PANDORAFMS

.Disco development

.Disco development

Paquetes ".disco"

Discovery permite cargar tanto *plugins* oficiales de Pandora FMS como personalizados.

Para cargar *plugins* personalizados es necesario generar un paquete `.disco` con todo lo necesario para que tanto la consola como el servidor de Pandora FMS sean capaces de:

- Mostrar el interfaz de configuración para nuevas tareas de discovery.
- Ejecutar las tareas de discovery configuradas en el entorno.

Un paquete `.disco` es un fichero zip con la extensión .disco el cual contiene al menos un fichero llamado discovery_definition.ini.

Opcionalmente, un paquete `.disco` podrá contener un fichero llamado logo.png el cual será el logo del *plugin* en la consola de Pandora FMS. Si no se añade un fichero para el logo, la consola automáticamente usa un logo por defecto.

Por último, típicamente, un fichero `.disco` contendrá todos los *scripts*, ejecutables y librerías necesarios por el servidor y la consola. Aunque no es obligatorio sí es lo común y recomendado, para evitar requisitos adicionales de instalación en los sistemas donde se quieran configurar y ejecutar las tareas que usen el *plugin*.

Por lo tanto, y en resumen, un fichero `.disco` contendrá:

- `discovery_definition.ini`.
- `logo.png` (opcional).
- *Scripts*, ejecutables y librerías.

Fichero "discovery_definition.ini"

El fichero discovery_definition.ini es el más importante dentro de un fichero `.disco`, ya que es el que contiene toda la definición del *plugin*.

Contiene tanto los parámetros que se mostrarán por consola para rellenarse en un formulario de definición de una tarea, como las ejecuciones que tendrá que realizar el servidor de Pandora FMS para las tareas del *plugin*.

Con el objetivo de facilitar su definición y procesado en la consola, el formato usado en un fichero `discovery_definition.ini` es el formato INI propio de PHP.

Un fichero `discovery_definition.ini` se compone fundamentalmente de 3 bloques:

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

Conceptos comunes

Durante las siguientes explicaciones será común usar algunos términos. En esta sección se explican dichos términos.

Una macro es una clave de texto única usada para almacenar información (valores, rutas a ficheros, ...). Cuando una macro es usada (por ejemplo durante una ejecución) lo que se quiere es usar la información que contenga en su lugar.

Una macro valida es una clave de texto que comienza y termina con guiones bajos _ y que entre medias solo puede contener letras (A-Z y a-z) y números (0-9).

Cuando se hable de STRING nos referiremos a cadenas de texto alfanuméricas.

Cuando se hable de NUMBER nos referiremos exclusivamente a números.

Cuando se hable de BOOL nos referiremos a valores true / false, 1 / 0, yes / no.

Cuando se hable de VALUE nos referiremos a un valor cualquiera (STRING, NUMBER o BOOL). Normalmente se usará en listas separadas por comas para representar valores de un mismo tipo.

Cuando se usen las letras N o M en mayúsculas, nos referiremos a un número entero positivo. Normalmente se usará junto con otros elementos como VALUE para indicar que puede repetirse más de una vez (Por ejemplo, VALUE_N).

Macros del servidor

Gran parte de la configuración de las ejecuciones para el servidor se basa en el uso de macros. Principalmente, éstas serán definidas por el usuario, pero existen algunas concretas propias del servidor de Pandora FMS que pueden resultar útiles:

- `__taskMD5__` → MD5 único de la tarea, generado a partir del ID de la tarea y el nombre corto de la aplicación. Es útil para generar elementos únicos durante las ejecuciones (como por ejemplo ficheros).
- `__taskInterval__` → Intervalo de ejecución de la tarea (segundos).
- `__taskGroup__` → Grupo de la tarea (Texto).
- `__taskGroupID__` → Grupo de la tarea (ID).
- `__temp__` → Directorio temporal del servidor configurado en el `pandora_server.conf` (temporal).
- `__incomingDir__` → Directorio de entrada del servidor de Pandora configurado en `pandora_server.conf` (incomingdir).

- `__consoleAPIURL__` → API URL configurado en el `pandora_server.conf` (console_api_url).
- `__consoleAPIPass__` → API pass configurada en el `pandora_server.conf` (console_api_pass).
- `__consoleUser__` → Usuario de la consola configurado en el `pandora_server.conf` (console_user).
- `__consolePass__` → Contraseña de la consola configurada en el `pandora_server.conf` (console_pass).
- `__pandoraServerConf__` → Ruta completa al fichero de configuración del servidor de Pandora FMS `pandora_server.conf`.

Estas macros en ningún caso son utilizadas por la consola de Pandora FMS.

