.Disco development

.Disco development

Forfaits ''.disco''

Discovery vous permet de charger à la fois des plugins Pandora FMS officiels et personnalisés.

Pour charger des plugins personnalisés, il est nécessaire de générer un package `.disco` avec tout le nécessaire pour que la console et le serveur Pandora FMS soient capables de:

- Afficher l'interface de configuration pour les nouvelles tâches de découverte.
- Exécutez les tâches de découverte configurées dans l'environnement.

Un package `.disco` est un fichier zip avec l'extension .disco qui contient au moins un fichier appelé discovery_definition.ini.

Facultativement, un package `.disco` peut contenir un fichier appelé logo.png qui sera le logo du plugin dans la console Pandora FMS. Si vous n'ajoutez pas de fichier de logo, la console utilise automatiquement un logo par défaut.

Enfin, typiquement, un fichier `.disco` contiendra tous les scripts, exécutables et bibliothèques nécessaires au serveur et à la console. Bien que ce ne soit pas obligatoire, c'est courant et recommandé, pour éviter des exigences d'installation supplémentaires sur les systèmes sur lesquels vous souhaitez configurer et exécuter les tâches qui utilisent le plugin.

Ainsi et en résumé, un fichier `.disco` contiendra:

- `discovery_definition.ini`.
- `logo.png` (facultatif).
- Scripts, exécutables et bibliothèques.

Fichier ''discovery_definition.ini''

Le fichier discovery_definition.ini est le plus important au sein d'un fichier `.disco`, puisque c'est celui qui contient toute la définition du plugin.

Il contient à la fois les paramètres qui seront affichés dans la console pour être renseignés dans un formulaire de définition de tâche, et les exécutions que devra effectuer le serveur Pandora FMS pour les tâches du plugin.

Afin de faciliter sa définition et son traitement dans la console, le format utilisé dans un fichier `discovery_definition.ini` est le format INI propre à PHP.

Un fichier `discovery_definition.ini` est essentiellement composé de 3 blocs:

- `discovery_extension_definition`.
- `config_steps` (facultatif).
- `tempfile_confs` (facultatif).

Concepts communs

Au cours des explications suivantes, il sera courant d'utiliser certains termes. Ces termes sont expliqués dans cette section.

Une macro est une clé de texte unique utilisée pour stocker des informations (valeurs, chemins d'accès aux fichiers, ...). Lorsqu'une macro est utilisée (par exemple lors d'une exécution), vous souhaitez plutôt utiliser les informations qu'elle contient.

Une macro valide est une clé de texte qui commence et se termine par des traits de soulignement « _ » et entre les deux ne peut contenir que des lettres (A-Z et a-z) et des chiffres (0-9).

Lorsque nous parlerons de STRING, nous ferons référence à des chaînes de texte alphanumériques.

Lorsque nous parlerons de NOMBRE, nous ferons exclusivement référence aux chiffres.

Lorsque nous parlons de BOOL, nous ferons référence aux valeurs true / false, 1 / 0, yes / no .

Lorsque nous parlons de VALUE, nous ferons référence à n'importe quelle valeur (STRING, NUMBER ou BOOL). Il sera normalement utilisé dans des listes séparées par des virgules pour représenter des valeurs du même type.

Lorsque les lettres majuscules N ou M sont utilisées, nous faisons référence à un entier positif. Il sera généralement utilisé conjointement avec d'autres éléments tels que VALUE pour indiquer qu'il peut être répété plusieurs fois (par exemple, VALUE_N).

Macros du serveur

Une grande partie de la configuration d'exécution du serveur est basée sur l'utilisation de macros. Généralement, ceux-ci seront définis par l'utilisateur, mais il y en a quelques-uns spécifiques au serveur Pandora FMS qui peuvent être utiles:

- `__taskMD5__` → MD5 unique de la tâche, généré à partir de l'ID de la tâche et du nom court de l'application. Il est utile pour générer des éléments uniques lors des exécutions (comme des fichiers).
- `__taskInterval__` → Intervalle d'exécution de la tâche (secondes).
- `__taskGroup__` → Groupe de tâches (Texte).
- `__taskGroupID__` → Groupe de tâches (ID).
- `__temp__` → Répertoire du serveur temporaire configuré dans `pandora_server.conf` (temporaire).

- `__incomingDir__` → Répertoire entrant du serveur Pandora configuré dans `pandora_server.conf` (incomingdir).
- `__consoleAPIURL__` → URL API configurée dans `pandora_server.conf` (console_api_url).
- `__consoleAPIPass__` → Pass API configuré dans `pandora_server.conf` (console_api_pass).
- `__consoleUser__` → Utilisateur de console configuré dans `pandora_server.conf` (console_user).
- `__consolePass__` → Mot de passe de la console configuré dans `pandora_server.conf` (console_pass).

Ces macros ne sont en aucun cas utilisées par la console Pandora FMS.

bloc découverte_extension_definition

Ce bloc définit la configuration principale du plugin et possède les paramètres suivants:

- nom court:
  - Obligatoire.
  - Définit le nom court du plugin.
  - Le nom court doit être unique.
  - Noms courts commençant par le préfixe « pandorafms ». Ils sont utilisés par les plugins officiels Pandora FMS.
  - Les noms courts ne peuvent contenir que des lettres (“A-Z” et “a-z”), des chiffres (`0-9`), des points (`.`), des traits d'union (-) et les traits de soulignement (_).
- section:
  - Obligatoire.
  - Définit la sectionion de la console où le plugin sera affiché.
  - Peut être app, cloud ou custom.
- nom:
  - Obligatoire.
  - Définit le nom du plugin qui sera affiché dans la console.
- version:
  - Obligatoire.
  - Définit la version du plugin qui sera affichée dans la console.
  - Il est recommandé de changer de version si des modifications sont apportées à un plugin, aussi mineures soient-elles.
- description:
  - Facultatif.
  - Définit la description du plugin qui sera affichée dans la console.
- fichier_exécution:
  - Facultatif.
  - Indique les chemins relatifs (au sein du fichier `.disco`) vers les scripts et exécutables utilisés par le plugin.
  - Tous les fichiers indiqués recevront une autorisation d'exécution.
  - Ce paramètre est un tableau dont les clés seront des macros valides qui serviront à indiquer l'utilisation des fichiers du plugin.
  - Comme il s'agit d'un array, ce paramètre peut être indiqué plusieurs fois à l'aide de touches différentes.
  - Par exemple:

```
execution_file[_exec1_] = "script1.py"
execution_file[_exec2_] = "other/script2.py"
```

- exécutable:
  - Obligatoire.
  - Définit le format des exécutions que le serveur Pandora FMS doit effectuer pour les tâches du plugin.
  - Ce paramètre est un tableau dont les clés seront positionnelles, c'est-à-dire qu'elles pourront être indiquées par des chiffres ou non.
  - Comme il s'agit d'un array, ce paramètre peut être indiqué plusieurs fois à l'aide de touches différentes.
  - Il sera courant d'utiliser des macros dans les exécutions définies à la fois pour indiquer les chemins d'accès aux scripts ou exécutables utilisés et pour indiquer les paramètres desdits scripts ou exécutables.
  - Le serveur Pandora FMS lancera chacune des exécutions définies dans l'ordre et stockera le résultat de chacune d'entre elles pour produire la sortie du plugin dans la tâche et obtenir son statut.
  - Par exemple:

```
exec[] = "'_exec1_' -p '_param1_'"
exec[] = "'_exec2_' -p '_param2_'"
```

- passencrypt_script:
  - Facultatif.
  - Définit le chemin relatif (au sein du fichier `.disco`) vers un script utilisé pour chiffrer les champs de type mot de passe depuis la console.
  - Le fichier indiqué recevra une autorisation d'exécution.
- passencrypt_exec:
  - Facultatif.
  - Définit le format d'exécution du script pour crypter les mots de passe depuis la console.
  - Le résultat attendu de ladite exécution sera un texte qui correspond au mot de passe crypté.
  - Prend uniquement en charge l'utilisation des macros _passencrypt_script_ et _password_ dans sa définition.
  - La macro _passencrypt_script_ est remplacée par le chemin d'accès au fichier défini dans passencrypt_script.
  - La macro _password_ est remplacée par la valeur du champ type de mot de passe qui doit être chiffré à tout moment.
  - Par exemple:

```
passencrypt_exec = "'_passencrypt_script_' --encrypt '_password_'"
```

