Mapping des informations collectées des champs du serveur VMWare vers les propriétés et les données Shinken

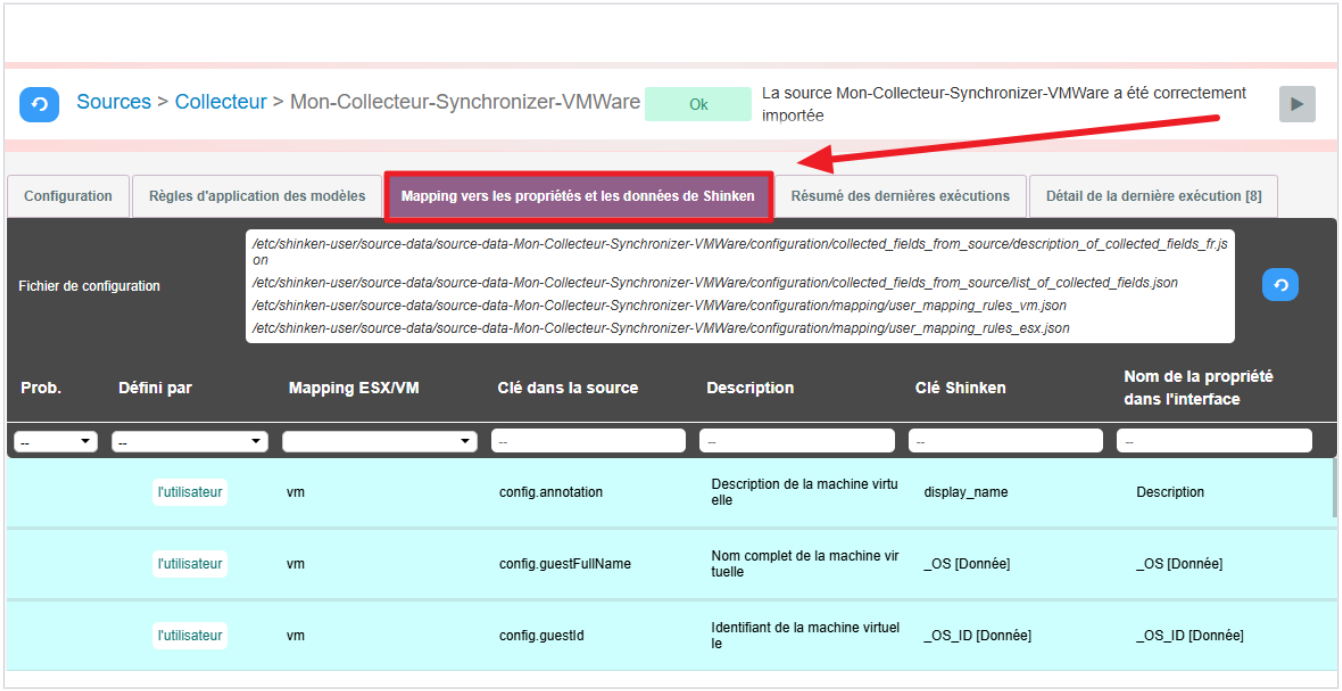
Sommaire

Concept

Ces règles permettent, lors de l'import, de mettre en relation un champ de l'API VMWare avec une propriété ou une donnée de Shinken en fonction du type de l'élément (*hôte, modèle d'hôte, etc ...*).

- Elle s'applique pour tous les types d'éléments présents dans l'interface de Shinken.

Elles peuvent être visualisées depuis l'onglet "Mapping vers les propriétés et les données de Shinken" de l'interface de la source.



Définition des règles de mapping

Pour ajouter une règle utilisateur d'application des modèles, il faut éditer le chemin suivant :

```
/etc/shinken-user/source-data/source-data-[nom de la source]/configuration/mapping/user_mapping_rules_vm.json
/etc/shinken-user/source-data/source-data-[nom de la source]/configuration/mapping/user_mapping_rules_esx.
json
```

Exemple de chemin pour la source livrée par défaut :

/etc/shinken-user/source-data/source-data-synchronizer-collector-vmware/configuration/mapping/user_mapping_rules_vm.json
/etc/shinken-user/source-data/source-data-synchronizer-collector-vmware/configuration/mapping/user_mapping_rules_esx.json

Comme pour les règles d'application des modèles, les règles de mapping sont écrites au format JSON.

