Module de source de type cfg-file-import

Sommaire

Description

Cette source permet d'importer de nouveaux éléments via des fichiers .cfg.

La syntaxe d'import de ces fichiers est la même que la syntaxe Nagios, plus des ajouts de Shinken Entreprise:

- Il est donc possible d'importer des fichiers de configuration Nagios dans Shinken, et ainsi migrer sa configuration Nagios vers Shinken Entreprise facilement.
- Les fichiers de configuration Shinken Framwork sont évidemment supportés et importables dans Shinken Entreprise.

Activation du collecteur

Vous pouvez essayer ce type de source soit en activant **vos propres collecteurs de type cfg-file-import** ou en activant le collecteur cfg-file-nagios présents par défaut.

Ajouter un collecteur d'import de fichier .cfg

Vous pouvez avoir plusieurs sources du type cfg-file-import, pour par exemple trier les éléments.

Choisissez un nom pour ce nouveau collecteur.

- Pour l'exemple, nous allons l'appeler "Mon-Collecteur-de-Fichier".
- Remplacer dans l'exemple le mot "Mon-Collecteur-de-Fichier" par le nom que vous aurez choisi.

Faites les opérations suivantes :

- Copier le fichier de définition de la source d'exemple : */etc/shinken-user-example/configuration/daemons/synchronizers/sources/cfg-file-import/cfg-file-import-example.cfg* dans le répertoire de définition des sources **/etc/shinken/sources/** et modifier son nom.
  (Exemple : /etc/shinken/sources/collector__cfg-file-import__Mon-Collecteur-de-Fichier.cfg)

```
cp  /etc/shinken-user-example/configuration/daemons/synchronizers/sources/cfg-file-import/cfg-file-import-example.cfg /etc/shinken/sources/collector__cfg-file-import__Mon-Collecteur-de-Fichier.cfg
```

- Mettre les droits d'écriture au fichier

```
chmod -R 664 /etc/shinken/sources/collector__cfg-file-import__Mon-Collecteur-de-Fichier.cfg
```

- Ouvrer ce fichier (collector__cfg-file-import__Mon-Collecteur-de-Fichier.cfg) :
  - Modifier la ligne source_name en remplaçant le nom par défaut "**cfg-file-example**" par le nom que vous avez choisi "**Mon-Collecteur-de-Fichier**".

```
source_name                Mon-Collecteur-de-Fichier
```

  - Modifier le chemin du fichier de définition vers celui présent dans le dossier de votre source

```
cfg_path     /etc/shinken-user/source-data/source-data__Mon-Collecteur-de-Fichier
/definition__Mon-Collecteur-de-Fichier.cfg
```

- Copier le dossier d'exemple de la source dans le répertoire des données des sources **/etc/shinken-user/source-data/** et donner lui un nom. *(Exemple : /etc/shinken-user/source-data/source-data__Mon-Collecteur-de-Fichier)*

```
cp -r /etc/shinken-user-example/configuration/daemons/synchronizers/sources/cfg-file-import/source-
data__cfg-file-import__example /etc/shinken-user/source-data/source-data__Mon-Collecteur-de-Fichier/
```

- Renommer le fichier de définition des chemins de la source pour lui donner le nom de votre source

```
mv /etc/shinken-user/source-data/source-data__Mon-Collecteur-de-Fichier/definition-cfg-file-import-
example.cfg /etc/shinken-user/source-data/source-data__Mon-Collecteur-de-Fichier/definition__Mon-
Collecteur-de-Fichier.cfg
```

  Par défaut, *definition__Mon-Collecteur-de-Fichier.cfg* indique que le collecteur importera tous les fichiers qui sont dans ce répertoire et les sous-répertoires. Mais vous pouvez le modifier cela (voir la page Importer ses propres fichiers CFG).

- Une fois que les fichiers ont été édités, vérifiez que les droits sont bons (*droit de lecture pour l'utilisateur Shinken*) .
  Si ce n'est pas le cas, effectuez la commande suivante :

```
chmod -R 764 /etc/shinken-user/source-data/source-data__Mon-Collecteur-de-Fichier
chown -R shinken:shinken  /etc/shinken-user/source-data/source-data__Mon-Collecteur-de-Fichier
```

- Ajouter le nom de la nouvelle source au Synchronizer en modifiant le paramètre **sources** du fichier **/etc/shinken/synchronizers /synchronizer-master.cfg**.

```
define synchronizer {
    [...]
    sources                  Source 1, Source 2, Source 3, Mon-Collecteur-de-Fichier
    [...]
}
```

- Redémarrez le Synchronizer pour qu'il puisse prendre en compte cette nouvelle source

```
service shinken-synchronizer restart
```

Activer le collecteur d'import de fichier cfg-file-nagios livré par défaut

Par défaut, l'installation ou la mise à jour de Shinken Entreprise va mettre à disposition une définition de collecteur d'import de fichier Nagios au format .cfg.