- passdecrypt_script:
  - Facultatif.
  - Définit le chemin relatif (au sein du fichier `.disco`) vers un script utilisé pour décrypter les champs de type mot de passe depuis la console.
  - Le fichier indiqué recevra une autorisation d'exécution.
- passdecrypt_exec:
  - Facultatif.
  - Définit le format d'exécution du script pour décrypter les mots de passe depuis la console.
  - Le résultat attendu de ladite exécution sera un texte qui correspond au mot de passe déchiffré.
  - Prend uniquement en charge l'utilisation des macros _passdecrypt_script_ et _password_ dans sa définition.
  - La macro _passdecrypt_script_ est remplacée par le chemin d'accès au fichier défini dans

passdecrypt_script.
    - La macro _password_ est remplacée par la valeur du champ type de mot de passe qui doit être déchiffrée à tout moment.
    - Par exemple:

```
passencrypt_exec = "'_passdecrypt_script_' --decrypt '_password_'"
```

- valeur par défaut:
    - Facultatif.
    - Définit les valeurs par défaut des différentes macros utilisées lors des exécutions, c'est-à-dire les valeurs par défaut des données stockées pour chaque tâche *Discovery* créée pour ce plugin.
    - Ce paramètre est un tableau dont les clés seront des macros valides.
    - Comme il s'agit d'un array, ce paramètre peut être indiqué plusieurs fois à l'aide de touches différentes.
    - Selon le type de champ utilisé dans les formulaires console (voir ci-dessous) les valeurs par défaut peuvent être:
        1. chaîne: CHAÎNE .
        2. numéro: NUMÉRO .
        3. mot de passe: STRING .
        4. zone de texte: CHAÎNE .
        5. case à cocher: BOOL .
        6. sélectionnez: VALUE_N .
        7. multisélection: [VALUE_1,VALUE_N] .
        8. arbre: [VALUE_1,VALUE_N] .
    - Il est recommandé d'inclure une valeur_par_défaut pour chacune des macros pouvant figurer dans les formulaires de configuration des tâches.
    - Par exemple:

```
default_value[_param1_] = "get_main_info"
default_value[_param2_] = ""
default_value[_param3_] = true
default_value[_param4_] = 0
default_value[_param5_] = "[A,B,C]"
default_value[_param6_] = "[]"
```

L'exemple suivant peut servir de base à ce bloc, puisqu'il comprend tous les paramètres expliqués avec des commentaires pour chaque cas:

```
[discovery_extension_definition]

; Mandatory
; Defines discovery application short name
; Short name must be unique
; Short names with "pandorafms." preffix are used by Pandora FMS for official
applications

short_name = DISCOVERY_APPLICATION_UNIQUE_NAME

; Mandatory
; Defines the section where application will be shown in console
; Possible values:
```

```
;    app
;    cloud
;    custom

section = app

; Mandatory
; Defines discovery application name, shown in console

name = DISCOVERY_APPLICATION_NAME

; Mandatory
; Defines discovery application version, shown in console

version = VERSION

; Optional
; Defines discovery application description, shown in console

description = DESCRIPTION

; Optional
; Defines execution files inside .disco
; Several execution files can be defined

execution_file[_EXEC_MACRO_N_] = SCRIPT.pl

; Mandatory
; Defines execution for discovery server
; Several executions can be defined
; At least 1 is required for server execution

exec[] = SERVER_EXECUTION

; Optional
; Defines password encrypt script

passencrypt_script = PASS_ENCRYPT_SCRIPT.pl

; Optional
; Defines password encrypt script execution format
; _passencrypt_script_ is replaced with the passencrypt_script file path
; _password_ is replaced with the string (password) to encrypt

passencrypt_exec = EXECUTION

; Optional
; Defines password decrypt script

passdecrypt_script = PASS_DECRYPT_SCRIPT.pl

; Optional
```

```
; Defines password encrypt script execution format
; _passdecrypt_script_ is replaced with the passdecrypt_script file path
; _password_ is replaced with the string (password) to encrypt

passdecrypt_exec = EXECUTION

; Optional
; Defines the default values for the fields when a new task is created
; By default all values are empty or not selected
; Several default values can be defined
; Possible values depending on field type:
;    string            - STRING
;    number            - NUMBER
;    password        - STRING
;    textarea        - STRING
;    checkbox        - BOOL
;    select            - VALUE_N
;    multiselect        - [VALUE_1,VALUE_N]
;    tree            - [VALUE_1,VALUE_N]

default_value[_FIELD_MACRO_N_] = DEFAULT_VALUE
```

bloc config_steps

Ce bloc définit les étapes de configuration des tâches du plugin et possède les paramètres suivants:

Ces règles s'appliquent à tous les paramètres de ce bloc:

- Les paramètres sont des tableaux dont les clés seront positionnelles, c'est-à-dire qu'elles peuvent être indiquées par des chiffres ou non. Il est recommandé d'indiquer les clés.
- S'agissant de tableaux, les paramètres peuvent être indiqués plusieurs fois à l'aide de touches différentes.
- Tous les paramètres qui partagent une clé font référence au même élément. C'est-à-dire qu'ils attribuent différentes valeurs (selon le paramètre) au même élément.

- nom:
  - Obligatoire.
  - Définit les noms des différentes étapes de configuration dans les tâches.
- script_data_fields:
  - Obligatoire si un custom_fields n'est pas défini pour la même clé.
  - Facultatif si un custom_fields est défini pour la même clé.
  - Définit le format des exécutions que la console Pandora FMS doit effectuer pour les formulaires générés dynamiquement. Par exemple, si vous souhaitez surveiller un environnement de virtualisation et que vous souhaitez obtenir une liste de machines virtuelles pour sélectionner

celles que vous souhaitez surveiller.
- Les clés utilisées doivent exister dans le tableau name de ce bloc.
- Il sera courant d'utiliser des macros dans les exécutions définies à la fois pour indiquer les chemins d'accès aux scripts ou exécutables utilisés et pour indiquer les paramètres desdits scripts ou exécutables.
- Le résultat des exécutions indiquées ici doit être un JSON au format suivant. Les clés JSON correspondent aux paramètres du bloc CUSTOM_FIELDS (voir ci-dessous):

```
[
    {
        "macro": "_FIELD_MACRO_N_",
        "mandatory_field": "BOOL",
        "name": "FIELD_NAME",
        "tip": "FIELD_TIP",
        "type": "FIELD_TYPE",
        "placeholder": "PLACEHOLDER",
        "show_on_true": "_FIELD_MACRO_N_",
        "encrypt_on_true": "_FIELD_MACRO_N_",
        "select_data": {
            "VALUE_1": "TEXT_1",
            "VALUE_N": "TEXT_N"
        },
        "tree_data": [
            {
                "name": "TEXT_1",
                "selectable": "BOOL_1",
                "macro": "_FIELD_MACRO_N_",
                "value": "VALUE_1",
                "children": [
                    {
                        "name": "TEXT_1_1",
                        "selectable": "BOOL_1_1",
                        "macro": "_FIELD_MACRO_N_",
                        "value": "VALUE_1_1",
                        "children": []
                    },
                    {
                        "name": "TEXT_1_N",
                        "selectable": "BOOL_1_N",
                        "macro": "_FIELD_MACRO_N_",
                        "value": "VALUE_1_N",
                        "children": []
                    }
                ]
            },
            {
                "name": "TEXT_N",
                "selectable": "BOOL_N",
                "macro": "_FIELD_MACRO_N_",
                "value": "VALUE_N",
                "children": []
            }
```

```
                ]
            }
        ]
```

- Les champs personnalisés:
  - Obligatoire si un script_data_fields n'est pas défini pour la même clé.
  - Facultatif si un script_data_fields est défini pour la même clé.
  - Définit les blocs de configuration du même fichier INI à partir duquel les champs du formulaire seront obtenus lors de la création de tâches dans la console Pandora FMS.
  - Les clés utilisées doivent exister dans le tableau name de ce bloc.
- fields_columns:
  - Facultatif.
  - Définit le nombre de colonnes dans lesquelles les champs du formulaire de tâche seront répartis à chaque étape de configuration.
  - Les clés utilisées doivent exister dans le tableau name de ce bloc.
  - Peut se voir attribuer la valeur 1 ou 2.
  - Sa valeur par défaut est 2.

Par exemple, ce serait un moyen de définir plusieurs étapes de configuration:

```
[config_steps]

name[1] = First step
script_data_fields[1] = "'_exec2_' -p '_param1_' --get_fields"

name[2] = Mid step
custom_fields[2] = custom_fields_1

name[3] = Last step
script_data_fields[3] = "'_exec2_' -p '_param2_' --get_fields"
custom_fields[3] = custom_fields_2
```

Si script_data_fields et custom_fields sont indiqués pour la même étape de configuration, le formulaire affichera d'abord les champs obtenus par script_data_fields puis ceux définis dans custom_fields.