La source VMWare ne concerne que les hôtes, donc les fichiers sources commenceront toujours par "hosts".

Elle doit respecter le format suivant :

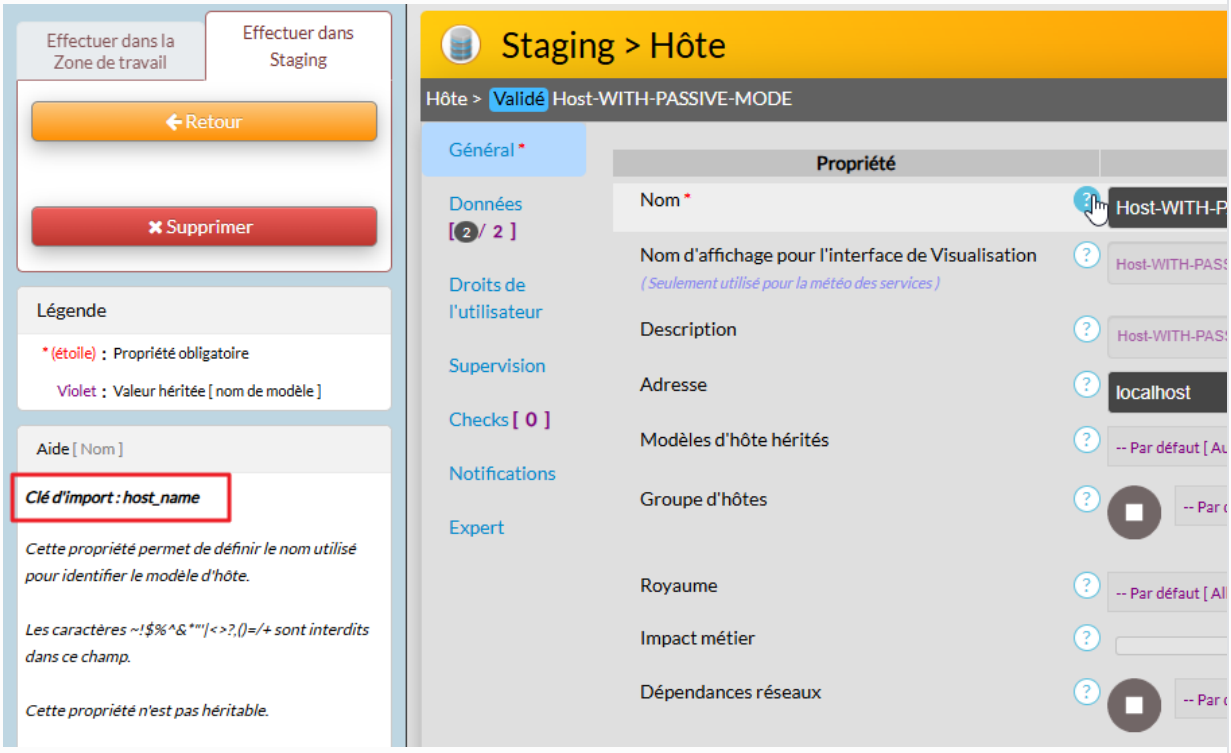
- ***Nom du champ** de l'API VMWare* : ***nom** de la **propriété** ou de la **donnée** Shinken*

Définition : propriété

Une propriété est une information nécessaire au moteur Shinken.

- Pour définir dans le mapping, une propriété Shinken, il faut utiliser le nom de la clé d'import de Shinken (*visible dans l'aide des pages d'édition des éléments*),
- La clé d'import de l'uuid des éléments (*pas trouvable dans l'interface*) est "uuid"

Clé d'import dans Shinken (visible dans l'aide)



Définition : donnée

Une donnée est une valeur qui va être ajoutée en complément des propriétés.

