Servers Plugin Développement

Servers Plugin Développement

Caractéristiques de base du plugin serveur

Le serveur *plugin* est exécuté par le Pandora FMS server Plugin, il doit donc présenter certaines caractéristiques particulières:

- Chaque exécution de *plugin* doit renvoyer une seule valeur. Cela doit être le cas parce que le Plugin server effectue une exécution pour chaque module *plugin*.
- Il doit pouvoir accéder à distance aux ressources à surveiller.
- Il est possible d'utiliser n'importe quel langage de programmation pris en charge par le système d'exploitation sur lequel le Pandora FMS server est installé.
- Toutes les dépendances ou logiciels nécessaires à l'exécution du *plugin* doivent être disponibles ou installés sur la même machine que celle qui exécute le Pandora FMS server.

Vous trouverez plus d'informations sur la surveillance à l'aide du serveur distant *plugins* à ce lien. Vous y trouverez des exemples plus simples et la manière dont ils fonctionnent avec les modules et les agents qui les contiennent. Dans cet article, la création du serveur *plugin* est abordée en détail.

Exemple de développement d'un plugin serveur

Le *plugin* suivant renvoie la somme du trafic entrant et sortant sur l'interface d'un appareil, les données étant obtenues par SNMP.

Le code *plugin* serait le suivant:

```
#!/usr/bin/perl -w

use strict;
use warnings;

sub get_param($) {
        my $param = shift;
        my $value = undef;

        $param = "-".$param;

        for(my $i=0; $i<$#ARGV; $i++) {

                if ($ARGV[$i] eq $param) {
                        $value = $ARGV[$i+1];
```

```
                        last;
                }

        }
        return $value;
}

sub usage () {
        print "iface_bandwith.pl version v1r1\n";
        print "\nusage: $0 -ip <device_ip> -community <community> -ifname
<iface_name>\n";
        print "\nIMPORTANT: This plugin uses SNMP v1\n\n";
}

#Global variables
my $ip = get_param("ip");
my $community = get_param("community");
my $ifname = get_param("ifname");

if (!defined($ip) ||
        !defined($community) ||
        !defined($ifname) ) {
        usage();
        exit;
}

#Browse interface name
my $res = `snmpwalk -c $community -v1 $ip .1.3.6.1.2.1.2.2.1.2 -On`;

my $suffix = undef;

my @iface_list = split(/\n/, $res);

foreach my $line (@iface_list) {

        #Parse snmpwalk line
        if ($line =~ m/^([\d|\.]+) = STRING: (.*)$/) {
                my $aux = $1;

                #Chec if this is the interface requested
                if ($2 eq $ifname) {

                        my @suffix_array = split(/\./, $aux);

                        #Get last number of OID
                        $suffix = $suffix_array[$#suffix_array];
                }
        }
}

#Check if iface name was found
```

```
if (defined($suffix)) {
        #Get octets stats
        my $inoctets = `snmpget $ip -c $community -v1
.1.3.6.1.2.1.2.2.1.10.$suffix -OUevqt`;
        my $outoctets = `snmpget $ip -c $community -v1
.1.3.6.1.2.1.2.2.1.16.$suffix -OUevqt`;

        print $inoctets+$outoctets;
}
```

Une partie importante du code est la fonction usage:

```
sub usage () {
        print "iface_bandwith.pl version v1r1\n";
        print "\nusage: $0 -ip <device_ip> -community <community> -ifname
<iface_name>\n";
        print "\nIMPORTANT: This plugin uses SNMP v1\n\n";
}
```

Cette fonction décrit la version et la manière d'utiliser *plugin*, elle est très importante et devrait toujours être affichée lors de l'exécution de *plugin* sans aucun paramètre ou avec une option `-h` ou `-h`:

```
--help
```

La valeur renvoyée par *plugin* est imprimée sur la sortie standard à l'avant-dernière ligne avec l'instruction suivante:

```
print $inoctets+$outoctets;
```

La valeur renvoyée par le plugin *plugin* est une donnée unique que le Plugin server de Pandora FMS ajoutera ultérieurement comme donnée au module associé.