L'exemple suivant peut servir de base à ce bloc, puisqu'il comprend tous les paramètres expliqués avec des commentaires pour chaque cas:

```
[config_steps]

; Following parameters can be setup for each configuration step
; Several steps can be defined using a different key (N)

; Mandatory
; Defines configuration step name

name[N] = STEP_NAME

; Mandatory if not custom_fields defined
; Defines configuration fields retrieved to console by a command execution
```

```
; execution_file scripts can be used to retrieve data
; Command execution output must be JSON as follows
;    [
;        {
;            "macro": "_FIELD_MACRO_N_",
;            "mandatory_field": "BOOL",
;            "name": "FIELD_NAME",
;            "tip": "FIELD_TIP",
;            "type": "FIELD_TYPE",
;            "placeholder": "PLACEHOLDER",
;            "show_on_true": "_FIELD_MACRO_N_",
;            "encrypt_on_true": "_FIELD_MACRO_N_",
;            "select_data": {
;                "VALUE_1": "TEXT_1",
;                "VALUE_N": "TEXT_N"
;            },
;            "tree_data": [
;                {
;                    "name": "TEXT_1",
;                    "selectable": "BOOL_1",
;                    "macro": "_FIELD_MACRO_N_",
;                    "value": "VALUE_1",
;                    "children": [
;                        {
;                            "name": "TEXT_1_1",
;                            "selectable": "BOOL_1_1",
;                            "macro": "_FIELD_MACRO_N_",
;                            "value": "VALUE_1_1",
;                            "children": []
;                        },
;                        {
;                            "name": "TEXT_1_N",
;                            "selectable": "BOOL_1_N",
;                            "macro": "_FIELD_MACRO_N_",
;                            "value": "VALUE_1_N",
;                            "children": []
;                        }
;                    ]
;                },
;                {
;                    "name": "TEXT_N",
;                    "selectable": "BOOL_N",
;                    "macro": "_FIELD_MACRO_N_",
;                    "value": "VALUE_N",
;                    "children": []
;                }
;            ]
;        }
;    ]

script_data_fields[N] = CONSOLE_EXECUTION_FIELDS
```

```
; Mandatory if not script_data_fields defined
; Defines custom configuration fields

custom_fields[N] = CUSTOM_FIELDS_N

; Optional
; Defines the number of fields columns for the configuration steps (1 or 2)

fields_columns[N] = M
```

bloc tempfile_confs

Ce bloc définit le format des fichiers de configuration temporaires utilisés par le plugin et possède les paramètres suivants:

- déposer:
  - Obligatoire.
  - Définit le contenu des fichiers de configuration temporaires qui peuvent être utilisés lors des exécutions du serveur et de la console Pandora FMS.
  - Ce paramètre est un tableau dont les clés seront des macros valides qui serviront à indiquer l'utilisation de fichiers temporaires dans le plugin.
  - Comme il s'agit d'un array, ce paramètre peut être indiqué plusieurs fois à l'aide de touches différentes.
  - Lorsqu'une des clés de ce tableau est indiquée lors d'une exécution, un fichier temporaire est généré dont le contenu est celui indiqué dans ce tableau et la valeur de la macro est remplacée par l'emplacement dudit tableau. fichier temporaire. Une fois l'exécution terminée, le fichier est supprimé.
  - Dans le contenu des fichiers de configuration temporaires, il est possible d'indiquer d'autres macros, afin qu'elles puissent être remplacées par la valeur correspondante.
  - Par exemple:

```
        file[_tempConf_] = "server _param1_
        user _param2_
        password _param3_
        log __temp__/__taskMD5__.log"
```

El siguiente ejemplo puede usarse como base para este bloque, ya que incluye todos los parámetros explicados con comentarios para cada caso:

```
[tempfile_confs]

; Mandatory
; Defines the content for the temporary file
; File will be used where temporary file macro is specified during executions
; File content replaces fields macros with their values

file[_TEMP_FILE_MACRO_N_] = _FIELD_MACRO_N_,_FIELD_MACRO_M_
```

L'exemple suivant peut servir de base à ce bloc, puisqu'il comprend tous les paramètres expliqués

avec des commentaires pour chaque cas:

```
[fichiertemp_confs]

; Obligatoire
; Vous définissez le contenu du fichier temporaire
; Le fichier sera utilisé là où la macro de fichier temporaire est spécifiée
lors des exécutions
; Le contenu du fichier remplace les macros de champs par leurs valeurs

fichier[_TEMP_FILE_MACRO_N_] = _FIELD_MACRO_N_,_FIELD_MACRO_M_
```

Blocs CUSTOM_FIELDS

Ce type de bloc peut avoir n'importe quel nom, à condition qu'il ait été attribué comme valeur dans le bloc config_steps pour son paramètre custom_fields. Ils définissent les champs des formulaires dans les étapes de configuration du plugin et ont les paramètres suivants:

Ces règles s'appliquent pour tous les paramètres de ce bloc:

- Les paramètres sont des tableaux dont les clés seront positionnelles, c'est-à-dire qu'elles peuvent être indiquées par des chiffres ou non. Il est recommandé d'indiquer les clés.
- S'agissant de tableaux, les paramètres peuvent être indiqués plusieurs fois à l'aide de touches différentes.
- Tous les paramètres qui partagent une clé font référence au même élément. C'est-à-dire qu'ils attribuent différentes valeurs (selon le paramètre) au même élément.

- macro:
  - Obligatoire.
  - Définit les macros dans lesquelles les valeurs de configuration des tâches seront stockées.
- champ_obligatoire:
  - Facultatif.
  - Définit à l'aide de valeurs de type BOOL si le champ de configuration du formulaire doit avoir une valeur ou non.
  - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
  - Par défaut, tous les champs sont obligatoires.
  - Même s'ils sont obligatoires, les champs de type multiselect et tree permettent de ne pas sélectionner des éléments.
- nom:
  - Obligatoire.
  - Définit les noms des champs de configuration du formulaire.
  - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
- conseil:
  - Facultatif.
  - Définit l'aide des champs de configuration du formulaire.
  - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
- taper:
  - Obligatoire.

    - Définit les types de champs de configuration du formulaire.
    - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
    - Ses valeurs possibles sont:
        - string: Le champ sera une zone de texte. Acceptera les valeurs de type STRING.
        - numéro: Le champ sera une zone de saisie de numéro. Acceptera les valeurs de type NUMBER.
        - mot de passe: Le champ sera une zone de texte dont la valeur sera masquée. Acceptera les valeurs de type STRING.
        - textarea: Le champ sera une large zone de texte. Acceptera les valeurs de type STRING.
        - case à cocher: Le champ acceptera les valeurs de type BOOL
        - select: Le champ sera une liste déroulante dans laquelle sélectionner une seule valeur. Acceptera les valeurs de type VALUE.
        - multiselect: Le champ sera une boîte de sélection multiple parmi plusieurs options. Acceptera les valeurs de type VALUE.
        - arbre: Le champ sera un arbre de différents niveaux dont les éléments pourront ou non être sélectionnés pour envoyer leurs valeurs. Acceptera les valeurs de type VALUE.
- espace réservé:
    - Facultatif si le type associé est string ou textarea.
    - Définit les textes qui seront affichés dans les zones de texte à titre d'exemple.
    - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
- show_on_true:
    - Facultatif.
    - Définit macros d'autres champs de type checkbox du formulaire comme valeurs, de sorte que lorsque ces macros ont une valeur true dans le formulaire, le champ souhaité soit affiché.
    - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
- encrypt_on_true:
    - Facultatif si le type associé est mot de passe.
    - Définit les macros d'autres champs de type checkbox du formulaire comme valeurs, de sorte que lorsque ces macros ont une valeur true dans le formulaire, le champ souhaité soit crypté lors de l'enregistrement de la configuration.
    - Le cryptage/déchiffrement sera effectué en fonction de la configuration dans le bloc discovery_extension_definition.
    - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
- select_data:
    - Obligatoire si le type associé est select ou multiselect.
    - Définit l'origine des valeurs pour les listes déroulantes.
    - Les sélecteurs peuvent être générés avec des données Pandora FMS ou personnalisés. Pour ce faire, il prend en charge les valeurs suivantes:
        - Le nom des autres blocs de configuration du même fichier INI à partir desquels seront obtenus des éléments de type select ou multiselect.
            - agent_groups: utilise les groupes d'agents comme données.
            - agents: utilisez des agents comme données.
            - module_groups: utilisez les groupes de modules comme données.
            - modules: utilisez les modules comme données.
            - module_types: utilisez les types de modules comme données.
            - tags: utilisez les balises comme données.
            - statut: utiliser les états comme données.
            - alert_templates: utilisez des modèles d'alerte comme données.
            - alert_actions: utilisez les actions d'alerte comme données.
            - intervalle: utilisez le sélecteur de temps comme données.
            - credentials.custom: utilise le sélecteur d'informations d'identification personnalisées comme données.