- Pour définir dans le mapping une donnée Shinken :
  - Il est recommandé d'utiliser la syntaxe "**DATA**" : "DATA(NOM_DE_LA_DONNEE_SHINKEN)".
  - Il faut que le nom de la donnée Shinken :
    - soit en majuscules
    - ne doit pas commencer par un underscore.
    - contient que des lettres ou des chiffres ou un underscore (_) ou un tiret (-).
  - Si la syntaxe utilisant **DATA**(...) est à privilégier ,
    - pour assurer la compatibilité ascendante, il est toujours possible de créer une donnée dans Shinken en lui donnant un nom en majuscule commençant par un underscore (*ex : _NOM_DONNEE_SHINKEN*) et en l'associant au champ récolté par la source.
    - Cette syntaxe est cependant **DÉPRÉCIÉE** .

Forcer la valeur des propriétés

Il est possible d'ajouter le suffixe **[FORCE]** (*sans espace*) à la **clé d'import** d'un élément Shinken afin de forcer la valeur associée (*voir la page* Forcer la valeur des noms des éléments et des propriétés de type liste (comme la propriété des modèles hérités)).

Exemple :

```
{
  "hosts": {
    "name": "host_name[FORCE]",
    "shinken.ipAddress" : "address"
  }
}
```

Lorsqu'une propriété est marquée avec **[FORCE]**, elle est traitée de la manière suivante par le mélange des sources :

- La valeur est considérée comme prioritaire par rapport aux autres sources.
- Si une autre source importe le même objet avec une valeur différente, cette dernière est ignorée.
- En cas de conflit entre plusieurs valeurs forcées, la priorité de la source détermine la valeur conservée.

Ajouter [FORCE] à une clé d'import contenant une liste (*comme la propriété "members" servant à définir les membres d'un groupe d'utilisateurs*) aura pour effet de remplacer complètement la liste des autres sources par celle fournie avec l'option [FORCE]

Liste des clés d'import pouvant être forcées

- Les clés d'import définissant les noms des éléments
  - Exemple : "host_name" pour les Hôtes.
- Et toutes les propriétés pouvant avoir des valeurs multiples : (*voir la page* Éditer les Eléments (hôte, clusters, checks, utilisateurs ...))
  - Exemple : "view_contacts" pour la propriété "Les utilisateurs qui voient l'hôte".

La liste exhaustive de toutes les clé est disponible ici : Liste des clés pouvant utiliser [FORCE]

Exemple de mapping dans le fichier "user_mapping_rules_vm.json"

```
{
  "hosts": {
    "name": "host_name",
    "shinken.ipAddress" : "address"
    "shinken.machine_type": "DATA(MACHINE_TYPE)",
    "config.product.osType": "DATA(OS_TYPE)",
        "shinken.tags": "DATA(TAGS)",
  }
}
```

Il existe 4 règles de mapping définies :

Clé VMWare	Clé Shinken	Description du mapping

name	host_name	La valeur VMWare "name" deviendra le nom de l'hôte (***propriété*** *host_name*).
shinken. ipAddress	address	La valeur VMWare "shinken.ipAddress" sera prise en compte comme adresse pour l'hôte (***propriété*** *address*).
shinken. machine_type	**DATA**(MACHINE_TYPE)	La valeur VMWare "shinken.machine_type" deviendra une **donnée** nommée MACHINE_TYPE sur l'hôte.
config.product. osType	**DATA**(OS_TYPE)	La valeur VMWare "config.product.osType" deviendra une **donnée** nommée OS_TYPE sur l'hôte.
shinken.tags	**DATA**(TAGS)	La valeur VMWare "shinken.tags" deviendra une **donnée** nommée TAGS sur l'hôte. • Vus que la valeur de shinken.tags est une liste dans l'API de VMWare, la valeur de la donnée TAGS vaudra la liste des tags séparés par des virgules. • Exemple : La valeur de shinken.tags **['Linux','Bordeaux','France']** deviendra la valeur **Linux,Bordeaux, France** dans TAGS

