります。




ブローカーモードを有効にしたソフトウエアエージェントは、新たな設定ファイルを生成します。同一のマシンで複数のソフトウエアエージェントを動かすのと同じように、オリジナルのソフトウエアエージェントと新たなブローカーがそれぞれの設定ファイルで動作します。

ブローカーモードのメインの機能は次の通りです。

- ローカルのデータを他のエージェントとして送信します。異なるソフトウエアの状態を異なるエージェントとしてモニタリングするのに便利です。
- リモートチェックを行ったデータを、ソフトウエアエージェントがインストールされているマシンから送られたかのように送信します。

ブローカーを設定するには、broker_agent <ブローカー名> という設定行を追加するだけです。次のように、必要な数分 *broker_agent* の設定行を追加するだけで複数のブローカーを作成することが可能です。

```
broker_agent dev_1
broker_agent dev_2
```

ブローカーを設定したら、オリジナルのソフトウエアエージェントの設定と同じような内容で、エージェント名の設定が異なる *dev_1.conf* および *dev_2.conf* ファイルを作成します。*dev_1.conf* および *dev_2.conf* の設定ファイルでのモジュール追加、削除をすることで、ブローカーの設定を変更します。

Pandora FMS ウェブコンソールで、プローカーが表示され、他のエージェントと同じように管理できます 。2つのブローカーでソフトウエアエージェントをインストールしていれば、ウェブコンソールでは、異なる 3つのエージェントでそれぞれのモジュールや設定が参照できることを意味します。

注意:ブローカーエージェントインスタンスは、ファイルコレクションを利用できません。コレクショ

ンを利用したい場合は、インスタンスのコレクションからファイルの“実行”ができますが、インスタンスではなく“メイン”エージェントで配布する必要があります。

ブローカーエージェントが有効な場合、実行中にメモリ内
にデータを保持しているモジュール (module_logevent と
MS Windows® における module_regexp) は動作しません。

## 利用例

### 異なるエージェントとしてのローカルデータベースのモニタリング

マシンの基本的な情報(CPU、メモリ、ディスク)および、インストールされているデータベースの情報を分けてモニタリングしたいとします。

このモニタリングを行うには次のような手段をとります。

- インストールしたソフトウエアエージェント: CPU、メモリ、ディスクをモニタします。
- データベース用のブローカー: データベース内の状態をモニタします。

これを行うには、ソフトウエアエージェントを CPU、メモリ、ディスクをモニタするマシンにインストールします。ソフトウエアエージェントの設定で次の行を追加します。

```
broker_agent DBApp
```

この設定を追加することにより、DBApp というブローカーエージェントを作成します。それにより、dbapp.confという設定ファイルができます。この設定ファイルには、データベースの状態をチェックするモジュールを追加します。

```
module_begin
module_name Num Users
module_type generic_data
module_exec get_db_users.pl
module_end

module_begin
module_name Num slows queries
module_type generic_data
module_exec get_db_slows_queries.pl
module_end
```

これにより、Pandora ウェブコンソールに 2つのエージェントが現れます。一つはマシン名で CPU、メモリ、ディスクのモジュールがあり、もう一つは、DBApp という名前で Num Users および Num slow queries というモジュールがあります。

### ブローカーを使ったリモートデバイスのモニタリング

この例では、Windows マシンにソフトウエアエージェントをインストールし、(CPU、メモリ、ディスクを)モニタリングしています。また、エージェントのインストールなしに 192.168.100.54 の IP を持ったルータをモニタリングしたいとします。この問題を解決するためにブローカーを利用できます。

次の設定で、ブローカーを作成します。

```
broker_agent routerFloor5
```

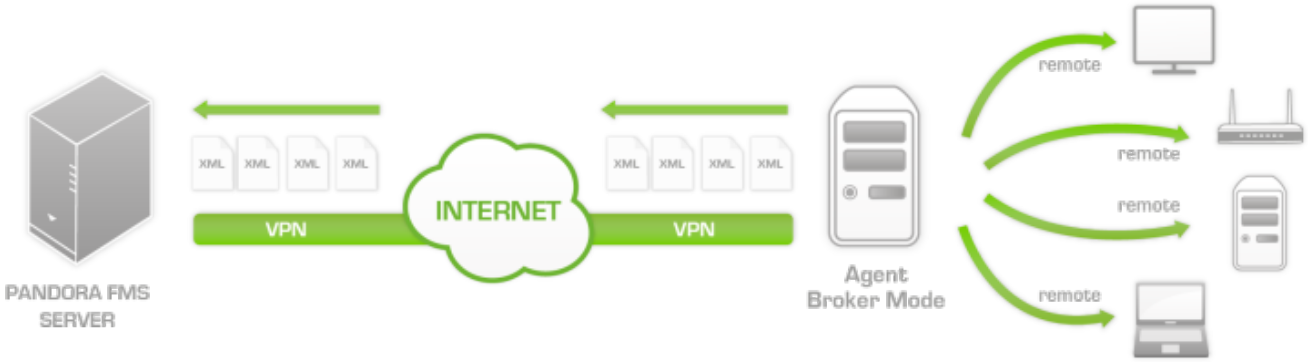
これにより、routerFloor5 という名のブローカーエージェントを作成します。ソフトウエアエージェントが Windows マシンにインストールされているので、Windows のソフトウエアエージェントの機能で ping および snmp でルータをモニタできます。それには、routerFloor5.conf ファイルに次の設定を行います。

```
module_begin
module_name Ping
module_type generic_proc
module_ping 192.168.100.54
module_ping_count 2
module_ping_timeout 500
module_end

module_begin
module_name Eth 1 up
module_type generic_data
module_snmpget
module_snmpversion 1
module_snmp_community public
module_snmp_agent 192.168.100.54
module_snmp_oid .1.3.6.1.2.1.2.2.1.1.1
module_end

module_begin
module_name Eth 2 up
module_type generic_data
module_snmpget
module_snmpversion 1
module_snmp_community public
module_snmp_agent 192.168.100.54
module_snmp_oid .1.3.6.1.2.1.2.2.1.1.2
module_end
```