Bloque discovery_extension_definition

Este bloque define la configuración principal del *plugin* y cuenta con los siguientes parámetros:

- short_name:
  - Obligatorio.
  - Define el nombre corto del *plugin*.
  - El nombre corto debe ser único.
  - Los nombres cortos que comiencen con el prefijo "pandorafms." son usados por los *plugins* oficiales de Pandora FMS.
  - Los nombres cortos solo pueden contener letras (`A-Z` y `a-z`), números (`0-9`), puntos (`.`), guiones (`-`) y guiones bajos (_).
- section:
  - Obligatorio.
  - Define la sección de la consola donde se mostrará el *plugin*.
  - Puede ser app, cloud o custom.
- name:
  - Obligatorio.
  - Define el nombre del *plugin* que se mostrará en la consola.
- version:
  - Obligatorio.
  - Define la versión del *plugin* que se mostrará en la consola.
  - Es recomendado cambiar la versión si se hacen cambios en un *plugin*, por menores que sean.
- description:
  - Opcional.
  - Define la descripción del *plugin* que se mostrará en la consola.
- execution_file:
  - Opcional.
  - Indica las rutas relativas (dentro del fichero `.disco`) a scripts y ejecutables usados por el *plugin*.
  - A todos los ficheros indicados se les dará permiso de ejecución.
  - Este parámetro es un array cuyas claves serán macros válidas que se usarán para indicar el uso de los ficheros del *plugin*.
  - Al tratarse de un array este parámetro puede indicarse varias veces usando distintas claves.
  - Por ejemplo:

```
```

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

- exec:
  - Obligatorio.
  - Define el formato de las ejecuciones que debe realizar el servidor de Pandora FMS para las tareas del *plugin*.
  - Este parámetro es un array cuyas claves serán posicionales, es decir, se podrán indicar con números o no indicarse.
  - Al tratarse de un array este parámetro puede indicarse varias veces usando distintas claves.
  - Será común usar macros en las ejecuciones definidas tanto para indicar las rutas a los scripts o ejecutables usados como para indicar los parámetros de dichos scripts o ejecutables.
  - El servidor de Pandora FMS lanzará cada una de las ejecuciones definidas en orden y almacenará el resultado de todas ellas para producir la salida del *plugin* en la tarea y obtener su estado.
  - Por ejemplo:

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

- passencrypt_script:
  - Opcional.
  - Define la ruta relativa (dentro del fichero `.disco`) a un script usado para encriptar campos de tipo contraseña desde la consola.
  - Al fichero indicado se le dará permiso de ejecución.
- passencrypt_exec:
  - Opcional.
  - Define el formato de la ejecución del script para encriptar contraseñas desde la consola.
  - El resultado esperado de dicha ejecución será un texto que se corresponda con la contraseña encriptada.
  - Solo admite el uso de las macros _passencrypt_script_ y _password_ en su definición.
  - La macro _passencrypt_script_ es sustituida por la ruta al fichero definido en passencrypt_script.
  - La macro _password_ es sustituida por el valor del campo tipo contraseña que deba encriptar en cada momento.
  - Por ejemplo:

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

- passdecrypt_script:
  - Opcional.
  - Define la ruta relativa (dentro del fichero `.disco`) a un script usado para desencriptar campos de tipo contraseña desde la consola.
  - Al fichero indicado se le dará permiso de ejecución.
- passdecrypt_exec:
  - Opcional.
  - Define el formato de la ejecución del script para desencriptar contraseñas desde la consola.
  - El resultado esperado de dicha ejecución será un texto que se corresponda con la contraseña desencriptada.
  - Solo admite el uso de las macros _passdecrypt_script_ y _password_ en su definición.
  - La macro _passdecrypt_script_ es sustituida por la ruta al fichero definido en passdecrypt_script.
  - La macro _password_ es sustituida por el valor del campo tipo contraseña que deba

desencriptar en cada momento.
    - Por ejemplo:

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

- default_value:
    - Opcional.
    - Define los valores por defecto para las distinta macros usadas durante las ejecuciones, es decir, los valores por defecto para los datos almacenados para cada tarea de *Discovery* creada para este *plugin*.
    - Este parámetro es un array cuyas claves serán macros válidas.
    - Al tratarse de un array este parámetro puede indicarse varias veces usando distintas claves.
    - Dependiendo del tipo de campo usado en los formularios de la consola (ver más adelante) los valores por defecto podrán ser:
        1. string: *STRING* .
        2. number: *NUMBER* .
        3. password: *STRING* .
        4. textarea: *STRING* .
        5. checkbox: *BOOL* .
        6. select: *VALUE_N* .
        7. multiselect: *[VALUE_1,VALUE_N]* .
        8. tree: *[VALUE_1,VALUE_N]* .
    - Se recomienda incluir un default_value para cada una de las macros que pueda haber en los formularios de configuración de las tareas.
    - Por ejemplo:

```
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_] = "[]"
```