                - credentials.aws: utilise le sélecteur d'informations d'identification AWS comme données.
                - credentials.azure: utilise le sélecteur d'informations d'identification Azure comme données.
                - credentials.gcp: utilise le sélecteur d'informations d'identification Google Cloud comme données.
                - credentials.sap: utilise le sélecteur d'informations d'identification SAP comme données.
                - credentials.snmp: utilise le sélecteur d'informations d'identification SNMP comme données.
                - credentials.wmi: utilise le sélecteur d'informations d'identification WMI comme données.
                - os: utiliser le système d'exploitation comme données.
        - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
        - Les valeurs de la macro si le type associé est multiselect seront un JSON de type array avec les différentes valeurs sélectionnées. Par exemple:

```
[1,5,12,23]
```

- tree_data:
    - Obligatoire si le type associé est tree.
    - Définit les blocs de configuration du même fichier INI à partir duquel les éléments de type tree seront obtenus.
    - Les clés utilisées doivent exister dans le tableau macro de ce bloc.
    - Les valeurs de la macro seront un JSON de type array avec les différentes valeurs sélectionnées. Par exemple:

```
["elementA","elementF","elementK"]
```

Par exemple, ce serait un moyen de définir plusieurs champs de configuration:

```
[custom_fields_1]

macro[1] = _param1_
name[1] = User
type[1] = string

macro[2] = _param2_
name[2] = Password
type[2] = password
encrypt_on_true[2] = _param3_

macro[3] = _param3_
name[3] = Encrypt password
type[3] = checkbox

macro[4] = _param4_
name[4] = Max threads
type[4] = number
mandatory_field[4] = false

macro[5] = _param5_
```

```
name[5] = Agents group
tip[5] = Agents are generated in this group
type[5] = select
select_data[5] = agent_groups

macro[6] = _param6_
name[6] = Mode
type[6] = select
select_data[6] = custom_select_1

macro[7] = _param7_
name[7] = Add extra options
type[7] = checkbox

macro[8] = _param8_
name[8] = Extra elements
type[8] = tree
tree_data[8] = custom_tree_1
show_on_true[8] = _param7_

macro[9] = _param9_
name[9] = Extra options
type[9] = textarea
placeholder[9] = "Add extra options here"
show_on_true[9] = _param7_
```

L'exemple suivant peut servir de base à ce bloc, puisqu'il comprend tous les paramètres expliqués avec des commentaires pour chaque cas:

```
[CUSTOM_FIELDS_N]

; Mandatory
; Defines configuration field unique macro
; Macros can be used for script executions, using their value in place

macro[N] = _FIELD_MACRO_N_

; Optional
; Defines if configuration field is mandatory or not
; By default all configuration fields are mandatory

mandatory_field[N] = BOOL

; Mandatory
; Defines configuration field name to be displayed in console
; Macro will be used as name if this field is empty

name[N] = FIELD_NAME

; Optional
; Defines a tip for the field, to be displayed in console
```

```
tip[N] = FIELD_TIP

; Mandatory
; Defines the field type to be displayed in console
; Possible values:
;     string
;    number
;     password
;     textarea
;     checkbox
;     select
;     multiselect
;    tree

type[N] = FIELD_TYPE

; Optional if type is string or textarea
; Defines a placeholder for the field, to be displayed in console

placeholder[N] = PLACEHOLDER

; Optional
; Field is shown in console only if assigned field macro exists, is checkbox and
is true

show_on_true[N] = _FIELD_MACRO_N_

; Optional if type is password
; Field value is encrypted when stored into database if assigned field macro
exists, is checkbox and is true
; Password encrypt and decrypt depends on scripts uploaded to the discovery
application

encrypt_on_true[N] = _FIELD_MACRO_N_

; Mandatory if type is select or multiselect
; Defines select data to be displayed in console
; Possible values:
;     agent_groups        - Uses agent groups names
;     agents                - Uses agents names
;     module_groups        - Uses module groups
;    modules                - Uses modules names
;    module_types        - Uses module types names
;    tags                - Uses module tags
;    status                - Uses module status names
;    alert_templates        - Uses alert templates names
;    alert_actions        - Uses alert actions names
;    interval            - Uses time interval selector
;    credentials.custom    - Uses pandora custom credentials selector
;    credentials.aws        - Uses pandora AWS credentials selector
;    credentials.azure    - Uses pandora Microsoft Azure credentials selector
;    credentials.gcp        - Uses pandora Google Cloud Platform credentials
```

```
selector
;    credentials.sap        - Uses pandora SAP credentials selector
;    credentials.snmp    - Uses pandora SNMP credentials selector
;    credentials.wmi        - Uses pandora WMI credentials selector
;    os                    - Uses pandora OS names
;     CUSTOM_SELECT_N        - Uses custom selector values
; multiselect fields value is a comma separated list of selected values

select_data[N] = FIELD_DATA

; Mandatory if type is tree
; Defines tree data to be displayed in console
; tree fields value is a comma separated list of selected values

tree_data[N] = CUSTOM_TREE_DATA_N
```

Blocs CUSTOM_SELECT

Ce type de bloc peut porter n'importe quel nom, à condition qu'il ait été attribué comme valeur dans un bloc CUSTOM_FIELDS pour son paramètre select_data. Ils définissent les options d'un menu déroulant du formulaire de configuration du plugin et possèdent les paramètres suivants:

- option:
  - Obligatoire.
  - Définit les options d'un sélecteur personnalisé.
  - Ce paramètre est un tableau dont les clés sont les valeurs des options du sélecteur personnalisé, et ses valeurs sont les textes affichés pour chaque option dans le sélecteur personnalisé.
  - Comme il s'agit d'un array, ce paramètre peut être indiqué plusieurs fois à l'aide de touches différentes.

Par exemple:

```
[custom_select_1]

option[v1] = Valor 1
option[v2] = Valor 2
option[v3] = Valor 3
option[v4] = Valor 4
option[v5] = Valor 5
```

L'exemple suivant peut être utilisé comme base pour ce bloc, car il inclut tous les paramètres expliqués avec des commentaires pour chaque cas:

```
[CUSTOM_SELECT_N]

; Mandatory
; Defines the value-text pair for the select or multiselect
```

```
; Several value-text pairs can be defined

option[VALUE_N] = TEXT_N
```

Blocs CUSTOM_TREE_DATA

Ce type de bloc peut avoir n'importe quel nom, à condition qu'il ait été attribué comme valeur dans un bloc CUSTOM_FIELDS pour son paramètre tree_data ou dans un autre bloc CUSTOM_TREE_DATA pour son paramètre enfants. Ils définissent les éléments d'une arborescence de formulaire de configuration de plugin et ont les paramètres suivants:

Ces règles s'appliquent pour tous les paramètres de ce bloc:

- Les paramètres sont des tableaux dont les clés seront positionnelles, c'est-à-dire qu'elles peuvent être indiquées par des chiffres ou non. Il est recommandé d'indiquer les clés.
- S'agissant de tableaux, les paramètres peuvent être indiqués plusieurs fois à l'aide de touches différentes.
- Tous les paramètres qui partagent une clé font référence au même élément. C'est-à-dire qu'ils attribuent différentes valeurs (selon le paramètre) au même élément.

- nom:
  - Obligatoire.
  - Définit les noms des différents éléments à ce niveau de l'arborescence.
- sélectionnable:
  - Facultatif.
  - Définit si les éléments à ce niveau de l'arborescence sont sélectionnables ou non.
  - Par défaut, tous les éléments de l'arborescence sont sélectionnables.
  - Les clés utilisées doivent exister dans le tableau name de ce bloc.
- macro:
  - Facultatif si le sélectionnable associé est true
  - Définit les macros pour lesquelles seront stockées les valeurs des éléments de ce niveau de l'arborescence sélectionnés dans le formulaire.* Les valeurs doivent être des macros valides.
  - La même macro peut être définie pour différents éléments d'un même arbre.
  - Les macros ne peuvent pas être identiques aux autres éléments en dehors de l'arborescence.
  - Si aucune macro n'est définie et que le sélectionnable associé est true, la valeur de l'élément de l'arbre sera stockée dans la macro définie pour l'arbre.
  - Les clés utilisées doivent exister dans le tableau name de ce bloc.
- valeur:
  - Obligatoire si le sélectionnable associé est true
  - Définit les valeurs des différents éléments de ce niveau de l'arbre.
  - Les clés utilisées doivent exister dans le tableau name de ce bloc.

1. Les valeurs de la macro seront un JSON de type array avec les différentes valeurs sélectionnées dans l'arborescence qui partagent la même macro. Par exemple:

```
        ["element1A","element1F","element1K"]
```

- enfants:
  - Facultatif.
  - Définit les noms des autres blocs de configuration du même fichier INI à partir desquels seront obtenus les éléments enfants de ce niveau de l'arborescence.
  - Les blocs de configuration du même INI qui ont été utilisés à un niveau supérieur de l'arborescence ne peuvent pas être référencés pour éviter des boucles infinies.
  - Les clés utilisées doivent exister dans le tableau name de ce bloc.