この例では、Pandora FMS のウェブコンソールには 2つのエージェントが表示されます。一つは CPU、メモリ、ディスクのモジュールを持った Windows マシン、もう一つは、Ping、Eth 1 up、Eth 2 up というモジュールを持った *routerFloor5* です。

### 直接通信できないネットワークのリモートモニタリング

デバイスをリモートからモニタする必要があるがPandora FMS のリモートサーバがそれらに直接通信できない場合があります。




この例では、本社からある会社のサイトのデバイスをリモートからモニタする必要があるとしますPandora FMS サーバは本社にあり、他の会社のサイトに VPN で接続しています。何らかの制限により Pandora のリモートサーバはリモートでアクセスできません。会社のサイトをモニタリングするには、ブローカーモードを使います。ソフトウエアエージェントは、異なるデバイスとして Pandora サーバに XML を送信できます。

ソフトウエアエージェントの設定ファイルでは、モニタするデバイスの数だけブローカーを追加します。設定例は次の通りです。

```
broker_agent device_1
broker_agent device_2
broker_agent device_3
broker_agent device_4
...
```

ブローカーが作成されると、それぞれのデバイスのモニタリングをそれぞれのブローカーの設定ファイルで設定できます。例えばWindows マシンで、device_1 の設定は次の通りです。

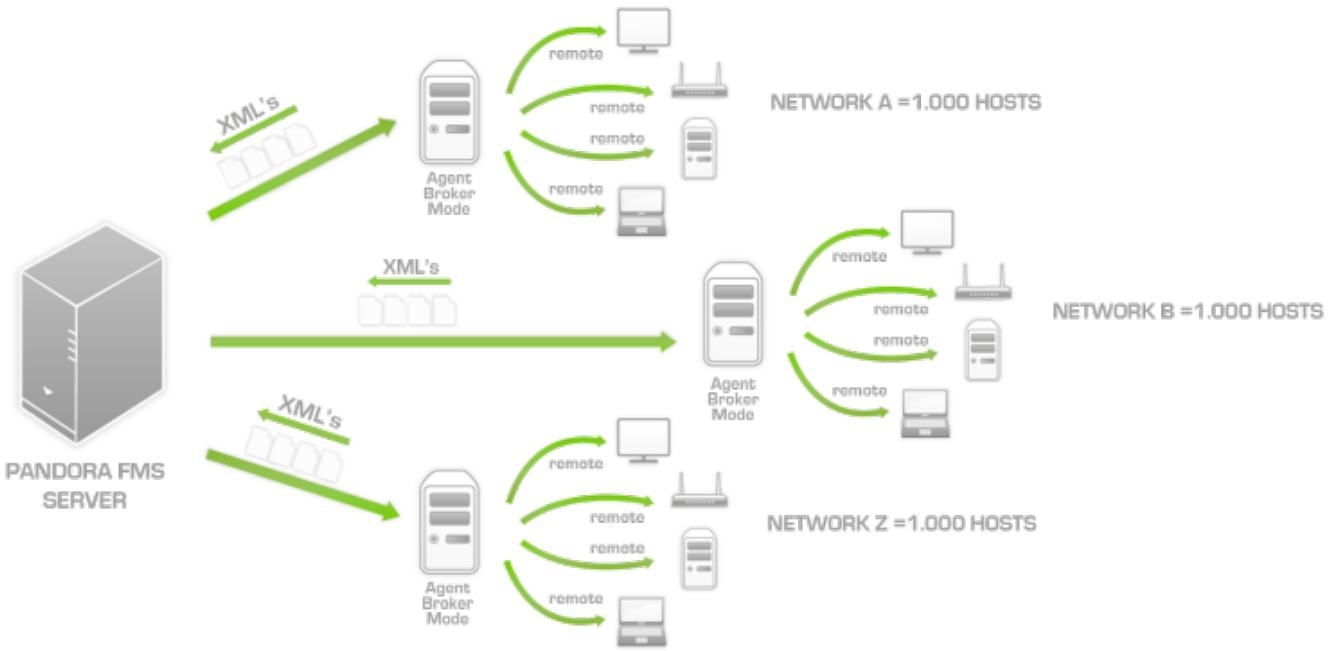
```
module_begin
module_name Ping
module_type generic_proc
module_ping 192.168.100.54
module_ping_count 2
module_ping_timeout 500
module_end

module_begin
module_name CPU_Load
module_type generic_data
module_wmiquery SELECT LoadPercentage FROM Win32_Processor
module_wmicolumn LoadPercentage
```

```
module_end

module_begin
module_name Mem_Free
module_type generic_data
module_wmiquery SELECT LoadPercentage FROM Win32_Memory
module_wmicolumn FreeMemory
module_end

module_begin
module_name Disk_Free
module_type generic_data
module_wmiquery SELECT LoadPercentage FROM Win32_Disk
module_wmicolumn FreeSpace
module_end
```