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

```
[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
```

Bloque config_steps

Este bloque define los pasos de configuración para las tareas del *plugin* y cuenta con los siguientes parámetros:

Estas normas se aplican para todos los parámetros de este bloque:

- Los parámetros son arrays cuyas claves serán posicionales, es decir, se podrán indicar con números o no indicarse. Se recomienda indicar las claves.
- Al traterse de arrays los parámetros pueden indicarse varias veces usando distintas claves.
- Todos los parámetros que comparten una clave, hacen referencia a un mismo elemento. Es decir, asignan distintos valores (según el parámetro) al mismo elemento.

- name:
  - Obligatorio.
  - Define los nombres de los distintos pasos de configuración en las tareas.
- script_data_fields:
  - Obligatorio si no se define un custom_fields para la misma clave.
  - Opcional si se define un custom_fields para la misma clave.
  - Define el formato de las ejecuciones que debe realizar la consola de Pandora FMS para los formularios que se generen dinámicamente. Por ejemplo, si se va a monitorizar un entorno de virtualización y se quiere sacar un listado de máquinas virtuales para seleccionar las que se desee monitorizar.
  - Las claves usadas deben existir en el array name de este bloque.
  - Será común usar macros en las ejecuciones definidas tanto para indicar las rutas a los scripts o ejecutables usados como para indicar los parámetros de dichos scripts o ejecutables.

- El resultado de las ejecuciones indicadas aquí debe ser un JSON con el siguiente formato. Las claves del JSON se corresponden con los parámetros del bloque CUSTOM_FIELDS (ver más adelante):

```
[
    {
        "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": []
            }
        ]
    }
]
```

- custom_fields:
  - Obligatorio si no se define un script_data_fields para la misma clave.
  - Opcional si se define un script_data_fields para la misma clave.
  - Define los bloques de configuración del mismo fichero INI a partir de los cuales se obtendrán los campos de los formularios al crear tareas en la consola de Pandora FMS.
  - Las claves usadas deben existir en el array name de este bloque.
- fields_columns:
  - Opcional.
  - Define el número de columnas en los que se distribuirán los campos de los formularios de las tareas en cada paso de configuración.
  - Las claves usadas deben existir en el array name de este bloque.
  - Puede asignarsele el valor 1 o 2.
  - Su valor por defecto es 2.

Por ejemplo, esta sería una forma de definir varios pasos de configuración:

```
[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
```

En caso de indicarse script_data_fields y custom_fields para un mismo paso de configuración, el formulario mostrará primero los campos obtenidos por el script_data_fields y a continuación los definidos en el custom_fields.

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

```
[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
```

Bloque tempfile_confs

Este bloque define el formato para los ficheros de configuración temporales usados por el *plugin* y cuenta con los siguientes parámetros:

- file:
  - Obligatorio.
  - Define los contenidos para ficheros de configuración temporales que pueden ser usados durante las ejecuciones del servidor y la consola de Pandora FMS.
  - Este parámetro es un array cuyas claves serán macros válidas que se usarán para indicar el uso de los ficheros temporales en el *plugin*.
  - Al tratarse de un array este parámetro puede indicarse varias veces usando distintas claves.
  - Cuando una de las claves de este array es indicada durante una ejecución, se genera un fichero temporal cuyo contenido es el indicado en este array y el valor de la macro se sustituye por la ubicación de dicho fichero temporal. Una vez finalizada la ejecución, el fichero es eliminado.
  - Dentro del contenido de los ficheros de configuración temporales, es posible indicar otras macros, para que sean sustituidas por el valor correspondiente.
  - Por ejemplo:

```
        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_
```

Bloques CUSTOM_FIELDS

Este tipo de bloques puede tener el nombre que se quiera, siempre que se haya asignado como valor dentro del bloque config_steps para su parámetro custom_fields. Definen los campos para

los formularios en los pasos de configuración del *plugin* y cuenta con los siguientes parámetros:

Estas normas se aplican para todos los parámetros de este bloque:

- Los parámetros son arrays cuyas claves serán posicionales, es decir, se podrán indicar con números o no indicarse. Se recomienda indicar las claves.
- Al tratarse de arrays los parámetros pueden indicarse varias veces usando distintas claves.
- Todos los parámetros que comparten una clave, hacen referencia a un mismo elemento. Es decir, asignan distintos valores (según el parámetro) al mismo elemento.