Pour pouvoir exécuter ce serveur *plugin*, il sera nécessaire d'installer les commandes snmpwalk et snmpget sur la machine qui exécute le serveur Pandora FMS.

Enregistrement manuel d'un plugin dans la console

- Plugin type:

La principale différence avec PFMS est que les plugins de Nagios *renvoient un niveau d'erreur pour indiquer si le test a réussi ou non*. Si vous souhaitez obtenir des données, et non un état (bon/mauvais), vous pouvez utiliser un plugin de type Nagios en mode “Standard” (pour MS Windows® vous utilisez Nagios wrapper for agent plugin).

- Max. timeout:

Il s'agit du délai d'expiration du module complémentaire. Si aucune réponse n'est reçue dans ce délai, le module sera marqué comme inconnu et sa valeur ne sera pas mise à jour. Il s'agit d'un facteur très important lors de la mise en œuvre de la surveillance avec des *plugins*, car si le temps d'exécution du *plugin* est supérieur à ce nombre, vous ne pourrez jamais obtenir de valeurs à partir de ce dernier. Cette valeur doit toujours être supérieure au temps normalement nécessaire pour que le *script* ou l'exécutable utilisé comme *plugin* renvoie une valeur. Si rien n'est spécifié, la valeur indiquée dans la configuration en tant que `plugin_timeout`.

- Plug-in command:

Il s'agit du chemin où se trouve la commande du *plugin*. Par défaut, si l'installation a été standard, ils seront dans le répertoire:

```
/usr/share/pandora_server/util/plugin/
```

Cependant, il peut s'agir de n'importe quel chemin dans le système.

Dans ce cas, tapez `/usr/share/pandora_server/util/plugin/udp_nmap_plugin.sh` dans le champ.

Le serveur exécutera ce *script*, il doit donc disposer de permissions d'accès et d'exécution.

- Plug-in parameters:

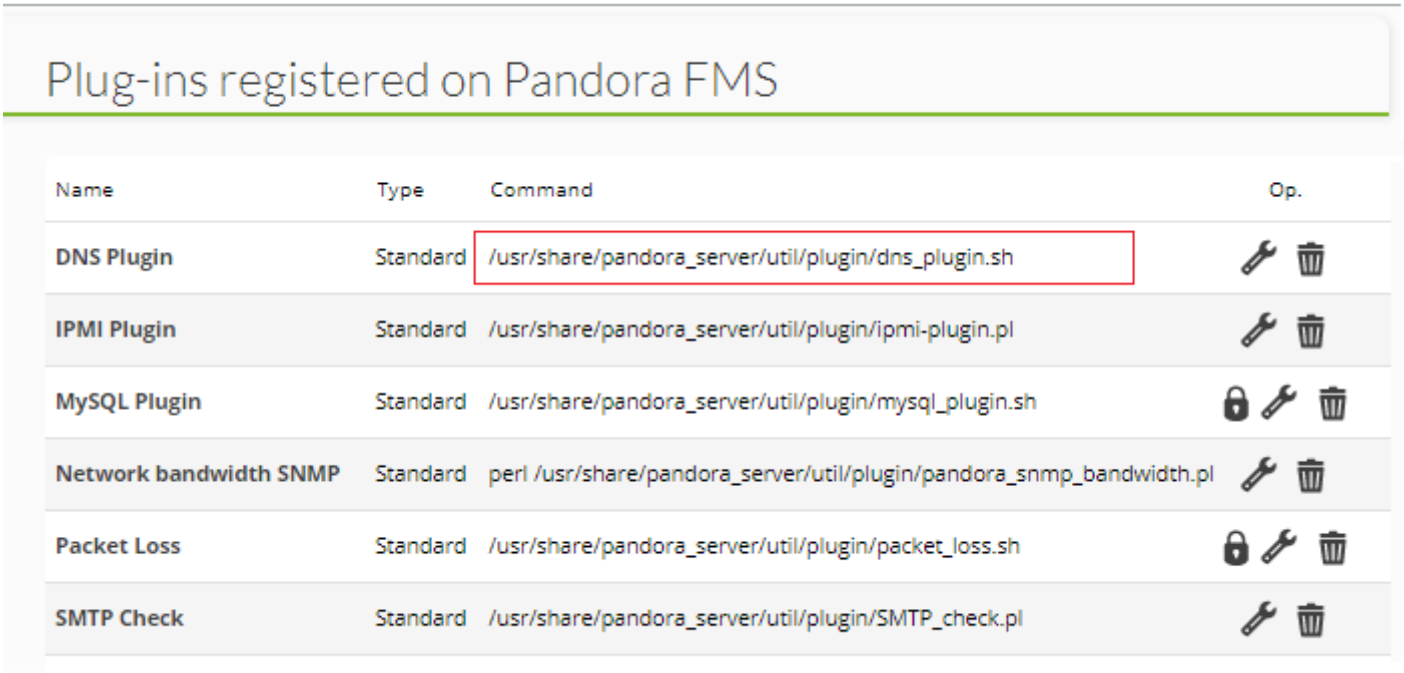
Une chaîne contenant les paramètres *plugin*, qui suivront la commande et un espace vide. Ce champ accepte des macros telles que `_field1_ _field2_ … _fieldN_`. La section suivante explique le fonctionnement des macros.