- Il est possible d'ajouter des commentaires dans ce fichier.
  Toutes les lignes qui commencent par le caractère # seront considérées comme des commentaires.
- La clé d'import Shinken n'est pas le nom affiché dans les pages des éléments, mais le nom de la propriété lors de l'import dans Shinken.
  Pour trouver le nom d'import d'une propriété, il est possible d'ouvrir l'aide d'une propriété dans une page d'édition d'un élément (*voir la page Éditer les Eléments (hôte, clusters, checks, utilisateurs ...) (paragraphe "Informations contextuelles")*).
- Retrouvez la liste complète des champs VMware collectés par la source sur la page Liste des champs collectés auprès des VCenters ou ESXis .
- Les listes de l'API de VMWare (*Par exemple : shinken.tags, shinken.tag_categories*) sont transformées en liste exploitable par Shinken.
  - Exemple : La valeur **['Linux','Bordeaux','France']** deviendra **Linux,Bordeaux,France.**

La propriété shinken _SE_UUID ne peut pas être mappée.

Opérateur pour les régles de mapping

Les opérateurs des règles de mapping permettent de transformer les données collectées afin de les adapter au format de Shinken et aux besoins de la supervision.

Opérateur VALUES pour les listes

La source VMWare peut fournir des champs dont les valeurs sont des listes (*par exemple le champ "shinken.tags"*).

Pour faire correspondre une valeur de liste dans un champ de Shinken, il est possible d'utiliser l'opérateur : "**VALUES**" .

Exemple

Une VM possède la liste de tags suivante : LINUX, FR, DATACENTER_42

Le mapping suivant :

Dans user_mapping_rules_vm.json

```
{
  "hosts": {
     "shinken.tags>VALUES(0)": "DATA(MACHINE_TYPE)",
     "shinken.tags>VALUES(=DATACENTER)": "DATA(DATACENTER)",
   }
}
```

va permettre de créer un hôte avec comme données :

Extrait d'Hôte dans Shinken

```
_MACHINE_TYPE LINUX
_DATACENTER   DATACENTER_42
```

- MACHINE_TYPE contient la **première valeur** (*indice 0*) de la liste.
- DATACENTER se voit affecter les valeurs qui contiennent au moins **DATACENTER** .