この設定で、異なる会社のサイト間で通信に制限があったとしても、リモート設定機能を利用でき、また、モニタした情報を Pandora FMS サーバへ送信することができます。

### ブローカーを使ったモニタリング負荷分散

ブローカーモードは、複数のネットワークでモニタリングの負荷を分散するのにとても便利です。




この例では、A から Z の複数のネットワークがあり、それぞれ 1000のデバイスがあります。Pandora FMS のリモートサーバの許容量は、役 2000エージェントです。そのため、負荷分散のためにブローカーモードでソフトウエアエージェントを利用することにします。ブローカーモードを有効にしたソフトウエアエージェントは、リモートでネットワークから全てのデバイスをモニタし、データを XML で Pandora FMS の中央サーバへ送ります。

それぞれのネットワークに、ブローカーモードを有効にしたエージェントがあります。モニタするデバイスの分だけブローカーを作成します。ソフトウエアエージェントの *Broker_Agent_Net_A* の設定は次のようになります。

```
broker_agent device_1
broker_agent device_2
broker_agent device_3
broker_agent device_4
...
```

さらに、それぞれのブローカーには、モニタするデバイスのモジュールを追加します。例えば、ブローカー *device_1* はルータで、次のような設定です。

```
module_begin
module_name Ping
module_type generic_proc
module_ping 192.168.100.54
module_ping_count 2
module_ping_timeout 500
module_end

module_begin
module_name Eth 1 up
module_type generic_data
module_snmpget
module_snmpversion 1
module_snmp_community public
module_snmp_agent 192.168.100.54
module_snmp_oid .1.3.6.1.2.1.2.2.1.1.1
module_end

module_begin
module_name Eth 2 up
module_type generic_data
module_snmpget
module_snmpversion 1
module_snmp_community public
module_snmp_agent 192.168.100.54
module_snmp_oid .1.3.6.1.2.1.2.2.1.1.2
module_end
```

他の例として、ブローカー *device_2* は次のようなモジュールで Windows マシンをモニタします。

```
module_begin
module_name Ping
module_type generic_proc
module_ping 192.168.100.54
module_ping_count 2
module_ping_timeout 500
module_end
```

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

module_begin
module_name Mem_Free
module_type generic_data
module_wmiquery SELECT LoadPercentage FROM Win32_Memory
module_wmicolumn FreeMemory
module_end

module_begin
module_name Disk_Free
module_type generic_data
module_wmiquery SELECT LoadPercentage FROM Win32_Disk
module_wmicolumn FreeSpace
module_end
```

ソフトウエアエージェントをブローカーモードを有効にして使うことで、数千のデバイスからのデータを簡単に負荷分散して収集することができます。

## ソフトウエアエージェントを使ったインベントリ

Pandora FMS ソフトウエアエージェントは、ハードウエアおよびソフトウエアのインベントリの機能をサポートします。インベントリシステムは、企業のサーバで使われている CPU□拡張カード、メモリ、パッチ、ソフトウエアなどの情報を取得することができます。また、インベントリに変化が発生したときにアラートを上げることもできます。例えば、ディスクを交換したり、アプリケーションが削除されたときです。

より詳細の情報は、ソフトウエアエージェントによるローカルインベントリを参照してください。

## ソフトウエアエージェントを使ったログ収集

詳細に関しては、ログ収集と監視 を参照してください。

## UDP リモートコマンド

ソフトウェアエージェントは、リモートリクエストを受信して処理を実行することができます。

UDP は本質的に安全ではないことに注意してください(ただ

し、応答を損なうことなくメッセージを送信するには効率的です)。

Pandora FMS サーバが担当するソフトウェアエージェントに要求を送信できるようにするには、以下を設定します。

- udp_server: この機能を有効化(1)または無効化(0)します。
- udp_server_port: ソフトウエアエージェントの UDP サーバのリスニングポートです。
- udp_server_auth_address: UDP サーバがリクエストを受け付ける IP アドレスです。0.0.0.0 に設定すると、すべての発信元からのリクエストを受け付けます。

変更を適用するためにソフトウエアエージェントを再起動します。

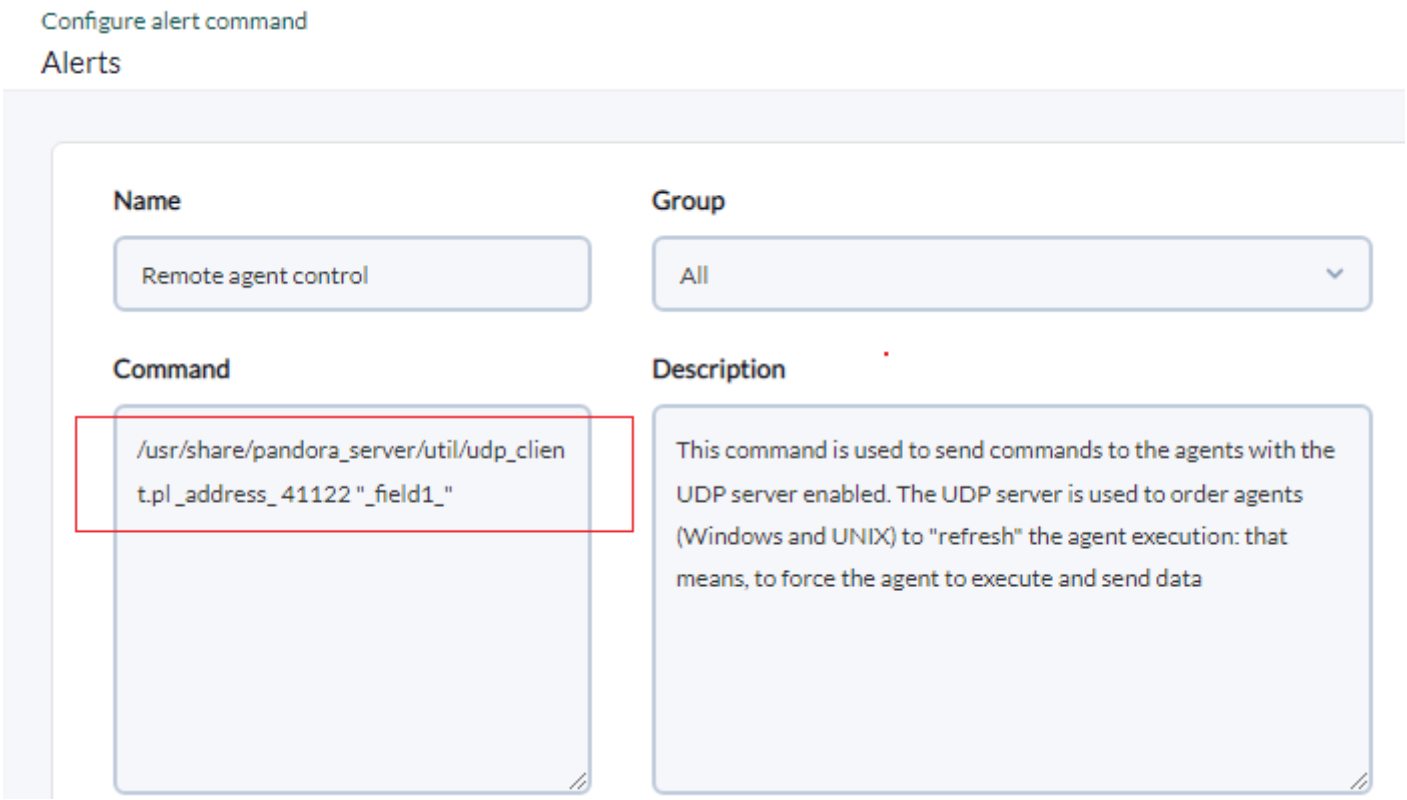
どこからでも要求を受け入れるために *0.0.0.0* に設定できますが、この方法はお勧めしません。 複数の Pandora FMS サーバがある場合や IPv6 を利用している場合は、異なる IP をカンマで区切って設定できます。たとえば IPv6 で 2001:0db8:0000:130F:0000:0000:087C:140B がある場合、その省略形は 2001:0db8:0:130F::87C:140B です。両方をコンマで区切って使用します。

### ソフトウエアエージェントへのサービス再起動要求の送信方法

以下のスクリプトを利用する必要があります。