- macro:
  - Obligatorio.
  - Define las macros en las que se almacenarán los valores de configuración de la tarea.
- mandatory_field:
  - Opcional.
  - Define mediante valores tipo BOOL si el campo de configuración del formulario debe tener un valor obligatoriamente o no.
  - Las claves usadas deben existir en el array macro de este bloque.
  - Por defecto todos los campos son obligatorios.
  - Aun siendo obligatorios, los campos de tipo multiselect y tree admiten que no se seleccionen elementos.
- name:
  - Obligatorio.
  - Define los nombres de los campos de configuración del formulario.
  - Las claves usadas deben existir en el array macro de este bloque.
- tip:
  - Opcional.
  - Define las ayudas de los campos de configuración del formulario.
  - Las claves usadas deben existir en el array macro de este bloque.
- type:
  - Obligatorio.
  - Define los tipos de los campos de configuración del formulario.
  - Las claves usadas deben existir en el array macro de este bloque.
  - Sus valores posibles son:
    - string: El campo será una caja de texto. Admitirá valores de tipo STRING.
    - number: El campo será una caja de introducción de números. Admitirá valores de tipo NUMBER.
    - password: El campo será una caja de texto cuyo valor se ocultará. Admitirá valores de tipo STRING.
    - textarea: El campo será una caja de texto amplia. Admitirá valores de tipo STRING.
    - checkbox: El campo admitirá valores de tipo BOOL
    - select: El campo será un desplegable en el que seleccionar un solo valor. Admitirá valores de tipo VALUE.
    - multiselect: El campo será una caja de selección múltiple a partir de varias opciones. Admitirá valores de tipo VALUE.
    - tree: El campo será un árbol de distintos niveles cuyos elementos podrán seleccionarse o no para enviar sus valores. Admitirá valores de tipo VALUE.
- placeholder:
  - Opcional si el type asociado es string o textarea.
  - Define los textos que se mostrarán en las cajas de texto a modo de ejemplos.
  - Las claves usadas deben existir en el array macro de este bloque.

- show_on_true:
  - Opcional.
  - Define macros de otros campos de tipo checkbox del formulario como valores, para que cuando dichas macros tengan un valor true en el formulario, se muestre el campo deseado.
  - Las claves usadas deben existir en el array macro de este bloque.
- encrypt_on_true:
  - Opcional si el type asociado es password.
  - Define macros de otros campos de tipo checkbox del formulario como valores, para que cuando dichas macros tengan un valor true en el formulario, se encripte al guardar la configuración el campo deseado.
  - La encriptación/desencriptación se realizará en base a la configuración en el bloque discovery_extension_definition.
  - Las claves usadas deben existir en el array macro de este bloque.
- select_data:
  - Obligatorio si el type asociado es select o multiselect.
  - Define el origen de los valores para los desplegables.
  - Los selectores se pueden generar con datos de Pandora FMS o personalizados. Para ello admite los siguientes valores:
    - El nombre de otros bloques de configuración del mismo fichero INI a partir de los cuales se obtendrán los elementos del tipo select o multiselect.
      - agent_groups: Utiliza los grupos de agentes como datos.
      - agents: Utiliza los agentes como datos.
      - module_groups: Utiliza los grupos de módulos como datos.
      - modules: Utiliza los módulos como datos.
      - module_types: Utiliza los tipos de módulos como datos.
      - tags: Utiliza los tags como datos.
      - status: Utiliza los estados como datos.
      - alert_templates: Utiliza las plantillas de alertas como datos.
      - alert_actions: Utiliza las acciones de alertas como datos.
      - interval: Utiliza el selector de tiempo como datos.
      - credentials.custom: Utiliza el selector de credenciales personalizadas como datos.
      - credentials.aws: Utiliza el selector de credenciales de AWS como datos.
      - credentials.azure: Utiliza el selector de credenciales de Azure como datos.
      - credentials.gcp: Utiliza el selector de credenciales de Google Cloud como datos.
      - credentials.sap: Utiliza el selector de credenciales de SAP como datos.
      - credentials.snmp: Utiliza el selector de credenciales de SNMP como datos.
      - credentials.wmi: Utiliza el selector de credenciales de WMI como datos.
      - os: Utiliza los OS como datos.
  - Las claves usadas deben existir en el array macro de este bloque.
  - Los valores de la macro si el type asociado es multiselect serán un JSON de tipo array con los distintos valores seleccionados. Por ejemplo:

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

- tree_data:
  - Obligatorio si el type asociado es tree.
  - Define los bloques de configuración del mismo fichero INI a partir de los cuales se obtendrán los elementos del tipo tree.
  - Las claves usadas deben existir en el array macro de este bloque.
  - Los valores de la macro serán un JSON de tipo array con los distintos valores seleccionados. Por ejemplo:

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

Por ejemplo, esta sería una forma de definir varios campos de configuración:

```
[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_
```

El siguiente ejemplo puede usarse como base para este bloque, ya que incluye todos los

parámetros explicados con comentarios para cada caso:

```
[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
```

Bloques CUSTOM_SELECT

Este tipo de bloques puede tener el nombre que se quiera, siempre que se haya asignado como valor dentro de un bloque CUSTOM_FIELDS para su parámetro select_data. Definen las opciones de un desplegable del formulario de configuración del *plugin* y cuenta con los siguientes parámetros:

- option:

- Obligatorio.
- Define las opciones de un selector personalizado.
- Este parámetro es un array cuyas claves son los valores de las opciones del selector personalizado, y sus valores son los textos mostrados para cada opción en el selector personalizado.
- Al tratarse de un array este parámetro puede indicarse varias veces usando distintas claves.

Por ejemplo:

```
[custom_select_1]

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

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

```
[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
```

Bloques CUSTOM_TREE_DATA

Este tipo de bloques puede tener el nombre que se quiera, siempre que se haya asignado como valor dentro de un bloque CUSTOM_FIELDS para su parámetro tree_data o dentro de otro bloque CUSTOM_TREE_DATA para su parámetro children. Definen los elementos de un árbol del formulario de configuración del *plugin* y cuenta con los siguientes parámetros:

Estas normas se aplican para todos los parámetros de este bloque:

- Los parámetros son arrays cuyas claves serán posicionales, es decir, se podrán indicar con números o no indicarse. Se recomienda indicar las claves.
- Al tratarse de arrays los parámetros pueden indicarse varias veces usando distintas claves.
- Todos los parámetros que comparten una clave, hacen referencia a un mismo elemento. Es decir, asignan distintos valores (según el parámetro) al mismo elemento.

- name:
  - Obligatorio.
  - Define los nombres de los distintos elementos de este nivel del árbol.
- selectable:
  - Opcional.
  - Define si los elementos de este nivel del árbol son seleccionables o no.
  - Por defecto todos los elementos del árbol son seleccionables.
  - Las claves usadas deben existir en el array name de este bloque.
- macro:
  - Opcional si el selectable asociado es true
  - Define las macros para las cuales se almacenarán los valores de los elementos de este nivel del árbol que se seleccionen en el formulario.
  - Los valores deben ser macros validas.
  - La misma macro puede ser definida para distintos elementos del mismo árbol.
  - Las macros no pueden ser las mismas que las de otros elementos fuera del árbol.
  - Si no se define una macro y el selectable asociado es true, el valor del elemento del árbol se almacenará en la macro definida para el árbol.
  - Las claves usadas deben existir en el array name de este bloque.
- value:
  - Obligatorio si el selectable asociado es true
  - Define los valores de los distintos elementos de este nivel del árbol.
  - Las claves usadas deben existir en el array name de este bloque.

1. Los valores de la macro serán un JSON de tipo array con los distintos valores seleccionados en el árbol que compartan la misma macro. Por ejemplo:

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

- children:
  - Opcional.
  - Define los nombres de otros bloques de configuración del mismo fichero INI a partir de los cuales se obtendrán los elementos hijos de este nivel del árbol.
  - No se puede hacer referencia a bloques de configuración del mismo INI que hayan sido usados en un nivel superior del árbol para evitar bucles infinitos.
  - Las claves usadas deben existir en el array name de este bloque.

Por ejemplo, esta sería una forma de definir varios elementos de un árbol:

```
[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
```

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