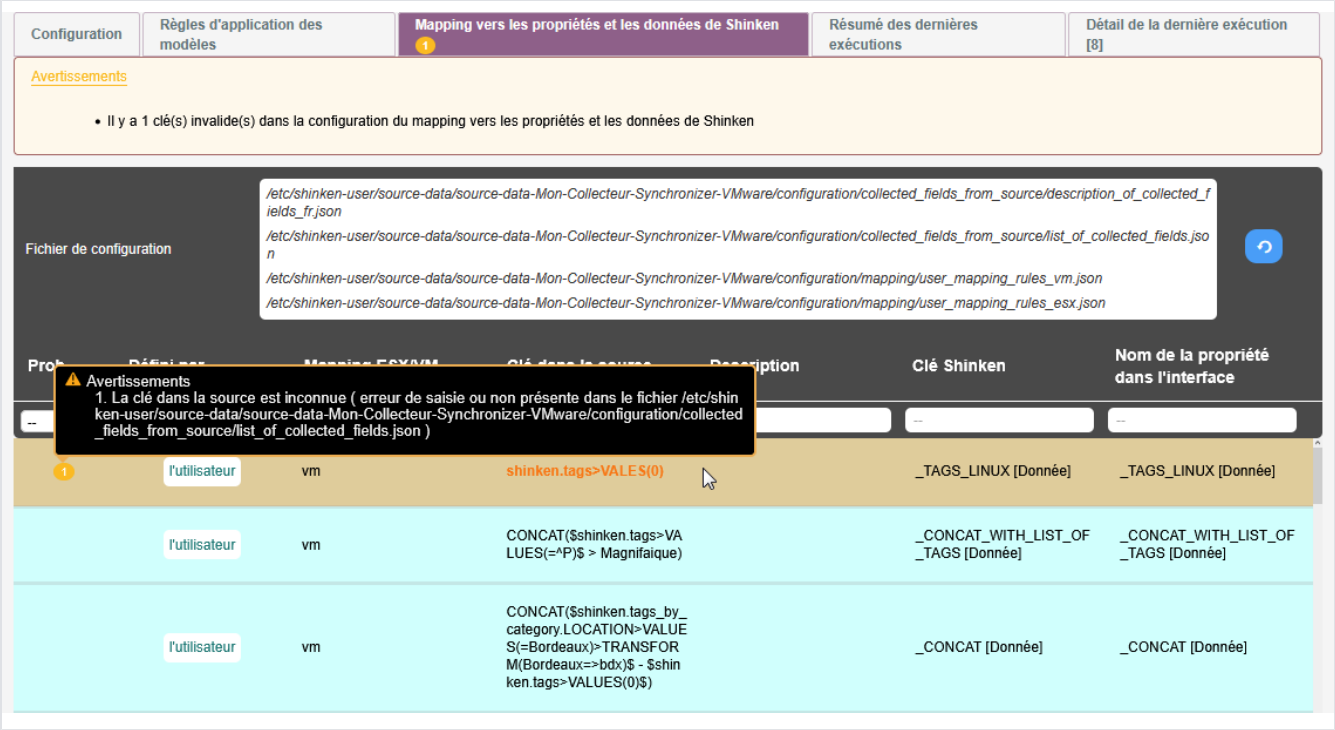
Définition de l'utilisation de VALUES

La syntaxe de la condition dans "VALUES(condition)" est la suivante :

Signification	Syntaxe	Description	Exemple
Index	nombre	Récupère l'élément à l'index indiqué. L'index doit être un nombre appartenant à l'ensemble des entiers naturels, incluant 0. Cette règle admet une seule exception afin de rendre possible de choisir le dernier élément d'une liste, en utilisant l'index -1. L'index 0 récupérera le premier élément de la liste, 1 le deuxième, 2 le troisième, etc...	`VALUES(0)`
Contient	**=** *texte*	La condition signifie que l'élément choisi **doit CONTENIR** le texte.	`VALUES (=PROD)`
Commence par	**=^** *texte*	Si l'expression commence par le caractère "^" , la condition signifie que l'élément choisi **doit COMMENCER** par le texte.	`VALUES (=^PROD)`
Termine par	= *texte* **$**	Si l'expression termine par le caractère "$" la condition signifie que l'élément choisi **doit TERMINER** par le texte.	`VALUES (=PROD$)`
Est strictement égal	= **^** *texte* **$**	Si l'expression commence par "^" ET termine par "$" , la condition signifie que l'élément choisi **doit être l'expression EXACTE.**	`VALUES (=^PROD$)`
Inversion	= **!^** *texte* **$**	Ajouter un " **!** " avant une autre expression provoque l'effet inverse de celle-ci : • = **!** texte --> L'élément choisi ne contient pas "texte". • = **!** **^** texte --> L'élément choisi ne commence pas par "texte". • = **!** texte **$** --> L'élément choisi ne termine pas par "texte". • = **!^** texte **$** --> L'élément choisi n'est pas strictement égal à "texte".	`VALUES(=!PROD)`

- Les conditions sur le texte ne font pas la distinction entre les majuscules et les minuscules.
  - Exemple : la règle suivante pourra récupérer la valeur "Bordeaux.

    `shinken.tags>`**`VALUES`**`(=BoRdeaUx)`
- Si plusieurs valeurs valident la condition, "**VALUES**" retourne une liste de valeurs.
- Si une liste est vide (*Exemple : aucun tags attaché à cette VM*), **VALUES** (*quels que soient les paramètres*) ne retourne rien.
- Si une faute de frappe est faite sur le mot **VALUES**, l'interface de mapping des sources indique une clé inconnue.



- **VALUES**(X) avec X ayant une valeur supérieure au nombre d'éléments présents dans la liste ne retourne rien (*Exemple : essayer de récupérer le 5ème élément sur une liste de 3 éléments*).