```
/usr/share/pandora_server/util/udp_client.pl
```

コマンドラインやから実行したり、コンソールの 設定済のコマンド “Remote agent control” を使ったアラートで使うことができます。




**カスタムリモートアクション**

ソフトウェアエージェントサービスの再起動アクションに加えて、カスタムアクションを指定できます。

```
process_<order_name>_start command
```

例えば、リモートから sshd サービスを起動したい場合は次のようにします。

```
process_sshd_start /etc/init.d/sshd start
```

ボタンをクリックするだけで複数のリモートアクションを
実行するスクリプトを呼び出すコマンドを作成することも
できます。

# ソフトウエアエージェントでのプラグインの利用

プラグインは、ソフトウェアエージェントから複雑で高度なチェックを実行することができるものです。単一の値だけではなく、複数のモジュールを返すことができます Pandora FMS サーバによって実行されるサーバプラグインとは異なり、エージェントプラグインはXMLでデータを返し、同時に1つまたは複数のモジュールのデータを返します。

### Windows システムでの実行

Windows では、デフォルトの全プラグインは、VBScript でプログラムされています。これらを実行するには、フルパスで適切なインタプリタを使用する必要があります。

以下に、Windows エージェントにデフォルトで含まれているプラグインのいくつかの使い方を示します。

```
module_plugin cscript.exe //B
"%ProgramFiles%\pandora_agent\util\logevent_log4x.vbs" Aplication System 300
module_plugin cscript.exe //B "%ProgramFiles%\pandora_agent\util\df.vbs"
module_plugin cscript.exe //B "%ProgramFiles%\pandora_agent\util\ps.vbs"
iexplore.exe myapp.exe
```

Windows エージェントでは、さまざまなプラグインを使用できます。

#### PowerShell チェックの利用

バージョン 776 以降には、module_exec_powershell があり、module_exec モジュールの使用ではサポートされていない特殊文字や複雑な命令 (1 つの命令が次の命令に結果を渡す) を含む、より複雑なコマンドを PowerShell に入力できるようになります。