```
[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
```

Selectores con datos de Pandora FMS

Dentro de los campos que se muestran en los formularios, los selectores simples y múltiples tienen la posibilidad de utilizar datos de la propia aplicación Pandora FMS.

Dependiendo de los datos que se quieran, los valores que se almacenarán y los que se usarán en las ejecuciones variaran:

- agent_groups:
  - En el selector usará los nombres de los grupos de agentes.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de grupo existe, y si existe usará su nombre como valor. Si no irá vacío.
- agents:
  - En el selector usará los alias de los agentes.
  - Almacenará sus nombres como datos.
  - En las ejecuciones comprobará si el agente existe, y si existe usará su nombre como valor. Si no irá vacío.
- module_groups:
  - En el selector usará los grupos de módulos.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de grupo existe, y si existe usará su nombre como valor. Si no irá vacío.
- modules:
  - En el selector usará los nombres de módulos.
  - Almacenará sus nombres como datos.
  - En las ejecuciones comprobará si existe un módulo con ese nombre, y si existe usará su nombre como valor. Si no irá vacío.
- module_types:
  - En el selector usara las descripciones de tipos de módulos.
  - Almacenará sus nombres como datos.
  - En las ejecuciones usará sus nombres como valores.
- tags:
  - En el selector usará los tags.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de tag existe, y si existe usará su nombre como valor. Si

no irá vacío.

- status:
  - En el selector usará los nombres de los estados de módulos.
  - Almacenará sus constantes como datos.
  - En las ejecuciones usará las contantes como valores:
    - 0: Normal
    - 2: Warning
    - 1: Critical
    - 3: Unknown
    - 5: Not init
- alert_templates:
  - En el selector usará los nombres de las plantillas de módulos.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de plantilla existe, y si existe usará su nombre como valor. Si no irá vacío.
- alert_actions:
  - En el selector usará los nombres de las acciones de alertas.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de acción existe, y si existe usará su nombre como valor. Si no irá vacío.
- interval:
  - En el selector usará el selector de intervalos.
  - Almacenará el tiempo en segundos.
  - En las ejecuciones usará el tiempo en segundos como valor.
- credentials.custom:
  - En el selector usará el selector de credenciales custom.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de credencial existe, y si existe usará un base64 de un JSON con los datos de las credenciales como valor. Si no irá vacío.
  - El JSON para este tipo de credenciales tendrá este formato:

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

- credentials.aws:
  - En el selector usará el selector de credenciales AWS.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de credencial existe, y si existe usará un base64 de un JSON con los datos de las credenciales como valor. Si no irá vacío.
  - El JSON para este tipo de credenciales tendrá este formato:

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

- credentials.azure:
  - En el selector usará el selector de credenciales Microsoft Azure.
  - Almacenará sus IDs como datos.
  - En las ejecuciones comprobará si el ID de credencial existe, y si existe usará un base64 de un JSON con los datos de las credenciales como valor. Si no irá vacío.

    - El JSON para este tipo de credenciales tendrá este formato:

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

- credentials.gcp:
    - En el selector usará el selector de credenciales Google Cloud Platform.
    - Almacenará sus IDs como datos.
    - En las ejecuciones comprobará si el ID de credencial existe, y si existe usará un base64 de un JSON con los datos de las credenciales como valor. Si no irá vacío.
    - El JSON para este tipo de credenciales tendrá este formato:

```
{
    "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"
}
```

- credentials.sap:
    - En el selector usará el selector de credenciales SAP.
    - Almacenará sus IDs como datos.
    - En las ejecuciones comprobará si el ID de credencial existe, y si existe usará un base64 de un JSON con los datos de las credenciales como valor. Si no irá vacío.
    - El JSON para este tipo de credenciales tendrá este formato:

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

- credentials.snmp:
    - En el selector usará el selector de credenciales SNMP.
    - Almacenará sus IDs como datos.
    - En las ejecuciones comprobará si el ID de credencial existe, y si existe usará un base64 de un JSON con los datos de las credenciales como valor. Si no irá vacío.
    - El JSON para este tipo de credenciales tendrá este formato:

```
{
    "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:
    - En el selector usará el selector de credenciales WMI.
    - Almacenará sus IDs como datos.
    - En las ejecuciones comprobará si el ID de credencial existe, y si existe usará un base64 de un JSON con los datos de las credenciales como valor. Si no irá vacío.
    - El JSON para este tipo de credenciales tendrá este formato:

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

- os:
    - En el selector usará los nombres de los OS.
    - Almacenará los IDs como datos.
    - En las ejecuciones comprobará si el ID de OS existe, y si existe usará su nombre como valor. Si no irá vacío.

Fichero base

El siguiente ejemplo puede usarse como base para crear ficheros discovery_definition.ini, ya que incluye todos los posibles bloques y parámetros explicados anteriormente con comentarios para cada caso:

```
;-------------------------------------;
; 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_
```

Ejemplo

A continuación mostramos un ejemplo de fichero discovery_definition.ini y como se mostraría su formulario en la consola de Pandora FMS:

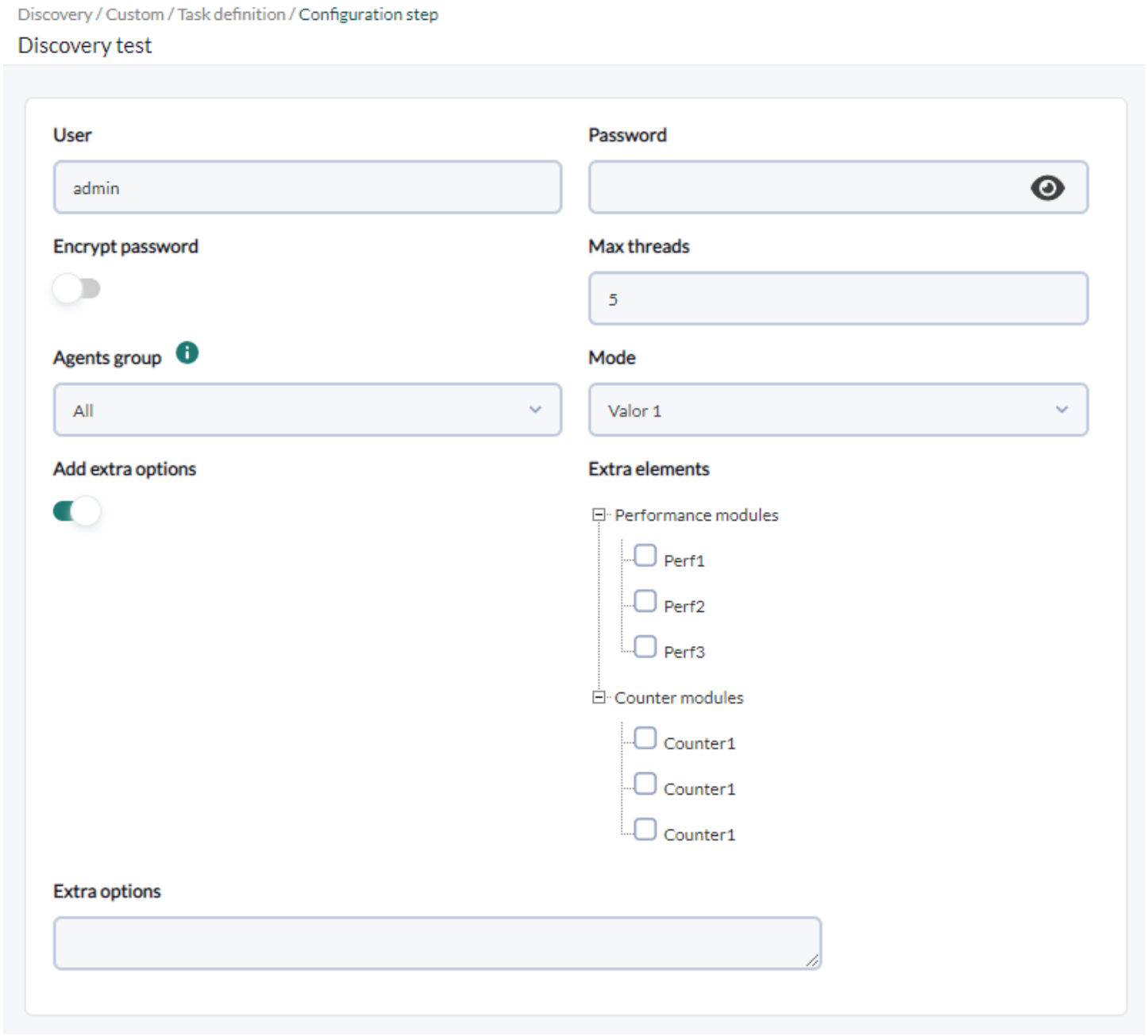
```
[discovery_extension_definition]

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_
```