- La configuration de ce collecteur se trouve par défaut dans le fichier : **/etc/shinken/sources/cfg-file-nagios.cfg**
- Le collecteur **cfg-file-sample** s'active comme les autres sources, c'est-à-dire en modifiant le fichier **/etc/shinken/synchronizers /synchronizer-master.cfg** (*ou le .cfg que vous utilisez pour définir les options du Synchronizer*)
  - Ce collecteur est déjà présent dans la liste des sources livrées par défaut, il n'y a donc rien à modifier.
  - S'il n'y est pas (*car modifié*), vous pouvez le rajouter pour qu'il soit de nouveau actif.

Exemple :

```
sources     source1,source2,source3,cfg-file-nagios
```

- Redémarrez le Synchronizer pour qu'il puisse prendre en compte cette nouvelle source

```
service shinken-synchronizer restart
```

Configuration

Voici le détail du fichier de configuration de la source qui se trouve (*suivant la procédure choisie*) :

- Soit dans le fichier que vous venez de créer en ajoutant le module (*par exemple /etc/shinken/sources/collector__cfg-file-import__Mon-Collecteur-de-Fichier.cfg*).
- Soit dans le fichier **/etc/shinken/sources/cfg-file-nagios.cfg**.

Exemple de fichier de configuration

Vous trouverez un exemple dans **/etc/shinken-user-example/configuration/daemons/synchronizers/sources/cfg-file-import/cfg-file-import-example.cfg**

```
#===============================================================================
# cfg-file-import-example
#===============================================================================
# Daemons that can load this source:
# - synchronizer
# This source import the cfg-config sample from Shinken Enterprise update.
#===============================================================================

define source {

    #======== Source identity ==============
    # Source name. Must be unique
    source_name                             cfg-file-import-example

    # Module type (to load module code). Do not edit.
    module_type                             cfg-file-import

    # description: display a description on the interface for this source
    description                             This source is about loading cfg files compatible with Shinken

    # enabled: is this source enabled or not
    enabled                                 1


    #======== Import interval and order ====
    # import_interval:  in minutes, what is the schedule import interval for this source.
    # note: 0 = don't schedule this source, will run only if an administrator launch it from the interface
    import_interval                         5

    # order: source order for a source imply if a source is before an another source when
    # merging data
    order                                   1


    #======== Location =====================
    # Location of the source definition file. This file will be used to define where the .cfg files are to
be imported.
    cfg_path                                        /etc/shinken-user/source-data/source-data-cfg-file-
import-example/definition-cfg-file-import-example.cfg


    #======== Synchronization keys =========
    # The list of properties to be used as sync_keys in addition to the item name. Properties not managed by
Shinken can be added here.
    # properties_used_as_synckey            address


    #======== Properties not stored ========
    # Properties which can be defined in the items from the source but which Shinken will not import.
    # not_stored_properties


    #======== SE_UUID in cfg file ==========
    # With this option if item in cfg hasn't a SE_UUID, the source will ask the synchronizer to search a
match in staging or in working area. If found, it will be inserted in the file.
    # update_cfg_with_staging_se_uuid       0

    # With this option if item in cfg hasn't a SE_UUID, the source will create it and write in the file
    # create_and_write_SEUUID_in_file       0  }
```

Détails des sections composant le fichier de configuration

Identification de la source

```
...
    #======== Source identity ==============
    # Source name. Must be unique
    source_name                             cfg-file-import-example

    # Module type (to load module code). Do not edit.
    module_type                             cfg-file-import

    # description: display a description on the interface for this source
    description                             This source is about loading cfg files compatible with Shinken

    # enabled: is this source enabled or not
    enabled                                 1
...
```

Il est possible de définir plusieurs instances de module de type **`cfg-file-import`** dans votre architecture Shinken.

- Chaque instance devra avoir un nom unique.

Nom	Type	Unité	Défaut	Commentaire
`source_name`	Texte	---	**cfg-file-import-example**	**Nous vous conseillons de choisir un nom en fonction de l'utilisation du module pour que votre configuration soit simple à maintenir.** Chaîne de caractères composée de lettres, de chiffres et des caractères _ et - . • Doit être unique. • D'une longueur maximum à 40 caractères. • Ne dois **pas contenir les caractères** *?*, *&*, *"*, *'*, *$*, */*, *#*, **, *;*
`module_type`	Texte	---	**cfg-file-import**	Ne peut être modifié
`description`	Texte	---		Contient une description de la source qui vous permet de retrouver l'objectif de cette source
`enabled`	Booléen	---	**0**	Permet d'activer/désactiver la source. **Remarque** *Si vous activer/desactiver la source depuis l'interface (page d'accueil), le fichier .cfg sera mis à jour.*

Interval d'import et ordre de la source

```
...
    #======== Import interval and order ====
    # import_interval:  in minutes, what is the schedule import interval for this source.
    # note: 0 = don't schedule this source, will run only if an administrator launch it from the interface
    import_interval                         5

    # order: source order for a source imply if a source is before an another source when
    # merging data
    order                                   1
...
```

Il est possible de définir un intervalle d'import afin que la source s'importe régulièrement.