- Parameters macros:

Il est possible d'ajouter un nombre illimité de macros à utiliser dans le champ de paramètres *plugin*. Ces macros apparaissent comme des champs de texte dans la configuration du module. La section suivante explique le fonctionnement des macros.

Macros de plugin

Prenons l'exemple du DNS Plugin installé par défaut sur un FMS Pandora server.



Ce module complémentaire permet à un domaine web et à son adresse IP correspondante d'être vérifiés par un serveur de résolution de domaine (serveur DNS).

- `-i` : Adresse IP connue du domaine.
- `-d` : Domaine web correspondant.
- `-s` : Serveur DNS à interroger.

Connaissant ces trois paramètres, procédez à l'enregistrement manuel d'un nouveau *plugin*, mettez dans le nom New DNS Plugin et dans la description recopiez le résumé précédent et aussi indiquez qu'il faut que le Module collecte pour le monitoring une valeur fausse (zéro) ou vraie (un). Ce type de données est également connu sous le nom de *boolean* et dans Pandora FMS est appelé `generic_proc`.

Dans la section des paramètres macro (Macro parameters) ajoutez trois champs macro `field1`, `field2` y `field3`.

Pour `_field1_` mettez la description correspondant au paramètre `-i` et ainsi de suite pour les paramètres `-d` et `-s`. Laissez les valeurs par défaut (default value) vides, écrivez dans l'aide (Help) de chacune d'elles le texte que vous jugez opportun.

Dans la zone de texte Commande de plugin, tapez:

```
/usr/share/pandora_server/util/plugin/dns_plugin.sh
```

Dans la zone de texte Plugin parameters, tapez:

```
-i _field1_ -d _field2_ -s _field3_
```

Lorsque ce plugin est utilisé dans un module, le DNS plugin est exécuté en remplaçant

chacun des champs par les paramètres écrits dans le module.

```
/usr/share/pandora_server/util/plugin/dns_plugin.sh -i _field1_ -d _field2_ -s
_field3_
```