```
# Example of Powershell execution module
module_begin
module_name Powershell
module_type generic_data_string
module_exec_powershell < command_1 > | < command_2 > | … | < command_N >
module_end
```

コマンドは、PFMS ソフトウェアエージェントによって処理されるように引用符無しにそのまま入力します (一方、PowerShell コマンドには引用符が必要な場合があります)。 コマンドが正しくない場合、エージェントログ (ファイル pandora_agent.log) にエラーが追記されます。

### Unix システムでの実行

Unix プラグインは、デフォルトでエージェントのディレクトリの “/plugin” 以下、/etc/pandora/plugins にあります。そのため、このディレクトリにある限りは実行時にフルパスを指定する必要はありません。

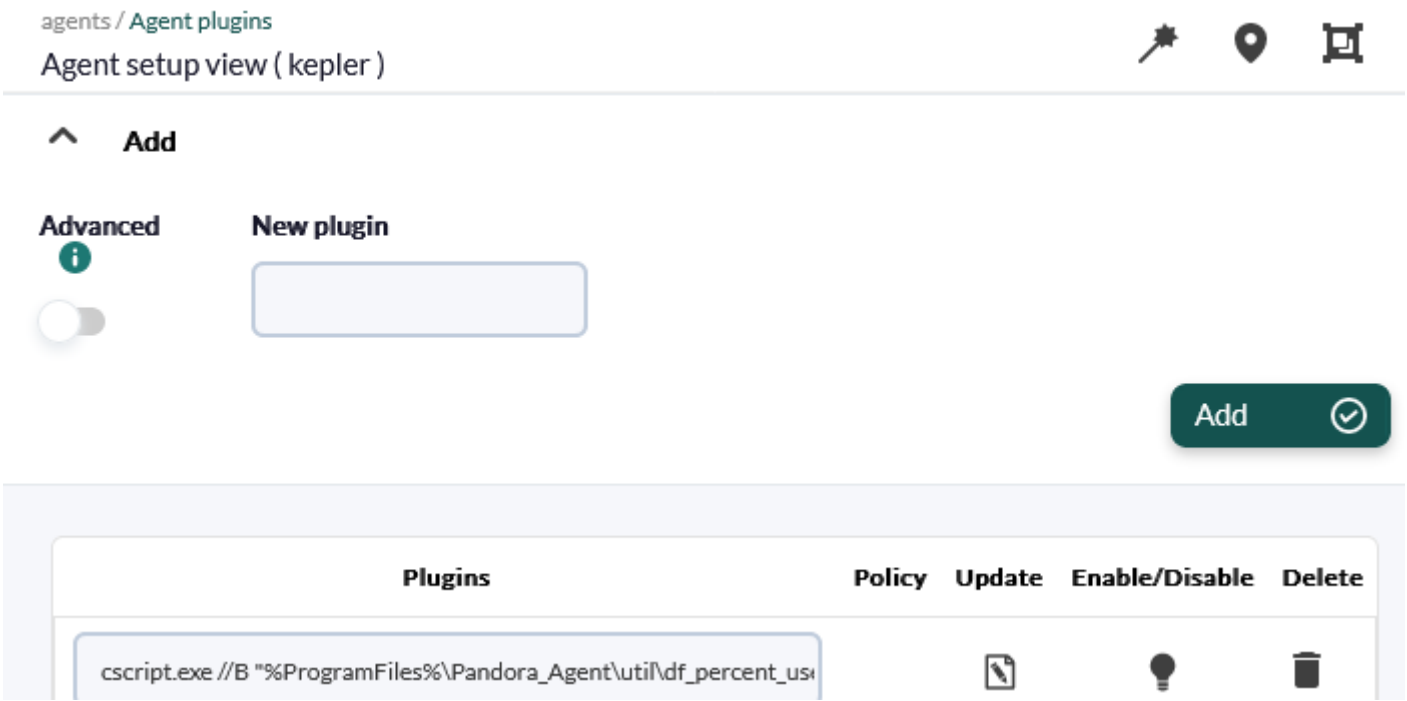
いくつかのプラグイン利用例を以下に示します。

```
  module_plugin grep_log /var/log/syslog Syslog .
  module_plugin pandora_df tmpfs /dev/sda1
```

Unix ソフトウエアエージェントには、デフォルトでいくつかのプラグインが含まれています。

### コンソールからのエージェントプラグインの管理

リモート設定を有効にすると、設定ファイルを直接編集せずに管理することができます。
管理画面のソフトウェアエージェントにプラグインエディタータブが表示されます。




例1

ソフトウエアエージェントのプラグインは、一つのデータもしくはグループ化したデータを返すことができます。Windows で一つのデータを返す *ps.vbs* の例を示します。次の設定でプラグインを実行します。

```
module_plugin cscript.exe //B "%ProgramFiles%\Pandora_Agent\util\ps.vbs"
IEXPLORE.EXE
```

プロセスがダウンしている場合は 0 を返し、プロセスがいる場合は 1 を返すモジュールです。

```
<module>
    <name><![CDATA[IEXPLORE.EXE]]></name>
    <description><![CDATA[Process IEXPLORE.EXE status]]></description>
    <data><![CDATA[1]]></data>
</module>
```

例2

次に Windows で複数のデータを返す *df.vbs* プラグインの例を示します。次の設定でプラグインを実行します。

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

プラグインは見つけたモジュールごとにモジュールを返します。出力結果は次の通りです。

```
<module>
    <name><![CDATA[C:]]></name>
    <description><![CDATA[Drive C: free space in MB]]></description>
    <data><![CDATA[805000]]></data>
</module>

<module>
    <name><![CDATA[D:]]></name>
    <description><![CDATA[Drive D: free space in MB]]></description>
    <data><![CDATA[90000]]></data>
</module>
```

### コンソールからの高度なエージェントプラグイン管理

module_begin タグと module_end タグ内にプラグイン定義を 'カプセル化' すると、エージェントのプラグイン設定にトークンを追加することができます。

module_interval や module_crontab などを、設定に入れることができるようになります。

### 独自エージェントプラグインの作成方法

プラグインは、任意のプログラミング言語で作成することができます。一般的なルール と開発の特別なルールに気を付けてください。

カスタムプラグインの終了コードは 0 になるようにしてください。そうでないとPandora FMS エージェントはプラグインでエラーが発生したものと認識し、出力を無視します。

#### シェルスクリプト(Linux/Unix)によるプラグインの例