Nom	Type	Unité	Défaut	Commentaire
`import_interval`	Entier positif	minute	**0**	Délai écoulé entre les imports automatiques de la source. Si 0, l'import de la source ne sera jamais exécuté automatiquement. (uniquement manuellement)
`order`	Entier positif	---	**2**	L'ordre de la source dans l'interface de configuration (*A un impact dans la fusion des données lors des imports de sources*). Voir la page Le Synchronizer pour plus d'information au sujet des fusions. **Remarque** *Si vous changez l'ordre depuis l'interface (page d'accueil), le fichier .cfg sera mis à jour.*

Emplacement des fichiers de configuration

```
...
    #======== Location =====================
    # Location of the source definition file. This file will be used to define where the .cfg files are to
be imported.
    cfg_path                                        /etc/shinken-user/source-data/source-data-cfg-file-
import-example/definition-cfg-file-import-example.cfg
...
```

Afin de récupérer les éléments Shinken, contenus dans des fichiers .cfg, il faut spécifier un unique fichier de définition de la source qui contiendra la liste des chemins vers les fichiers à importer. (Voir la page Importer ses propres fichiers CFG)

Nom	Type	Unité	Défaut	Commentaire
`cfg_path`	Texte	---	**---**	Emplacement du fichier de définition de la source. Ce fichier servira à définir où sont les fichiers .cfg à importer.

Clés de synchronisation (sync_key)

```
...
    #======== Synchronization keys =========
    # The list of properties to be used as sync_keys in addition to the item name. Properties not managed by
Shinken can be added here.
    # properties_used_as_synckey            address
...
```

Si le paramètre properties_used_as_synckey :

- N'est pas défini,
  - sa valeur par défaut est à *address*. Tout élément ayant la propriété *address* aura la valeur de cette propriété ajoutée dans les clés de synchronisation.
- Est vide,
  - aucune propriété ne sera ajoutée dans les clés de synchronisation.
- Est redéfini à *display_name*
  - seule la valeur de cette propriété sera ajoutée dans les clés de synchronisation.
  - Remarque : plusieurs propriétés peuvent être définies comme clef de synchronisation (cela dépendra de vos besoins)

```
properties_used_as_synckey host_name, display_name, address
```

Défini la liste des propriétés qui seront utilisées pour générer les clés de synchronisation.

À noter : On ne peut pas supprimer le nom et le _SE_UUID, mais on peut les compléter.

Nom	Type	Unité	Défaut	Commentaire
`properties_used_as_synckey`	Texte	---	**address**	Défini la liste de propriétés qui seront utilisées en plus du nom et du _SE_UUID de l'élément pour générer les clés de synchronisation (sync_key). Ce paramètre est optionnel. Si ce paramètre n'est pas présent, sa valeur par défaut vaut propriété "address". S'il est défini à vide, la propriété "address" ne sera pas utilisée comme synckey.

Ajout des _SE_UUID dans les fichiers de configuration

```
...
    #======== SE_UUID in cfg file ==========
    # With this option if item in cfg hasn't a SE_UUID, the source will ask the synchronizer to search a
match in staging or in working area. If found, it will be inserted in the file.
    # update_cfg_with_staging_se_uuid       0

    # With this option if item in cfg hasn't a SE_UUID, the source will create it and write in the file
    # create_and_write_SEUUID_in_file       0
...
```

Si un élément importé dans un des fichiers .cfg ne contient pas pas de _SE_UUID, Shinken peut lui en générer un ou le chercher depuis les éléments existant dans staging. Il est alors possible de faire en sorte que cet _SE_UUID soit retranscrit dans le fichier .cfg grâce à ce paramètres.

Nom	Type	Unité	Défaut	Commentaire
`update_cfg_with_staging_se_uuid`	Booléen	---	**0**	Ce paramètre permet de retrouver le _SE_UUID de l'élément s'il a déjà été importé ou s'il se mélange avec un autre élément. Si un élément existe en staging, le _SE_UUID est ajouté dans le fichier .cfg. Cela permet de modifier le nom ou les autres propriétés constituant les sync_keys dans le .cfg, sans perdre le mélange avec l'élément de staging. • 0 : Ne recherche pas le _SE_UUID pour l'écrire dans le fichier .cfg • 1 : Recherche le _SE_UUID des éléments importés depuis staging et les écrits dans les fichiers .cfg
`create_and_write_SEUUID_in_file`	Booléen	---	**0**	Ce paramètre crée un _SE_UUID pour chaque élément importé par la source, avant que les éléments ne soient mélangés. Cela empêche les éléments d'être mélangés avec d'autres éléments. Le _SE_UUID est ajouté dans le fichier .cfg. • 0 : Ne crée pas le _SE_UUID • 1 : Crée et ajoute les _SE_UUID des éléments importés dans les fichiers .cfg

Propriétés non récupérées

```
...
    #======== Properties not stored ========
    # Properties which can be defined in the items from the source but which Shinken will not import.
    # not_stored_properties
...
```

Il est possible de définir des propriétés que la source ne devra pas récupérer.

Dans le cas de cette source, il se peut que vous ne soyez pas l'auteur des fichiers cfg et que vous ne vouliez pas importer certaines propriétés.

Nom	Type	Unité	Défaut	Commentaire
`not_stored_properties`	Texte	---	**---**	Empêche la récupération de certaines propriétés des éléments récoltés