Par exemple, ce serait une façon de définir plusieurs éléments d'un arbre:

```
[custom_tree_1]

name[1] = Performance modules
selectable[1] = false
children[1] = custom_tree_1_A

name[2] = Counter modules
selectable[2] = false
children[2] = custom_tree_1_B

[custom_tree_1_A]

name[1] = Perf1
macro[1] = _param10_
value[1] = p1

name[2] = Perf2
macro[2] = _param10_
value[2] = p2

name[3] = Perf3
macro[3] = _param10_
value[3] = p3

[custom_tree_1_B]

name[1] = Counter1
macro[1] = _param11_
value[1] = c1

name[2] = Counter1
macro[2] = _param11_
value[2] = c2

name[3] = Counter1
macro[3] = _param11_
value[3] = c3
```

L'exemple suivant peut servir de base à ce bloc, puisqu'il comprend tous les paramètres expliqués

avec des commentaires pour chaque cas:

```
[CUSTOM_TREE_DATA_N]

; Mandatory
; Defines the name for the tree element
; Several names can be defined

name[N] = TEXT_N

; Optional
; Defines if tree element is selectable or not
; By default all tree elements are selectable
; Several selectables can be defined

selectable[N] = VALUE_N

; Optional if selectable is true
; Defines the macro where value is stored for the tree element
; Several macros can be defined
; Same macro can be defined for several tree elements inside the same tree
; Macro can't be the same than other outside the tree
; If no macro is defined, value is stored for the global tree macro
; Tree values are stored and used the same way as multiselect values do

macro[N] = FIELD_MACRO_N

; Mandatory if selectable is true
; Defines the value for the tree element
; Several values can be defined

value[N] = VALUE_N

; Optional
; Defines the children elements for the tree element
; CUSTOM_TREE_DATA_M can't be the same than in an upper level
; Several children can be defined

children[N] = CUSTOM_TREE_DATA_M
```

Sélecteurs avec données Pandora FMS

Dans les champs affichés dans les formulaires, les sélecteurs simples et multiples ont la possibilité d'utiliser les données de l'application Pandora FMS elle-même.

En fonction des données souhaitées, les valeurs qui seront stockées et celles qui seront utilisées dans les exécutions varieront:

- groupes_agents:

    - Dans le sélecteur vous utiliserez les noms des groupes d'agents.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID du groupe existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.
- agents:
    - Dans le sélecteur, vous utiliserez les alias d'agent.
    - Stockera vos noms sous forme de données.
    - Lors des exécutions, il vérifiera si l'agent existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.
- module_groups:
    - Dans le sélecteur vous utiliserez les groupes de modules.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID du groupe existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.
- modules:
    - Dans le sélecteur, vous utiliserez les noms des modules.
    - Stockera vos noms sous forme de données.
    - Lors des exécutions, il vérifiera si un module portant ce nom existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.
- module_types:
    - Dans le sélecteur, vous utiliserez les descriptions des types de modules.
    - Stockera vos noms sous forme de données.
    - Lors des exécutions, il utilisera leurs noms comme valeurs.
- Mots clés:
    - Dans le sélecteur vous utiliserez les balises.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID de balise existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.
- statut:
    - Dans le sélecteur vous utiliserez les noms des états du module.
    - Magasinrenverra ses constantes sous forme de données.
    - Dans les exécutions, il utilisera les constantes comme valeurs:
        - 0: Normal
        - 2: Avertissement
        - 1: Critique
        - 3: Inconnu
        - 5: Pas d'initialisation
- modèles_d'alerte:
    - Dans le sélecteur, vous utiliserez les noms de modèles de modules.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID du modèle existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.
- alerte_actions:
    - Dans le sélecteur vous utiliserez les noms des actions d'alerte.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID d'action existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.
- intervalle:
    - Dans le sélecteur, vous utiliserez le sélecteur d'intervalle.
    - Il stockera l'heure en secondes.
    - Dans les exécutions, il utilisera le temps en secondes comme valeur.
- credentials.custom:

    - Dans le sélecteur, vous utiliserez le sélecteur d'informations d'identification personnalisées.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID d'identification existe, et s'il existe, il utilisera une base64 d'un JSON avec les données d'identification comme valeur. Sinon, il restera vide.
    - Le JSON pour ce type d'identifiants aura ce format:

```
{
    "user":"USER",
    "password":"PASSWORD"
}
```

- credentials.aws:
    - Dans le sélecteur, vous utiliserez le sélecteur d'informations d'identification AWS.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID d'identification existe, et s'il existe, il utilisera une base64 d'un JSON avec les données d'identification comme valeur. Sinon, il restera vide.
    - Le JSON pour ce type d'identifiants aura ce format:

```
{
    "access_key_id":"ACCESS_KEY_ID",
    "secret_access_key":"SECRET_ACCESS_KEY"
}
```

- informations d'identification.azure:
    - Dans le sélecteur, vous utiliserez le sélecteur d'informations d'identification Microsoft Azure.
    - Stockera vos identifiants sous forme de données.
    - Lors des exécutions, il vérifiera si l'ID d'identification existe, et s'il existe, il utilisera une base64 d'un JSON avec les données d'identification comme valeur. Sinon, il restera vide.
    - Le JSON pour ce type d'identifiants aura ce format:

```
{
    "client_id":"CLIENT_ID",
    "application_secret":"APPLICATION_SECRET",
    "tenant_domain":"TENANT_DOMAIN",
    "subscription_id":"SUBSCRIPTION_ID"
}
```

- informations d'identification.gcp:
- Dans le sélecteur, vous utiliserez le sélecteur d'informations d'identification Google Cloud Platform.
- Stockera vos identifiants sous forme de données.
- Lors des exécutions, il vérifiera si l'ID d'identification existe, et s'il existe, il utilisera une base64 d'un JSON avec les données d'identification comme valeur. Sinon, il restera vide.
- Le JSON pour ce type d'identifiants aura ce format:

```
{
    "type":"service_account",
    "project_id":"PROJECT_ID",
    "private_key_id":"PRIVATE_KEY_ID",
    "private_key":"PRIVATE_KEY",
    "client_email":"CLIENT_EMAIL",
    "client_id":"CLIENT_ID",
    "auth_uri":"AUTH_URI",
    "token_uri":"TOKEN_URI",
```

```
    "auth_provider_x509_cert_url":"AUTH_PROVIDER_X509_CERT_URL",
    "client_x509_cert_url":"CLIENT_X509_CERT_URL"
}
```

- informations d'identification.sap:
  - Dans le sélecteur, vous utiliserez le sélecteur d'informations d'identification SAP.
  - Stockera vos identifiants sous forme de données.
  - Lors des exécutions, il vérifiera si l'ID d'identification existe, et s'il existe, il utilisera une base64 d'un JSON avec les données d'identification comme valeur. Sinon, il restera vide.
  - Le JSON pour ce type d'identifiants aura ce format:

```
{
    "user":"USER",
    "password":"PASSWORD"
}
```

- informations d'identification.snmp:
  - Dans le sélecteur, vous utiliserez le sélecteur d'informations d'identification SNMP.
  - Stockera vos identifiants sous forme de données.
  - Lors des exécutions, il vérifiera si l'ID d'identification existe, et s'il existe, il utilisera une base64 d'un JSON avec les données d'identification comme valeur. Sinon, il restera vide.
  - Le JSON pour ce type d'identifiants aura ce format:

```
{
    "community":"COMMUNITY",
    "version":"VERSION",
    "securityLevelV3":"SECURITY_LEVEL_V3",
    "authUserV3":"USER_AUTH_V3",
    "authMethodV3":"AUTH_METHOD_V3",
    "authPassV3":"AUTH_PASS_V3",
    "privacyMethodV3":"PRIVACY_METHOD_V3",
    "privacyPassV3":"PRIVACY_PASS_V3"
}
```

- credentials.wmi:
  - Dans le sélecteur, vous utiliserez le sélecteur d'informations d'identification WMI.
  - Stockera vos identifiants sous forme de données.
  - Lors des exécutions, il vérifiera si l'ID d'identification existe, et s'il existe, il utilisera une base64 d'un JSON avec les données d'identification comme valeur. Sinon, il restera vide.
  - Le JSON pour ce type d'identifiants aura ce format:

```
{
    "user":"USER",
    "password":"PASSWORD",
    "namespace":"NAMESPACE"
}
```

- système d'exploitation:
  - Dans le sélecteur, il utilisera les noms du système d'exploitation.
  - Stockera les identifiants sous forme de données.
  - Dans les exécutions, il vérifieIl vérifiera si l'ID du système d'exploitation existe, et s'il existe, il utilisera son nom comme valeur. Sinon, il restera vide.