Fonctionnement

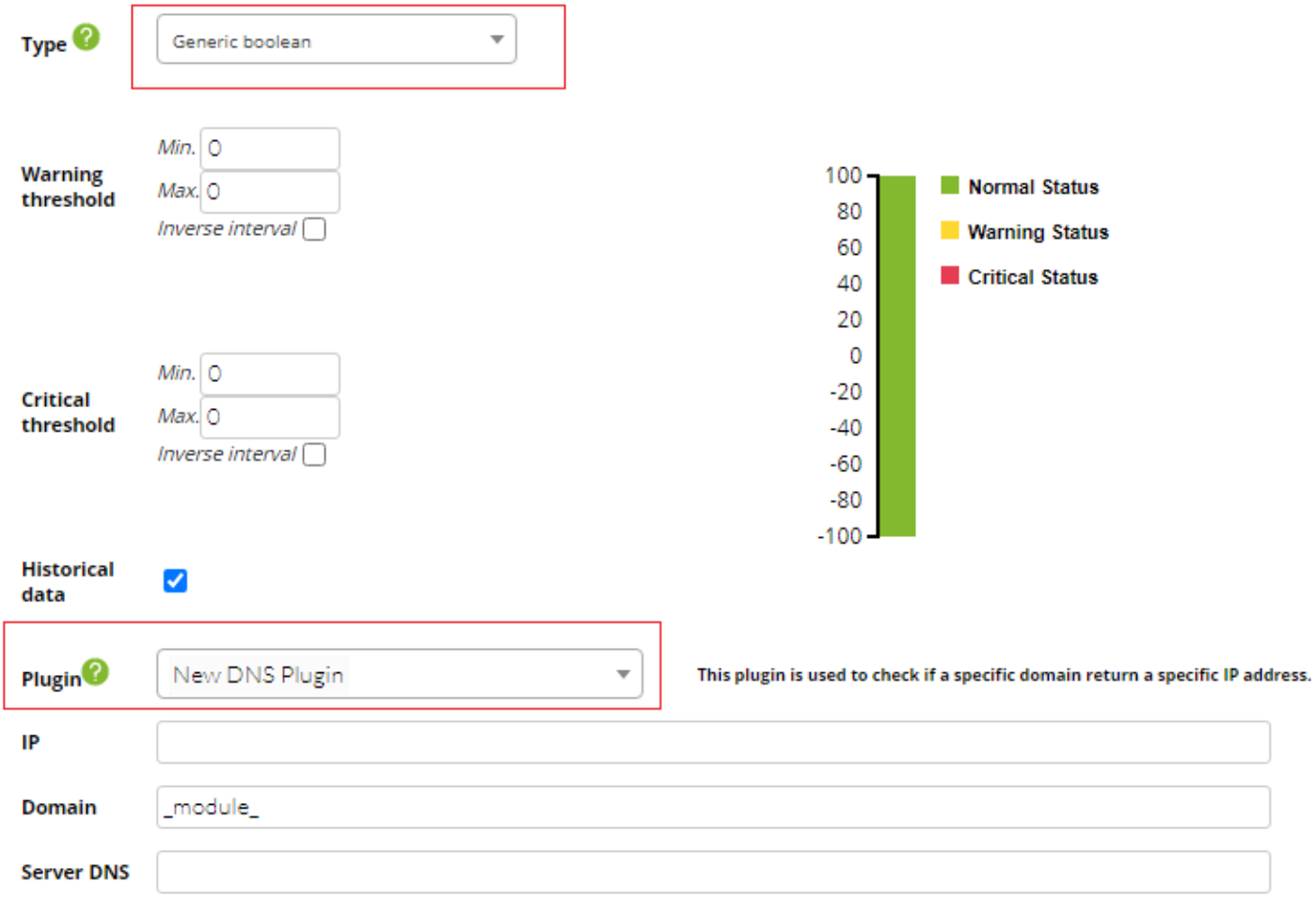
De la même manière que `_field1_`, `_field2_`, (...), `_fieldN_`, les macros fonctionnent également, à ceci près qu'elles contiennent des valeurs spéciales à la fois pour les modules et pour les agents qui contiennent ces modules.