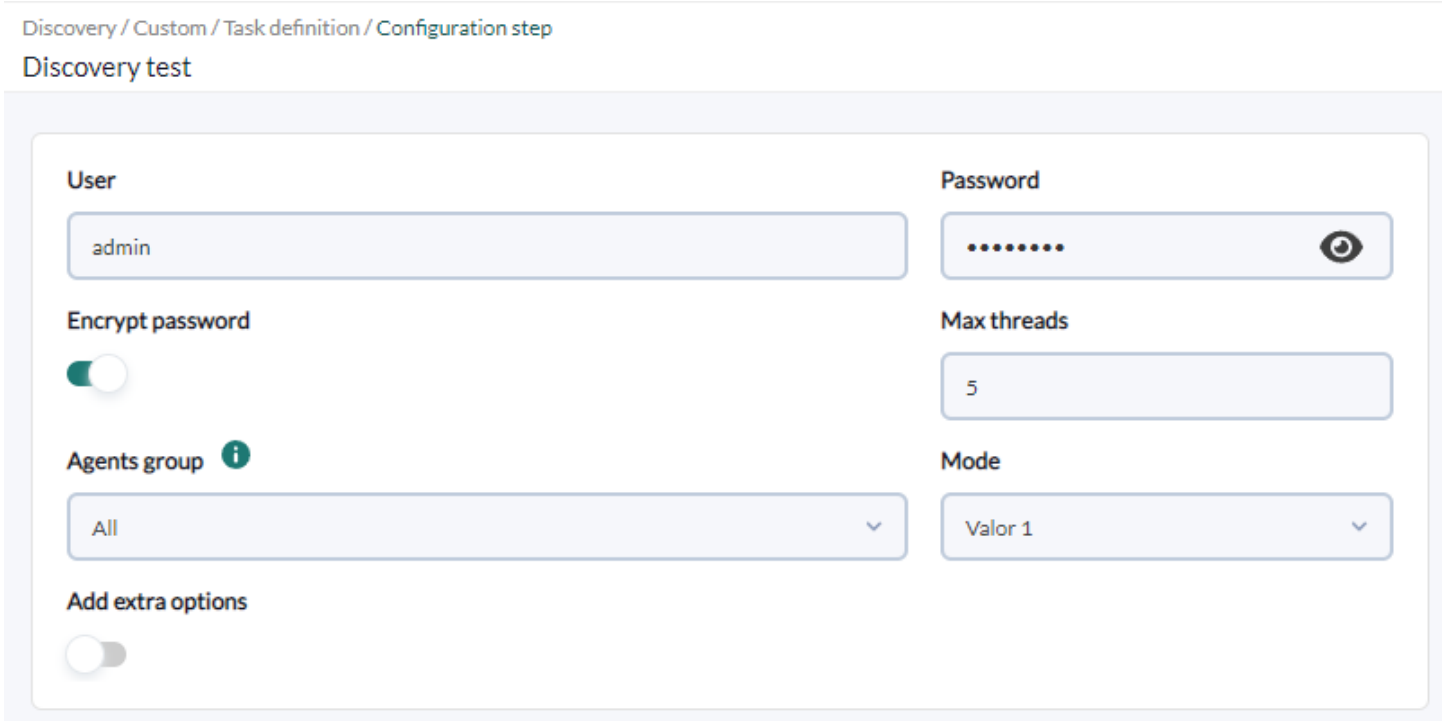
```
value[1] = c1

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

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

Con la definición de arriba, al crear una tarea para la aplicación, este sería el formulario que se mostraría:



Los campos “Extra elements” y “Extra options” se ocultan en el momento en que “Add extra

options” queda sin seleccionar, y el campo “Password” encriptaría su valor al haber seleccionado el campo “Encrypt password”:



En el momento de ejecutar esta tarea por el servidor, genería un fichero temporal (por ejemplo `/var/spool/pandora/data_in/discovery/tmp/6d7fce9fee471194aa8b5b6e47267f03`) cuyo contenido podría ser:

```
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
```

Y el cual usaría durante la ejecución, que sería similar a esta:

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

Scripts y ejecutables

El servidor de Pandora FMS se encargará de ejecutar los *scripts* y ejecutables definidos en el fichero `discovery_definition.ini` y determinará el resultado y sumario de la ejecución de cada tarea en base a la salida y código de error de cada una de las ejecuciones realizadas.

Para determinar el estado comprobará el código de error devuelto por cada una de las ejecuciones que realice. Si al menos una de las ejecuciones devuelve un código de error distinto de `0`, el estado de la tarea se considerará failed.

Para mostrar el sumario de la tarea, para cada una de las ejecuciones que realice el servidor de Pandora FMS recogerá su salida, tanto la salida estándar (STDOUT) como la salida de errores (STDERR). Normalmente se esperará obtener una salida con el siguiente formato JSON mínimo:

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

En la consola la clave summary se leera como una tabla de dos columnas, considerando sus claves como los elementos a la izquierda y sus valores como los elementos a la derecha.

La clave info se leera como información adicional, añadiendo una fila más a la tabla de sumario.

Cualquier otra clave del JSON o en caso de de no devolver un JSON o uno que no cumpla esa estructura, el resto de información se añadirá en una fila más en la tabla de sumario.

Se generará una tabla de sumario para cada una de las ejecuciones, enumerándolas en base al orden de as ejecuciones realizadas por el servidor. De este modo la tabla “Summary 1” se correspondería con la primera ejecución, la tabla “Summary 2” con la segunda y así sucesivamente.

Esto pretende asegurar que en la consola sea visible en todo momento el resultado de la última ejecución, tanto si ha sido existosa como si ha sido fallida.

Finalmente, a parte de las acciones que lleven a cabo los scripts o ejecutables, el servidor de Pandora FMS podrá procesar agentes y módulos a partir de la salida de las ejecuciones. Para ello el servidor de Pandora FMS esperará recibir en la salida JSON una clave adicional monitoring_data con la información de cada uno de los agentes y módulos que deba procesar, pero los datos de esta clave no se almacenarán en el sumario de ejecución.

De esta forma, el formato de salida JSON esperado para los *plugins* de Discovery es este:

```
{
    "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"
                }
            ]
        }
    ]
}
```