Opérateur TRANSFORM pour les transformations

Grâce à l'opérateur **TRANSFORM**, il est possible de transformer du texte.

- Exemple : L'API VMWare fournit la valeur "Préproduction" et l'on veut transformer cette valeur en "PREPROD" à la place.

La syntaxe de `TRANSFORM` est la suivante :

```
TRANSFORM(transformation1,transformation2,transformation3)
```

- TRANSFORM permet d'indiquer plusieurs transformations, ces dernières sont séparées par des virgules.
- Attention à ne pas mettre d'espace avant ou après une virgule, autrement, il sera pris en compte dans la transformation.
- Il n'y a actuellement pas de limite sur le nombre de transformations autorisées.

Pour écrire une **transformation**, il faut séparer **le texte à transformer** par **"=>**" du **texte désiré** (*désigné ci-après par le mot "séparateur"*) :
TEXTE1=>TEXTE2

Voici un exemple de syntaxe avec 2 transformations :

```
TRANSFORM(BORDEAUX=>BDX,PARIS=>PAR)
```

`TRANSFORM` permet également de **supprimer** du texte en laissant vide la partie à droite du séparateur (`=>`) censée contenir la modification à apporter au texte.

```
TRANSFORM(DEAUX=>,IS=>)
```

Exemple avec une machine possédant les tags suivants :

ENV	• Production_Bordeaux • Production_Paris

Et le mapping suivant :

```
{
  "hosts": {
     "shinken.tags_by_category.ENV>VALUES(=Bordeaux)>TRANSFORM(Production=>Prod,Bordeaux=>bdx)": "DATA
(TRANSFORM_OPERATION)",
  }
}
```

La donnée **TRANSFORM_OPERATION** contiendra donc la valeur suivante :

- Prod_bdx

Lorsque l'opérateur TRANSFORM est appelé sur une liste de valeurs (*Exemple : `shinken.tags`*), il va appliquer la ou les transformations demandées sur chaque élément de la liste individuellement.

Exemple avec une machine possédant les balises (*tags*) suivantes :

OS	• CentOS • Alma • macOS
ENV	• Production_Bordeaux • Preproduction_Bordeaux

Et le mapping suivant :

```
{
  "hosts": {
     "shinken.tags_by_category.ENV>VALUES(=production)>TRANSFORM(Production=>Prod,Preproduction=>Preprod,
Bordeaux=>bdx)": "DATA(TRANSFORM_OPERATION)",
  }
}
```

La donnée **TRANSFORM_OPERATION** contiendra donc les valeurs suivantes :

- Prod_bdx,Preprod_bdx

Maintenant, si l'on veut modifier chacun des tags de la catégorie "OS" pour le faire précéder de "OS_" :

- `CentOS` deviendrait `OS_CentOS`.
- `Alma` deviendrait `OS_Alma`.
- `macOS` deviendrait `OS_macOS`.

Dans ce cas, il faut employer `$CURRENT$`, ce mot-clé prendra la valeur de la balise en train d'être modifiée.

La règle de mapping pour effectuer cette opération est la suivante :

```
"shinken.tags_by_category.OS>TRANSFORM($CURRENT$=>OS_$CURRENT$)": "DATA(TRANSFORMED_OS)"
```

Le caractère $ est utilisé ici pour indiquer à TRANSFORM d'aller chercher les valeurs à faire apparaître, il n'est donc pas possible de l'utiliser dans un autre but que celui-ci.

Opérateur CONCAT pour les concaténations

Il est possible de concaténer plusieurs valeurs de l'API VMWare pour former la valeur d'une propriété d'un élément de Shinken. Cela est possible grâce à l'opérateur : **CONCAT**.

Exemple : Une VM possédant les listes de tags suivantes :

OS	Linux
ENV	Production
LOCATION	Bordeaux

Et les informations doivent être affichées de la manière suivante dans la ***donnée*** nommée _**MACHINE_INFOS** : "**Production - Linux - Bordeaux**"

La syntaxe se fera de la manière suivante dans le fichier de Mapping.