Fichier de base

L'exemple suivant peut servir de base à la création de fichiers discovery_definition.ini, car il inclut tous les blocs et paramètres possibles expliqués ci-dessus avec des commentaires pour chaque cas:

```
;-------------------------------------;
; DISCOVERY APPLICATION INI FILE BASE ;
;-------------------------------------;

; Mandatory
; Defines global application data

[discovery_extension_definition]

; Mandatory
; Defines discovery application short name
; Short name must be unique
; Short names with "pandorafms." prefix are used by Pandora FMS for official
applications

short_name = DISCOVERY_APPLICATION_UNIQUE_NAME

; Mandatory
; Defines the section where application will be shown in console
; Possible values:
;    app
;    cloud
;    custom

section = app

; Mandatory
; Defines discovery application name, shown in console

name = DISCOVERY_APPLICATION_NAME

; Mandatory
; Defines discovery application version, shown in console

version = VERSION

; Optional
; Defines discovery application description, shown in console

description = DESCRIPTION

; Optional
; Defines execution files inside .disco
; Several execution files can be defined
```

```
execution_file[_EXEC_MACRO_N_] = SCRIPT.pl

; Mandatory
; Defines execution for discovery server
; Several executions can be defined
; At least 1 is required for server execution

exec[] = SERVER_EXECUTION

; Optional
; Defines password encrypt script

passencrypt_script = PASS_ENCRYPT_SCRIPT.pl

; Optional
; Defines password encrypt script execution format
; _passencrypt_script_ is replaced with the passencrypt_script file path
; _password_ is replaced with the string (password) to encrypt

passencrypt_exec = EXECUTION

; Optional
; Defines password decrypt script

passdecrypt_script = PASS_DECRYPT_SCRIPT.pl

; Optional
; Defines password encrypt script execution format
; _passdecrypt_script_ is replaced with the passdecrypt_script file path
; _password_ is replaced with the string (password) to encrypt

passdecrypt_exec = EXECUTION

; Optional
; Defines the default values for the fields when a new task is created
; By default all values are empty or not selected
; Several default values can be defined
; Possible values depending on field type:
;    string            - STRING
;    number            - NUMBER
;    password        - STRING
;    textarea        - STRING
;    checkbox        - BOOL
;    select            - VALUE_N
;    multiselect        - [VALUE_1,VALUE_N]
;    tree            - [VALUE_1,VALUE_N]

default_value[_FIELD_MACRO_N_] = DEFAULT_VALUE

;-------------------------------------;

; Optional
```

```
; Defines application configuration steps

[config_steps]

; Following parameters can be setup for each configuration step
; Several steps can be defined using a different key (N)

; Mandatory
; Defines configuration step name

name[N] = STEP_NAME

; Mandatory if not custom_fields defined
; Defines configuration fields retrieved to console by a command execution
; execution_file scripts can be used to retrieve data
; Command execution output must be JSON as follows
;    [
;        {
;            "macro": "_FIELD_MACRO_N_",
;            "mandatory_field": "BOOL",
;            "name": "FIELD_NAME",
;            "tip": "FIELD_TIP",
;            "type": "FIELD_TYPE",
;            "placeholder": "PLACEHOLDER",
;            "show_on_true": "_FIELD_MACRO_N_",
;            "encrypt_on_true": "_FIELD_MACRO_N_",
;            "select_data": {
;                "VALUE_1": "TEXT_1",
;                "VALUE_N": "TEXT_N"
;            },
;            "tree_data": [
;                {
;                    "name": "TEXT_1",
;                    "selectable": "BOOL_1",
;                    "macro": "_FIELD_MACRO_N_",
;                    "value": "VALUE_1",
;                    "children": [
;                        {
;                            "name": "TEXT_1_1",
;                            "selectable": "BOOL_1_1",
;                            "macro": "_FIELD_MACRO_N_",
;                            "value": "VALUE_1_1",
;                            "children": []
;                        },
;                        {
;                            "name": "TEXT_1_N",
;                            "selectable": "BOOL_1_N",
;                            "macro": "_FIELD_MACRO_N_",
;                            "value": "VALUE_1_N",
;                            "children": []
;                        }
;                    ]
```

```
;                 },
;                 {
;                     "name": "TEXT_N",
;                     "selectable": "BOOL_N",
;                     "macro": "_FIELD_MACRO_N_",
;                     "value": "VALUE_N",
;                     "children": []
;                 }
;             ]
;         }
;     ]

script_data_fields[N] = CONSOLE_EXECUTION_FIELDS

; Mandatory if not script_data_fields defined
; Defines custom configuration fields

custom_fields[N] = CUSTOM_FIELDS_N

; Optional
; Defines the number of fields columns for the configuration steps (1 or 2)

fields_columns[N] = M

;-------------------------------------;

; Mandatory if not script_data_fields defined
; Defines custom configuration fields to be used by configuration steps

[CUSTOM_FIELDS_N]

; Mandatory
; Defines configuration field unique macro
; Macros can be used for script executions, using their value in place

macro[N] = _FIELD_MACRO_N_

; Optional
; Defines if configuration field is mandatory or not
; By default all configuration fields are mandatory

mandatory_field[N] = BOOL

; Mandatory
; Defines configuration field name to be displayed in console
; Macro will be used as name if this field is empty

name[N] = FIELD_NAME

; Optional
; Defines a tip for the field, to be displayed in console
```

```
tip[N] = FIELD_TIP

; Mandatory
; Defines the field type to be displayed in console
; Possible values:
;     string
;    number
;     password
;     textarea
;     checkbox
;     select
;     multiselect
;    tree

type[N] = FIELD_TYPE

; Optional if type is string or textarea
; Defines a placeholder for the field, to be displayed in console

placeholder[N] = PLACEHOLDER

; Optional
; Field is shown in console only if asigned field macro exists, is checkbox and
is true

show_on_true[N] = _FIELD_MACRO_N_

; Optional if type is password
; Field value is encrypted when stored into database if asigned field macro
exists, is checkbox and is true
; Password encrypt and decrypt depends on scripts uploaded to the discovery
application

encrypt_on_true[N] = _FIELD_MACRO_N_

; Mandatory if type is select or multiselect
; Defines select data to be displayed in console
; Possible values:
;     agent_groups        - Uses agent groups names
;     agents                - Uses agents names
;     module_groups        - Uses module groups
;    modules                - Uses modules names
;    module_types        - Uses module types names
;    tags                - Uses module tags
;    status                - Uses module status names
;    alert_templates        - Uses alert templates names
;    alert_actions        - Uses alert actions names
;    interval            - Uses time interval selector
;    credentials.custom    - Uses pandora custom credentials selector
;    credentials.aws        - Uses pandora AWS credentials selector
;    credentials.azure    - Uses pandora Microsoft Azure credentials selector
;    credentials.gcp        - Uses pandora Google Cloud Platform credentials
```

```
selector
;    credentials.sap        - Uses pandora SAP credentials selector
;    credentials.snmp    - Uses pandora SNMP credentials selector
;    credentials.wmi        - Uses pandora WMI credentials selector
;    os                    - Uses pandora OS names
;     CUSTOM_SELECT_N        - Uses custom selector values
; multiselect fields value is a comma separated list of selected values

select_data[N] = FIELD_DATA

; Mandatory if type is tree
; Defines tree data to be displayed in console
; tree fields value is a comma separated list of selected values

tree_data[N] = CUSTOM_TREE_DATA_N

;-------------------------------------;

; Mandatory if custom tree defined
; Defines custom tree values to be used by configuration fields

[CUSTOM_TREE_DATA_N]

; Mandatory
; Defines the name for the tree element
; Several names can be defined

name[N] = TEXT_N

; Optional
; Defines if tree element is selectable or not
; By default all tree elements are selectable
; Several selectables can be defined

selectable[N] = VALUE_N

; Optional if selectable is true
; Defines the macro where value is stored for the tree element
; Several macros can be defined
; Same macro can be defined for several tree elements inside the same tree
; Macro can't be the same than other outside the tree
; If no macro is defined, value is stored for the global tree macro
; Tree values are stored and used the same way as multiselect values do

macro[N] = FIELD_MACRO_N

; Mandatory if selectable is true
; Defines the value for the tree element
; Several values can be defined

value[N] = VALUE_N
```

```
; Optional
; Defines the children elements for the tree element
; CUSTOM_TREE_DATA_M can't be the same than in an upper level
; Several children can be defined

children[N] = CUSTOM_TREE_DATA_M

;-------------------------------------;

; Mandatory if custom select or multiselect defined
; Defines custom select or multiselect values to be used by configuration fields

[CUSTOM_SELECT_N]

; Mandatory
; Defines the value-text pair for the select or multiselect
; Several value-text pairs can be defined

option[VALUE_N] = TEXT_N

;-------------------------------------;

; Optional
; Defines temporary configuration files to be created and used during scrips
executions

[tempfile_confs]

; Mandatory
; Defines the content for the temporary file
; File will be used where temporary file macro is specified during executions
; File content replaces fields macros with their values

file[_TEMP_FILE_MACRO_N_] = _FIELD_MACRO_N_,_FIELD_MACRO_M_
```

Exemple

Ci-dessous, nous montrons un exemple du fichier discovery_definition.ini et comment sa forme serait affichée dans la console Pandora FMS:

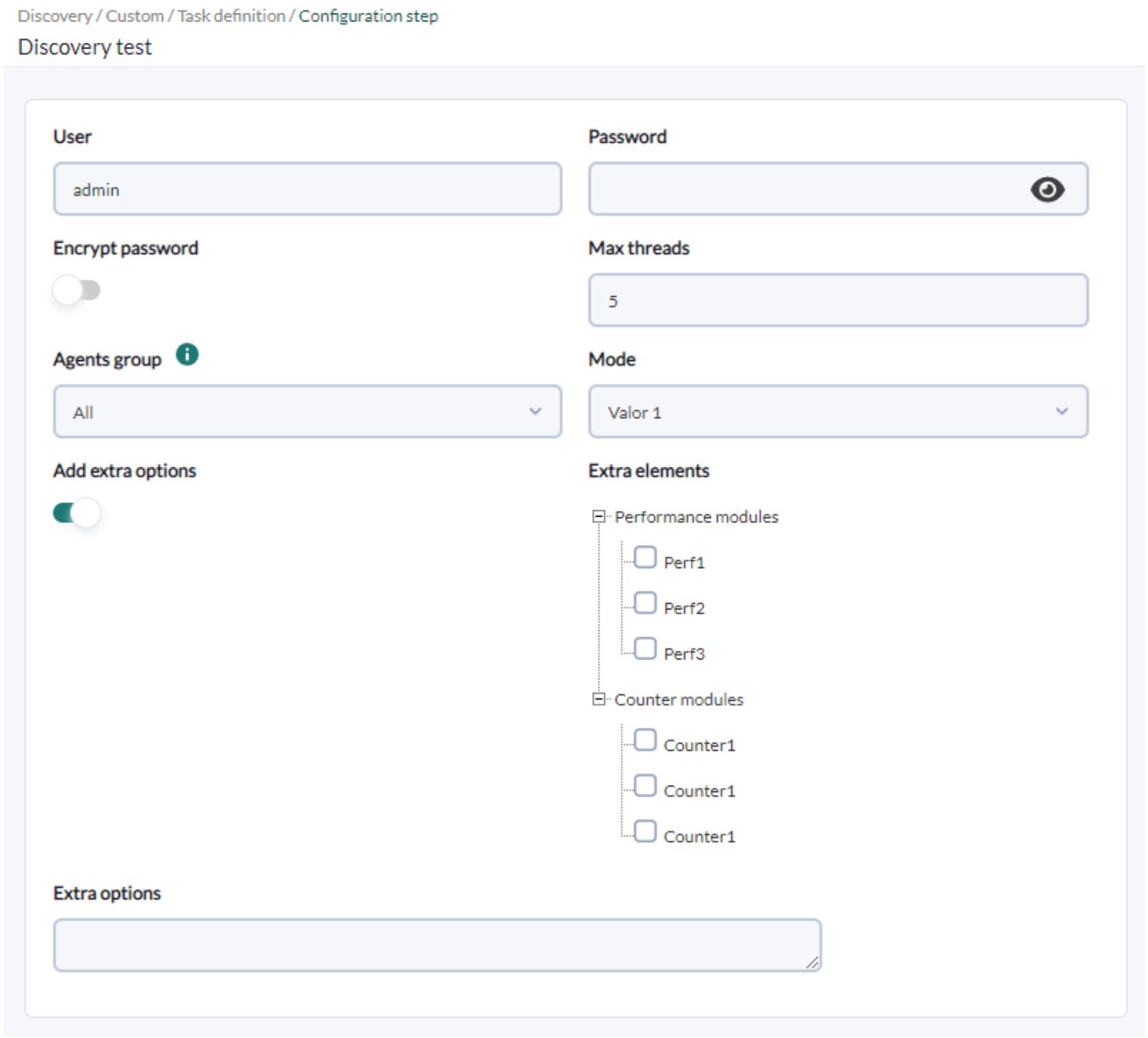
```
[discovery_extension_definition]

short_name = discoveryTest
section = custom
name = Discovery test
version = "1.0"
description = A test discovery //plugin//

execution_file[_exec1_] = test.py
```

```
exec[] = "'_exec1_' -c '_tempConf_'"

passencrypt_script = pass_encrypter.py
passencrypt_exec = "'_passencrypt_script_' --encrypt '_password_'"

passdecrypt_script = pass_encrypter.py
passencrypt_exec = "'_passdecrypt_script_' --decrypt '_password_'"

default_value[_param1_] = "admin"
default_value[_param2_] = ""
default_value[_param3_] = false
default_value[_param4_] = 5
default_value[_param5_] = 0
default_value[_param6_] = v1
default_value[_param7_] = true
default_value[_param8_] = "[]"
default_value[_param9_] = ""
default_value[_param10_] = "[]"
default_value[_param11_] = "[]"

[config_steps]

name[1] = Configuration step
custom_fields[1] = custom_fields_1

[tempfile_confs]

file[_tempConf_] = "user _param1_
password _param2_
encrypt_pass _param3_

threads _param4_

group _param5_

mode _param6_
extra_options _param7_
extra_elements _param8_
extra_perfs _param10_
extra_counters _param11_

log __temp__/__taskMD5__.log

_param9_"

[custom_fields_1]

macro[1] = _param1_
name[1] = User
type[1] = string

macro[2] = _param2_
```

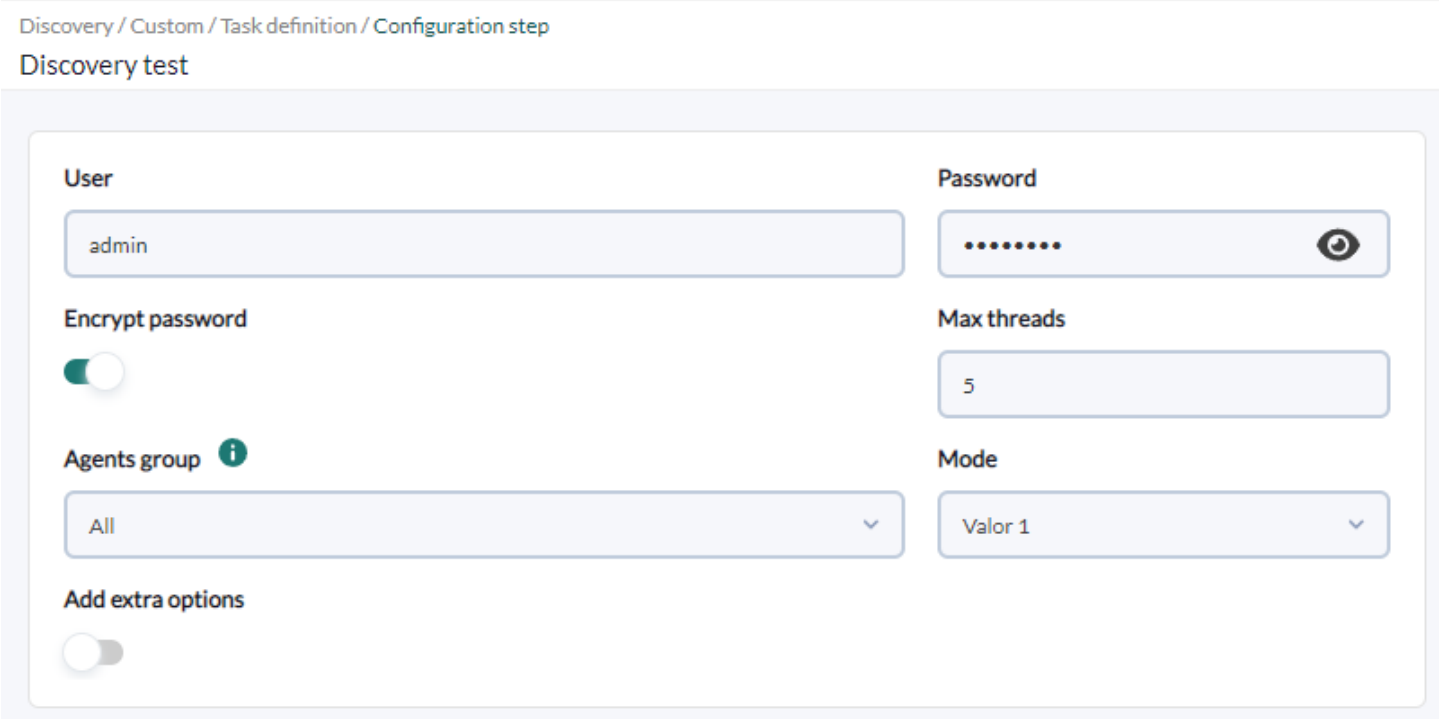
```
name[2] = Password
type[2] = password
encrypt_on_true[2] = _param3_

macro[3] = _param3_
name[3] = Encrypt password
type[3] = checkbox

macro[4] = _param4_
name[4] = Max threads
type[4] = number
mandatory_field[4] = false

macro[5] = _param5_
name[5] = Agents group
tip[5] = Agents are generated in this group
type[5] = select
select_data[5] = agent_groups

macro[6] = _param6_
name[6] = Mode
type[6] = select
select_data[6] = custom_select_1

macro[7] = _param7_
name[7] = Add extra options
type[7] = checkbox

macro[8] = _param8_
name[8] = Extra elements
type[8] = tree
tree_data[8] = custom_tree_1
show_on_true[8] = _param7_

macro[9] = _param9_
name[9] = Extra options
type[9] = textarea
placeholder[9] = "Add extra options here"
show_on_true[9] = _param7_

[custom_select_1]

option[v1] = Valor 1
option[v2] = Valor 2
option[v3] = Valor 3
option[v4] = Valor 4
option[v5] = Valor 5

[custom_tree_1]

name[1] = Performance modules
selectable[1] = false
```

```
children[1] = custom_tree_1_A

name[2] = Counter modules
selectable[2] = false
children[2] = custom_tree_1_B

[custom_tree_1_A]

name[1] = Perf1
selectable[1] = true
macro[1] = _param10_
value[1] = p1

name[2] = Perf2
selectable[2] = true
macro[2] = _param10_
value[2] = p2

name[3] = Perf3
selectable[3] = true
macro[3] = _param10_
value[3] = p3

[custom_tree_1_B]

name[1] = Counter1
selectable[1] = true
macro[1] = _param11_
value[1] = c1

name[2] = Counter1
selectable[2] = true
macro[2] = _param11_
value[2] = c2

name[3] = Counter1
selectable[3] = true
macro[3] = _param11_
value[3] = c3
```