```
#!/bin/bash
# Detect if local Mysql is without password
# First, do we have a running MySQL?
CHECK_MYSQL=`netstat -an | grep LISTEN | grep ":3306 "`
if [ ! -z "$CHECK_MYSQL" ]
then
```

```
        CHECK_MYSQL_ROOT=`echo "select 1234" | mysql -u root 2> /dev/null | grep
1234`
        if [ -z "$CHECK_MYSQL_ROOT" ]
        then
        echo "<module>"
        echo "<type>generic_proc</type>"
        echo "<name>mysql_without_pass</name>"
        echo "<data>1</data>"
        echo "<description>MySQL have a password</description>"
        echo "</module>"
        else
        echo "<module>"
        echo "<type>generic_proc</type>"
        echo "<name>mysql_without_pass</name>"
        echo "<data>0</data>"
        echo "<description>MySQL do not have a password</description>"
        echo "</module>"
        fi
fi

exit 0
```

**VBScript (Windows) によるプラグインの例**

```
' df.vbs
' Returns free space for avaible drives.
' ------------------------------------

Option Explicit
On Error Resume Next

' Variables
Dim objWMIService, objItem, colItems, argc, argv, i

' Parse command line parameters
argc = Wscript.Arguments.Count
Set argv = CreateObject("Scripting.Dictionary")
For i = 0 To argc - 1
    argv.Add Wscript.Arguments(i), i
Next

' Get drive information
Set objWMIService = GetObject ("winmgmts:\\.\root\cimv2")
Set colItems = objWMIService.ExecQuery ("Select * from Win32_LogicalDisk")

For Each objItem in colItems
    If argc = 0 Or argv.Exists(objItem.Name) Then
        If objItem.FreeSpace <> "" Then
            Wscript.StdOut.WriteLine "<module>"
            Wscript.StdOut.WriteLine "    <name><![CDATA[" & objItem.Name &
"]]></name>"
```

```
            Wscript.StdOut.WriteLine "    <description><![CDATA[Drive " &
objItem.Name & " free space in MB]]></description>"
            Wscript.StdOut.WriteLine "    <data><![CDATA[" &
Int(objItem.FreeSpace /1048576) & "]]></data>"
            Wscript.StdOut.WriteLine "</module>"
            Wscript.StdOut.flush
        End If
    End If
Next
```

### エージェントでの nagios プラグインの利用

Nagios には多くのプラグインがあり、Pandora FMS で利用することができます。プラグインサーバで nagios 互換モード を使ってリモートでプラグインを利用することができます。 ただし、ステータスを取得するのみで、いくつかの nagios プラグインが持っている詳細情報を扱うことができません。

この問題は、ソフトウエアエージェントで nagios プラグインラッパーを使うことで解決できます。ラッパーは、Unix エージェントにデフォルトで付属しています。Pandora FMS Windows エージェント用の同様のプラグインは、 Pandora FMS リソースライブラリからダウンロードできます。

nagios プラグイン用のプラグインラッパーは何をするのでしょうか。

nagios プラグインをそれ独自のパラメータを使って実行し、出力データを Pandora FMS で使いやすいように、次のような 2種類のデータに変換します。

- ステータス情報: 正常(1)、異常(0)、警告(2)、不明() および、その他(4)。デフォルトでは、proc モジュールを利用します。正常および異常状態は、デフォルトで認識します。もし、警告やその他の状態を利用したい場合は、手動でモジュールのしきい値を設定してください。
- 詳細情報: 通常は文字列データで、モジュールの説明フィールドに入ります。例えば、次のような形になります。

```
<![CDATA["OK: successfully logged in"]]>
```

#### 例

実行権限がついた pop3 プラグイン (/tmp/check_pop3_login) があったとします。これは、pop3 アクセスが正常かどうかチェックします。リモートホストに接続し、ユーザ名とパスワードを送信し、問題ないかを確認します。コマンドラインからは、次のように実行できます。

```
/tmp/check_pop3_login  mail.artica.es sanler@artica.es mypass
```

次のような結果が返ってきます。

```
OK: successfully logged in.
```

異常の場合はつぎのようになります。

```
Critical: unable to log on
```

ラッパーの利用は簡単で、コマンドの前に、ラッパーとモジュール名を設定します。

```
/etc/pandora/plugins/nagios_plugin_wrapper sancho_test /tmp/check_pop3_login
mail.artica.es sanler@artica.es mypass
```

これで、エージェントプラグイン用に完全な XML を生成します。

```
<module>
<name>sancho_test</name>
<type>generic_proc</type>
<data>0</data>
<description><![CDATA[Critical: unable to log on]]></description>
</module>
```

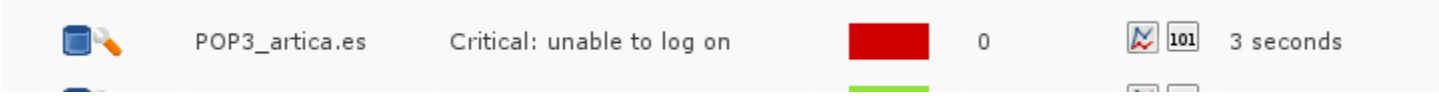
または、

```
<module>
<name>sancho_test</name>
<type>generic_proc</type>
<data>1</data>
<description><![CDATA[OK: successfully logged in.]]></description>
</module>
```

pandora_agent.conf の全設定は次のようになります。