Reprenons l'exemple de la section précédente où les valeurs par défaut du `_fieldN_` ont été laissées vides. Modifiez et pour la valeur par défaut de `_field2_` placez la macro `_module_`.

Si un module ou un composant utilise ce serveur *plugin*, une icône de verrouillage apparaîtra et ne pourra pas être mise à jour.

Cette macro `_module_` renvoie le nom du Module utilisé par le *plugin* et sera placé à l'utilisateur ou à la politique lors de la création du Module. Pour le vérifier, il faut créer un nouvel Agent appelé DNS verify et ajouter un nouveau Module à l'aide de l'option Create a new plugin server module.

Une fois que vous entrez dans le formulaire d'édition du nouveau module, dans Plugin sélectionnez dans la liste New DNS Plugin et la macro `_module_` apparaîtra comme suit:



N'oubliez pas de définir le type de données à collecter sur Generic boolean et pour le nom du nouveau module, définissez simplement le domaine web pour lequel vous allez vérifier l'adresse IP correspondante par rapport à un serveur DNS spécifié. Le *token* `critical_on_error` du PFMS server est configuré par défaut pour que modules en état inconnu deviennent critiques. Sauvegardez et testez l'opération.

L'avantage de cette méthode est que vous aurez autant de modules que de domaines web à vérifier et qu'ils seront facilement identifiables dans les différents composants (tableaux de bord, rapports, et cetera) par leur nom.

Liste des macros

- `_agent_`: Alias de l'agent à utiliser dans la macro. Si aucun alias n'est attribué, le nom de l'agent sera utilisé.
- `_agentalias_`: Alias de l'agent à utiliser dans la macro.
- `_agentdescription_`: Description de l'agent à utiliser dans la macro.
- `_agentstatus_`: État actuel de l'agent lorsqu'il est utilisé dans la macro.
- `_agentgroup_`: Nom du groupe d'agents qui doit utiliser la macro.
- `_agentname_`: Nom de l'agent à utiliser dans la macro (voir aussi `_agent_`).
- `_address_`: Adresse de l'agent qui doit utiliser la macro.
- `_module_`: Nom du module dans lequel la macro doit être utilisée.
- `_modulegroup_`: Nom du groupe de modules dans lequel la macro doit être utilisée.
- `_moduledescription_`: Description du module d'utilisation de la macro.

- `_modulestatus_`: Statut du module à utiliser par la macro.
- `_moduletags_`: URLs associées aux *tags* des modules.
- `_id_module_`: Identifiant du module à utiliser par la macro.
- `_id_agente_`: Identifiant de l'agent qui utilise la macro, utile pour créer des URL permettant d'accéder à la Console du Pandora FMS.
- `_id_group`: Identifiant du groupe d'agents à utiliser par la macro.
- `_interval_`: Intervalle d'exécution du module pour l'utilisation de la macro.
- `_target_ip_`: Adresse IP de la cible du module à utiliser par la macro.
- `_target_port_`: Port de la cible du module pour l'utilisation de la macro.
- `_policy_`: Nom de la police à laquelle le module appartient (le cas échéant).
- `_plugin_parameters_`: Paramètres de l'extension du module à utiliser par la macro.
- `_email_tag_`: Boîtes aux lettres électroniques associées aux balises (*tags*) des modules.
- `_phone_tag_`: Téléphones associés aux *tags* des modules.
- `_name_tag_`: Nom des balises associées au module à utiliser par la macro.

Conditionné en PSPZ

Le plugin Zipfile (.pspz) du Pandora FMS server

Il existe un moyen d'enregistrer les *plugins* et les modules qui utilisent le nouveau *plugin* (comme la bibliothèque de modules qui dépend du *plugin*). Il s'agit essentiellement d'une extension d'administration permettant de télécharger un fichier dans le répertoire `.pspz`, décrit en détail dans les sections suivantes. Le système lit le fichier, décompresse et installe les binaires et/ou les *scripts* dans le système. En outre, il enregistre le *plugin* et crée tous les modules définis dans le `.pspz` dans la bibliothèque de modules de Pandora FMS (composants réseau).