Avec la définition ci-dessus, lors de la création d'une tâche pour l'application, ce serait le formulaire qui s'afficherait:



Les champs “Éléments supplémentaires” et “Options supplémentaires” sont masqués lorsque “Ajouter des options supplémentaires” est désélectionné, et le champ “Mot de passe” crypterait sa valeur en ayant sélectionné le champ “Crypter le mot de passe”:



Lors de l'exécution de cette tâche par le serveur, un fichier temporaire était généré (par exemple `/var/spool/pandora/data_in/discovery/tmp/6d7fce9fee471194aa8b5b6e47267f03`) dont le contenu pouvait être:

```
user admin
password MjZhYjBkYjkwZDcyZTI4YWQwYmExZTIyZWU1MTA1MTAgIC0K
encrypt_pass 1

threads 2

group Applications

mode v1
extra_options 1
extra_elements []
extra_perfs ["p1", "p2"]
extra_counters ["c1"]

log /tmp/b026324c6904b2a9cb4b88d6d61c81d1.log
```

Et que j'utiliserais lors de l'exécution, qui ressemblerait à ceci:

```
'/var/spool/pandora/data_in/discovery/discoveryTest/test.py' -c
'/var/spool/pandora/data_in/discovery/tmp/6d7fce9fee471194aa8b5b6e47267f03'
```

Scripts et exécutables

Le serveur Pandora FMS se chargera d'exécuter les scripts et exécutables définis dans le fichier `discovery_definition.ini` et déterminera le résultat et le résumé de l'exécution de chaque

tâche en fonction de la sortie et du code d'erreur de chacune des tâches. ... les exécutions ont eu lieu.

Pour déterminer l'état, il vérifiera le code d'erreur renvoyé pour chacune des exécutions qu'il effectue. Si au moins une des exécutions renvoie un code d'erreur autre que « 0 », l'état de la tâche sera considéré comme échec.

Pour afficher le résumé de la tâche, pour chacune des exécutions effectuées par le serveur Pandora FMS, il collectera sa sortie, à la fois la sortie standard (STDOUT) et la sortie d'erreur (STDERR). En règle générale, vous vous attendez à obtenir une sortie au format JSON minimal suivant:

```
{
    "summary": {
        "SUMMARY_FIELD_1": "SUMMARY_VALUE_1",
        "SUMMARY_FIELD_N": "SUMMARY_VALUE_N"
    },
    "info": "ADDITIONAL_INFO"
}
```

Dans la console la clé résumé sera lue comme un tableau à deux colonnes, en considérant ses clés comme les éléments de gauche et ses valeurs comme les éléments de droite.

La clé info sera lue comme information supplémentaire, ajoutant une ligne supplémentaire au tableau récapitulatif.

Toute autre clé JSON ou si un JSON n'est pas renvoyé ou n'est pas conforme à cette structure, le reste des informations sera ajouté dans une ligne supplémentaire dans le tableau récapitulatif.

Un tableau récapitulatif sera généré pour chacune des exécutions, les listant en fonction de l'ordre des exécutions effectuées par le serveur. De cette façon, le tableau “Résumé 1” correspondrait à la première exécution, le tableau “Résumé 2” à la seconde et ainsi de suite.

Cela vise à garantir que le résultat de la dernière exécution soit visible à tout moment dans la console, qu'elle ait réussi ou échoué.

Enfin, outre les actions effectuées par les scripts ou exécutables, le serveur Pandora FMS pourra traiter les agents et les modules en fonction du résultat des exécutions. Pour ce faire, le serveur Pandora FMS s'attendra à recevoir dans la sortie JSON une clé supplémentaire monitoring_data avec les informations de chacun des agents et modules qu'il doit traiter, mais les données de cette clé ne seront pas stockées dans le résumé d'exécution.

Ainsi, le format de sortie JSON attendu pour les plugins Discovery est le suivant:

```
{
    "summary": {
```

```
        "SUMMARY_FIELD_1": "SUMMARY_VALUE_1",
        "SUMMARY_FIELD_N": "SUMMARY_VALUE_N"
    },
    "info": "ADDITIONAL_INFO",
    "monitoring_data": [
        {
            "agent_data": {
                "agent_name": "AGENT_NAME",
                "agent_alias": "AGENT_ALIAS",
                "os": "OS",
                "os_version": "OS_VERSION",
                "interval": "INTERVAL",
                "id_group": "ID_GROUP",
                "address": "ADDRESS",
                "description": "DESCRIPTION",
                "agent_version": "AGENT_VERSION",
                "parent_agent_name": "PARENT_AGENT_NAME",
                "timezone_offset": "TIMEZONE_OFFSET"
            },
            "module_data": [
                {
                    "name": "NAME",
                    "data": "DATA",
                    "type": "TYPE",
                    "description": "DESCRIPTION",
                    "max": "MAX",
                    "min": "MIN",
                    "post_process": "POST_PROCESS",
                    "module_interval": "MODULE_INTERVAL",
                    "min_critical": "MIN_CRITICAL",
                    "max_critical": "MAX_CRITICAL",
                    "min_warning": "MIN_WARNING",
                    "max_warning": "MAX_WARNING",
                    "disabled": "DISABLED",
                    "min_ff_event": "MIN_FF_EVENT",
                    "datalist": "DATALIST",
                    "status": "STATUS",
                    "unit": "UNIT",
                    "timestamp": "TIMESTAMP",
                    "module_group": "MODULE_GROUP",
                    "custom_id": "CUSTOM_ID",
                    "str_warning": "STR_WARNING",
                    "str_critical": "STR_CRITICAL",
                    "critical_instructions": "CRITICAL_INSTRUCTIONS",
                    "warning_instructions": "WARNING_INSTRUCTIONS",
                    "unknown_instructions": "UNKNOWN_INSTRUCTIONS",
                    "tags": "TAGS",
                    "critical_inverse": "CRITICAL_INVERSE",
                    "warning_inverse": "WARNING_INVERSE",
                    "quiet": "QUIET",
                    "module_ff_interval": "MODULE_FF_INTERVAL",
                    "alert_template": "ALERT_TEMPLATE",
```

```
                    "crontab": "CRONTAB",
                    "min_ff_event_normal": "MIN_FF_EVENT_NORMAL",
                    "min_ff_event_warning": "MIN_FF_EVENT_WARNING",
                    "min_ff_event_critical": "MIN_FF_EVENT_CRITICAL",
                    "ff_timeout": "FF_TIMEOUT",
                    "each_ff": "EACH_FF",
                    "module_parent": "MODULE_PARENT",
                    "module_parent_unlink": "MODULE_PARENT_UNLINK",
                    "cron_interval": "CRON_INTERVAL",
                    "ff_type": "FF_TYPE",
                    "min_warning_forced": "MIN_WARNING_FORCED",
                    "max_warning_forced": "MAX_WARNING_FORCED",
                    "min_critical_forced": "MIN_CRITICAL_FORCED",
                    "max_critical_forced": "MAX_CRITICAL_FORCED",
                    "str_warning_forced": "STR_WARNING_FORCED",
                    "str_critical_forced": "STR_CRITICAL_FORCED"
                }
            ]
        }
    ]
}
```