```
module_plugin nagios_plugin_wrapper POP3_artica.es /tmp/check_pop3_login
mail.artica.es sanler@artica.es mypass
```

これは、モジュール内で次のように表示されます。(障害時)




## KeepAlive によるモニタリング

KeepAlive モジュールは、リモート設定が有効になっていない場合でも コンソールから作成することができます。また、pandora_agent.conf ファイルの変更は発生しません。

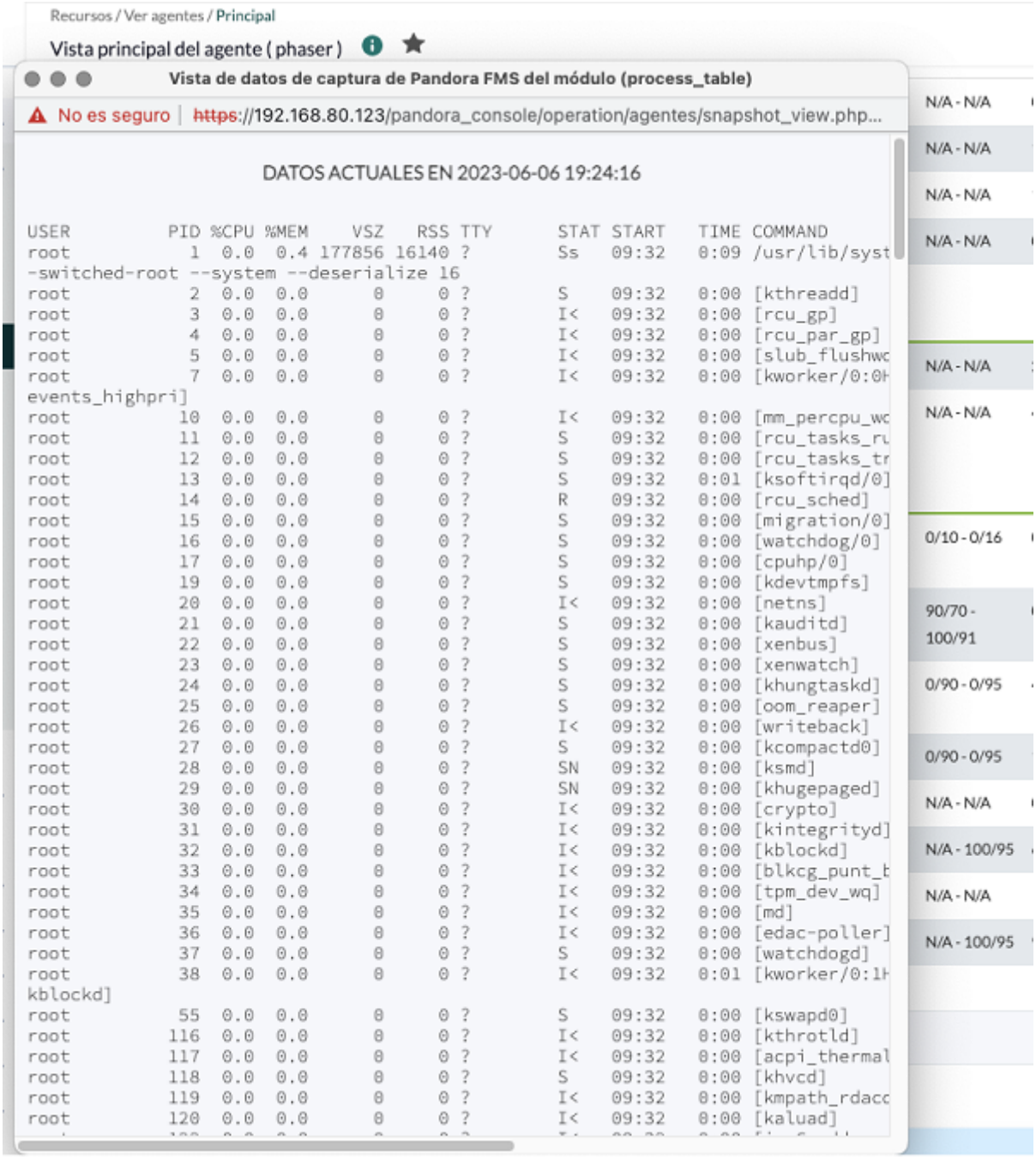
Pandora FMS の固有のモジュールで keep_alive と呼ばれるタイプで、ソフトウェアエージェント

が情報の送信を停止した場合に警告するために使用されます。

設定には、モジュールタブに移動する必要があります (管理(Management) → エージェント管理(Manage agents) → エージェント名をクリック → モジュール(Modules))

モジュールの作成(Create module) をクリックし、データサーバモジュールの新規作成(Create a new data server module) → 作成(Create) → 新しいモジュールの名前を入力 → 作成(Create) を選択します。

## コマンドスナップショット



top や netstat -n などの複数行の出力を持つコマンドは、モジュールによって完全にキャプチャできます。 モジュールはテキストタイプとして設定する必要があります。例:

```
module_begin
module_name process_table
module_type generic_data_string
module_exec ps aux
module_description Command snapshot of running processes
module_group System
module_end
```

- 動作させるには、Pandora FMS コンソール (セットアップ) とこの情報を収集するエージェントの両方を適切に設定し、情報が *未処理* テキストであることを確認する必要があります。
- Web コンソールの設定 で、オプション コマンドラインスナップショット(Command line shapshot) を有効にする必要があります。

## 画像の監視と表示

この監視では、base64 エンコードを使用したテキスト形式の画像を含む文字列型 (generic_data_string または async_string) のモジュールを定義でき、特定の文字列の代わりにその画像を表示できます。

例:

```
#!/bin/bash
echo "<module>"
echo "<name>Actual leader</name>"
echo "<type>async_string</type>"
echo "<data><![CDATA[....]]></data>"
echo "</module>"
```

上記のファイルをエージェント上で保存し(またはファイルコレクションで配り)し、次のように実行します。

```
module_plugin <ファイルへのフルパス>
```

## Windows における特定の監視

- プロセスの名前に空白が含まれている場合は、“ ” を使用しない でください。
- プロセスの名前は、拡張子 .exe を含め、Windows のタスク管理 （taskmngr ) に表示されているものと同じである必要があります。
- 大文字と小文字を尊重する ことが重要です。

## プロセスモニタリングと、プロセスウォッチドック

### プロセスモニタリング

module_proc パラメータは、指定した名前のプロセスがそのマシンで動いているかどうかをチェックします。例:

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

module_async yes のパラメータを指定する必要があります。

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

### プロセスウォッチドック

Windows のソフトウエアエージェントのウォッチドック機能は、プロセスがダウンしたときに再起動することができます。ウォッチドックは、モジュールが非同期の場合にのみ動作します。

ウォッチドックモジュールの設定例を以下に示します。

```
module_begin
module_name Notepad
module_type generic_data
module_proc notepad.exe
module_description Notepad
module_async yes
module_watchdog yes
module_user_session yes
module_start_command "%SystemRoot%\notepad.exe"
module_startdelay 3000
module_retrydelay 2000
module_retries 5
module_end
```

notepad.exe プロセスがいなくなるたびに、

```
%SystemRoot%\notepad.exe
```

が実行されます(Windows の章の冒頭にある一般的なルールを参照してください)。 また、再起動のリトライは 3秒間隔で 5回まで実施し、1回のタイムアウトは 2秒という設定です。この例では notepad.exe プロセスはユーザのセッションで起動します。

## サービスモニタリングと、サービスウォッチドック

### サービスモニタリング

module_service パラメータは、指定したサービスがマシンで動作しているかどうかをチェックします。モジュールの設定例を以下に示します。

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

サービス名にスペースが含まれる場合、« “ “ » は使わないようにしてください。サービス名を見つけるには Windows サービスマネージャのサービス名フィールドを見てください。大文字、小文字の確認が重要です 

サービスがダウンしたときにソフトウエアエージェントがすぐに通知して欲しい場合は、module_async yes を追加する必要があります。モジュールの設定例を以下に示します。

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

### サービスウォッチドック

プロセスと同様に、ダウンしたサービスを再起動できるウォッチドックモードがあります。ウォッチドックを使ったモジュール定義例は次の通りです。

```
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
```

ウォッチドックの定義には、特別なパラメータは必要ありません。サービスの定義内にあるためです。

## 基本リソースのモニタリング

MS Windows® 用 Pandora FMS ソフトウェアエージェントをインストールすると、必要な基本モジュールが含まれています。その一部は有効になっていますが、その他は リモート設定 (またはエージェントの .conf ファイルをローカルで編集) によって有効化する必要があります。

### CPU のモニタリング

CPU をモニタするには、CPU 使用率を返す module_cpuusage パラメータを利用します。

次のような設定で、CPU の ID を元に CPU をモニタすることができます。

```
module_begin
module_name CPU_1
module_type generic_data
module_cpuusage 1
module_description CPU usage for CPU 1
module_end
```

また、次のように、すべての CPU の平均使用率をモニタすることもできます。

```
module_begin
module_name CPU Usage
module_type generic_data
module_cpuusage all
module_description CPU Usage for all system
module_end
```

### メモリのモニタリング

メモリをモニタするには、システムの空きメモリ容量を返す module_freememory と、空き率をパーセンテージで返す module_freepercentmemory の 2つのパラメータを利用できます。

module_freememory を使ったモジュールの例を以下に示します。

```
module_begin
module_name FreeMemory
module_type generic_data
module_freememory
```

```
module_description Non-used memory on system
module_end
```

module_freepercentmemory を使ったモジュールの例を以下に示します。

```
module_begin
module_name FreePercentMemory
module_type generic_data
module_freepercentmemory
module_end
```

### ディスクのモニタリング

ディスクスペースをモニタするには、空き容量を返す module_freedisk と、空き率をパーセンテージで返す module_freepercentdisk の 2つのパラメータを利用できます。両方のパラメータには、モニタするドライブ名の指定が必要で、 ”:” を忘れないように してください。

module_freedisk を使ったモジュールの設定例を以下に示します。

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

module_freepercentdisk を使ったモジュールの設定例を以下に示します。

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

### WMI クエリ

Pandora FMS のソフトウエアエージェントは、システムに関連した情報や外部の情報を保持するのに使われる共通の技術である WMI クエリおよび ODBC 接続を使って情報を取得することができます。

module_wmiquery パラメータを使って、ソフトウエアエージェントはローカルで WMI クエリを実行することができます。クエリを実行するには、実行するクエリを module_wmiquery パラメータで設定し、取得したい情報を持つカラムを module_wmicolumn で指定します。

例えば、次の設定ではインストールされているサービス一覧を取得できます。

```
module_begin
module_name Services
```

```
module_type generic_data_string
module_wmiquery Select Name from Win32_Service
module_wmicolumn Name
module_end
```

WMI を使って CPU ロードも取得できます。

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

Pandora FMS ドキュメント一覧に戻る