Fichier de mapping

```
{
  "hosts": {
     "CONCAT($shinken.tags.OS>VALUES(0)$ - $shinken.tags.ENV>VALUES(0)$ - $shinken.tags.LOCATION>VALUES
(0)$)": "DATA(MACHINE_INFOS)",
   }
}
```

Le caractère $ est utilisé ici pour indiquer à CONCAT d'aller chercher les valeurs à faire apparaître, il n'est donc pas possible de l'utiliser dans un autre but que celui-ci.

L'opérateur CONCAT peut aussi récupérer les valeurs de n'importe quel nom de champ.

Exemple : Avec le fichier de configuration de la source VMWare suivant :

```
{
  "hosts": {
    "name": "host_name",
    "shinken.ipAddress" : "address"
    "shinken.machine_type": "DATA(MACHINE_TYPE)",
    "CONCAT($name$ - $shinken.ipAddress$)": "DATA(MACHINE_INFOS)",
  }
}
```

et les valeurs :

- "name" valant : "Ubuntu-Preprod"
- "shinken.ipAddress" valant : "192.168.1.42"

La donnée **MACHINE_INFOS** vaudra : "**Ubuntu-Preprod - 192.168.1.42**"

Si un champ est **vide**, alors **`CONCAT`** ne fournira pas de valeur pour ce champ.

Exemple, avec les mêmes données que l'exemple précédent, sauf que "name" n'a aucune valeur, la donnée **MACHINE_INFOS** vaudra : " **- 192.168.1.42**"

Opérateur CONCAT_ON_ALL pour les concaténations sur les listes

Comme pour l'opérateur `TRANSFORM` évoqué plus haut, il est également possible d'appliquer une opération de concaténation sur une liste, à l'aide de l'opérateur **`CONCAT_ON_ALL`** .

De la même manière que lorsque l'opération **`TRANSFORM`** est appliquée sur une liste, il est possible d'utiliser **`$VALUE$`** avec **`CONCAT_ON_ALL`** pour récupérer la valeur de l'élément sur lequel l'opération est en train d'être effectuée.

Exemple :

Une VM possédant les listes de tags suivantes :

OS	• Alma • CentOS

Et le fichier de mapping suivant :

```
{
  "hosts": {
    "name": "host_name",
    "shinken.ipAddress" : "address"
    "shinken.machine_type": "DATA(MACHINE_TYPE)",
  }
}
```

Si on l'on souhaite associer le nom de la machine à chaque OS et le mettre dans une donnée spécifique (*ex MACHINE_OS*), on peut ajouter la règle suivante au fichier de mapping :

```
"shinken.tags_by_category.OS>CONCAT_ON_ALL($name$ has OS: $CURRENT$)": "DATA(MACHINE_OS)"
```

Cela génèrera une liste d'éléments qui sera présente dans la data "_MACHINE_OS" :

```
MyHost has OS Alma,MyHost has OS CentOS
```

Limitation

Il n'est actuellement pas possible d'utiliser le **CONCAT_ON_ALL** dans les autres opérateurs, mais cela sera possible dans une prochaine version.

Par exemple la règle suivante :

```
"CONCAT($shinken.tags_by_category.ENV>CONCAT_ON_ALL($name$ has tag  $CURRENT$)$)": "DATA
(CONCAT_ON_ALL_IN_ANOTHER_RULE)"
```

Ne fonctionne pas

Le caractère $ est utilisé ici pour indiquer à CONCAT_ON_ALL d'aller chercher les valeurs à faire apparaître, il n'est donc pas possible de l'utiliser dans un autre but que celui-ci.

Opérateur MANY_FIELDS_BY_REGEX pour règles de mapping dynamiques (expressions régulières)

L'opérateur **MANY_FIELDS_BY_REGEX** permet de mapper plusieurs champs récoltés par la source à importer dans Shinken à l'aide d'une expression régulière en une seule règle.