Cette section décrit comment créer un `.pspz.`

Package File

Un `.pspz` est un fichier compressé au format zip avec deux lignes:

plugin_definition.ini: contient la spécification du *plugin* et du module. Il doit porter exactement ce nom (il est *case sensitive*, sensible à la casse).

< script_file >: est le *plugin script binary* lui-même. Il peut porter n'importe quel nom valide. Un exemple de fichier `.pspz` (lui-même compressé en `.zip` pour inclure sa documentation) peut être téléchargé à ce lien.

Structure de la plugin_definition.ini

En-tête/Définition

Il s'agit d'un fichier INI classique avec des sections optionnelles. La première section, qui est la plus importante, est une section à nom fixe appelée `plugin_definition`. Voici un exemple:

```
[plugin_definition]
name = Remote SSH exec
filename = ssh_pandoraplugin.sh
description = This plugin execute remotely any command provided
timeout = 20
execution_command =
execution_postcommand =
ip_opt = -h
user_opt = -u
port_opt =
pass_opt =
plugin_type = 0
total_modules_provided = 1
```

`filename`: Il doit avoir le même nom que le script inclus dans le fichier `.pspz`, nommé `.pspz`, nommé `.pspz` avant comme `< script_file >`. Dans cet exemple, il s'agit d'un script *shell* (format `.sh`) nommé `ssh_pandoraplugin.sh`.

`_opt`: Voici les options d'enregistrement *plugin*, présentées dans le formulaire d'enregistrement “manuel” du plugin dans la Console de Pandora FMS.

`plugin_type`: `0` pour un *plugin* standard de Pandora FMS, et `1` pour un *plugin* de type Nagios.

`total_modules_provided`: Spécifie combien de modules sont définis dans les sections suivantes du fichier `.ini`. Vous devez en définir au moins un (à utiliser dans au moins un exemple).

`execution_command`: S'il est utilisé, il doit être placé devant *script*. Il peut s'agir d'un interpréteur, tel que `java -jar`. Ainsi, le *plugin* sera appelé à l'exécution, à partir du serveur de plugin Pandora FMS, avec le code suivant:

```
java -jar < plugin_path >/< plugin_filename >
```

`execution_postcommand`: S'il est utilisé, il définit les paramètres supplémentaires transmis à *plugin* après le `< plugin_filename >`, qui est invisible pour l'utilisateur.

Définition du module / Composants du réseau

Vous devez définir ici le même nombre de modules que celui défini dans `total_modules_provided` dans la section précédente.

Si vous avez par exemple quatre modules, les noms des sections doivent être les suivants: `module1`, `module2`, `module3` y `module4`.

Il s'agit d'un exemple de définition de module:

```
[module1]
name = Load Average 1Min
description = Get load average from command uptime
id_group = 12
type = 1
max = 0
min = 0
module_interval = 300
id_module_group = 4
id_modulo = 4
plugin_user = root
plugin_pass =
plugin_parameter = "uptime | awk '{ print $10 }' | tr -d ','"
max_timeout = 20
history_data = 1
min_warning = 2
min_critical = 5
str_warning = "danger"
min_critical = "alert"
min_ff_event = 0
tcp_port = 0
critical_inverse = 0
warning_inverse = 0
critical_instructions = "Call the department head"
warning_instructions = "Call the server manager to reduce the load."
unknown_instructions = "Verify that the Pandora FMS agent is running"
```

Quelques éléments à prendre en compte:

- Tous les champs *must* sont définis. Si vous n'avez pas de données, laissez le champ vide; voir le champ `plugin_pass` dans l'exemple ci-dessus.
- Utilisez des guillemets doubles par paires **"…"** pour définir des valeurs contenant des caractères spéciaux ou des espaces, comme le champ `plugin_parameter` dans l'exemple ci-dessus. Les fichiers INI contenant des caractères tels que `' "/ -_ ( ) [ ]` et autres, *de/must* ont des guillemets doubles. Essayez d'éviter d'utiliser le caractère `"` pour les données. Si vous devez l'utiliser, *escape* avec la combinaison `\"`.
- Si vous avez des doutes sur l'utilité ou la signification de ces champs, vous pouvez consulter `tnetwork_component` dans la base de données du Pandora FMS car elle contient pratiquement les

mêmes champs. Lorsqu'un nouveau composant réseau est créé, il est stocké dans cette base de données. Essayez de créer un composant de réseau qui utilise votre *plugin* et analysez l'enregistrement d'entrée dans cette table pour comprendre toutes les valeurs.

- `id_module`: Il doit toujours être égal à 4 (ce qui signifie qu'il s'agit d'un module *plugin*).
- `type`: Définit le type de module : generic_data (1), generic_proc (2), generic_data_string (3) ou generic_data_inc (4) comme défini dans `ttipo_modulo`.
- `id_group`: Il s'agit de la PK (*primary key*) de la table des groupes contenant les définitions des groupes. Le groupe 1 correspond à “tous les groupes” (All), et agit comme un groupe spécial.
- `id_module_group`: Il provient de la table `tmodule_group`. Il s'agit d'une association de modules par fonctionnalité, purement descriptive. Vous pouvez utiliser 1 pour le module General module group.

Versión 2

Depuis Pandora FMS v5.1.SP1, les *plugins* du serveur utilisent des macros.

Ces plugins seront différenciés par l'extension du fichier pspz2. S'il n'a pas cette extension, un PSPZ de type 1 sera supposé (pas de macros dynamiques dans l'extension distante *plugin*).

Le fichier `plugin_definition.ini` a également été modifié. Les champs suivants ont été ajoutésGitLab PFMS 14504 . :

Dans la section `plugin_definition`:

- `total_macros_provided` qui définit le nombre de macros dynamiques que possède le *plugin*.

Dans la section `module<N>`:

- `macro_<N>_value` qui définit la valeur de ce module à l'aide de cette macro dynamique; si elle n'existe pas, c'est la valeur par défaut qui est retenue.

Et vous devez créer une section pour chaque macro dynamique, par exemple:

```
[macro_<N>]
hide = 0
description = <your_description>
help = <text_help>
value = <your_value>
```

- Les macros doivent être appelées dans la section `execution_postcommand` pour effectuer la substitution (voir exemple).
- La version précédente est toujours prise en charge. Si le paramètre version n'est pas défini, la version est supposée être 1.

Exemple

Définition d'un plugin v2 (`.pspz2`) à des fins didactiques:

```
[plugin_definition]
name = PacketLoss
filename = packet_loss.sh
description = "Measure packet loss in the network in %"
timeout = 20
ip_opt =
execution_command =
execution_postcommand =
parameters = _field1_ _field2_
user_opt =
port_opt =
pass_opt =
plugin_type = 0
total_modules_provided = 1
total_macros_provided = 2

[macro_1]
hide = 0
description = Timeout
help = Timeout in seconds
value = 5

[macro_2]
hide = 0
description = Target IP
help = IP adddress
value = 127.0.0.1

[module1]
name = Packet loss
description = "Measure target packet loss in % "
id_group = 10
type = 1
max = 0
min = 0
module_interval = 300
id_module_group = 2
id_modulo = 4
max_timeout = 20
history_data = 1
min_warning = 30
min_critical = 40
min_ff_event = 0
tcp_port = 0
macro_1_value = 5
macro_2_value = localhost
unit = %
```

Retour à l'index de la documentation du Pandora FMS