Couramment, le mapping entre les champs récoltés par la source et les propriétés de Shinken est exhaustif. Cependant, lors de la récupération des données, il est possible d'avoir un grand nombre de champs différents à importer dans Shinken.

Dans ce cas, le fonctionnement habituel ne suffit plus, c'est pour cela que les règles de mapping peuvent désormais faire appel à des expressions régulières.

Une expression régulière est une formule qui sert à rechercher ou reconnaître des motifs dans du texte.

Par exemple :

- Trouver toutes les clés qui commencent par "EXTRAINFO-" et qui sont en deux parties séparées par un tiret (-).
  - (*Pour une aide à propos des expressions régulières, voir la page* : Guide des expressions régulières)

Pour les utiliser dans le mapping, il faut les écrire sous le format suivant dans la colonne "Propriétés dans Excel" :

```
MANY_FIELDS_BY_REGEX(EXPRESSION)
```

où "EXPRESSION" est l'expression régulière.

Il est possible de récupérer les groupes matchés dans l'expression régulière (*voir la page suivante pour une aide sur les groupes dans les expressions régulières :* Guide des expressions régulières - groupe) pour construire le nom de la donnée dans Shinken. Pour ce faire, il faut utiliser "$" suivi du numéro du regroupement (*$1, $2, $3, etc.*)

Il n'est pas possible d'avoir plus de regroupements dans le nom de la donnée qu'indiqués dans l'expression régulière. On ne peut donc pas faire correspondre la règle "MANY_FIELDS_BY_REGEX(EXTRAINFO-(.*)-(.*))" aux données "DATA($1_$2_$3)" car l'expression ne recherche que deux regroupements.

Exemple d'expression régulière

```
{
  "hosts": {
     "MANY_FIELDS_BY_REGEX(runtime\\.([0-9A-Za-z_]+))": "DATA(RUNTIME_$1)",
  }
}
```

Avec cette expression régulière, la valeur de toutes les colonnes dont le nom :

- Commence par "runtime."
- Est constitué d'une autre partie ayant uniquement des caractères alphanumériques ou un underscore (_).
- Deviendront des données dont les noms reprendront la dernière partie.

- **$1** correspond à la partie capturée par les premières parenthèses.

Exemple :

Les valeurs dans la colonne "runtime. **bootTime** " seront stockées dans des données qui s'appelleront "RUNTIME_ **BOOTTIME** ".

Désactivation d'un mapping

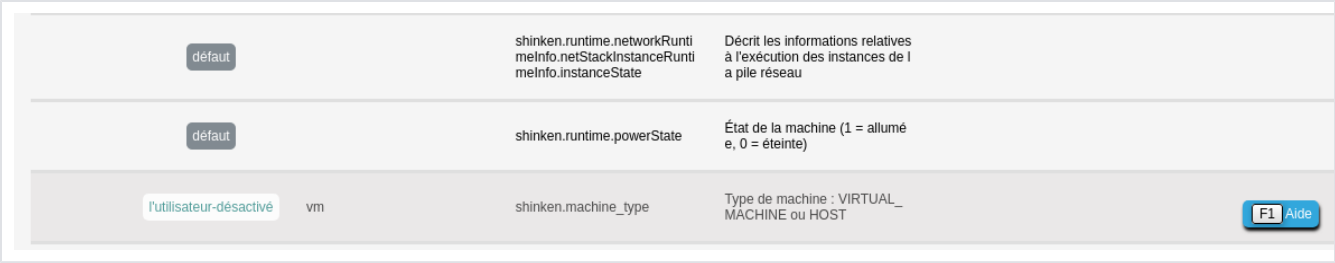
Les mappings peuvent être surchargés en créant un mapping avec le nom de la clé que l'on veut désactiver.

- Pour le champ en question, au lieu de choisir une propriété Shinken *(ou donnée), il faut utiliser le mot-clé "**disable** "*

Exemple:

Désactivation du champ **shinken.machine_type**

```
{
  "hosts":{
    "shinken.machine_type" : "disable",
    "config.product.fullName" : "display_name"
  }
}
```



Visualiser la liste des mappings

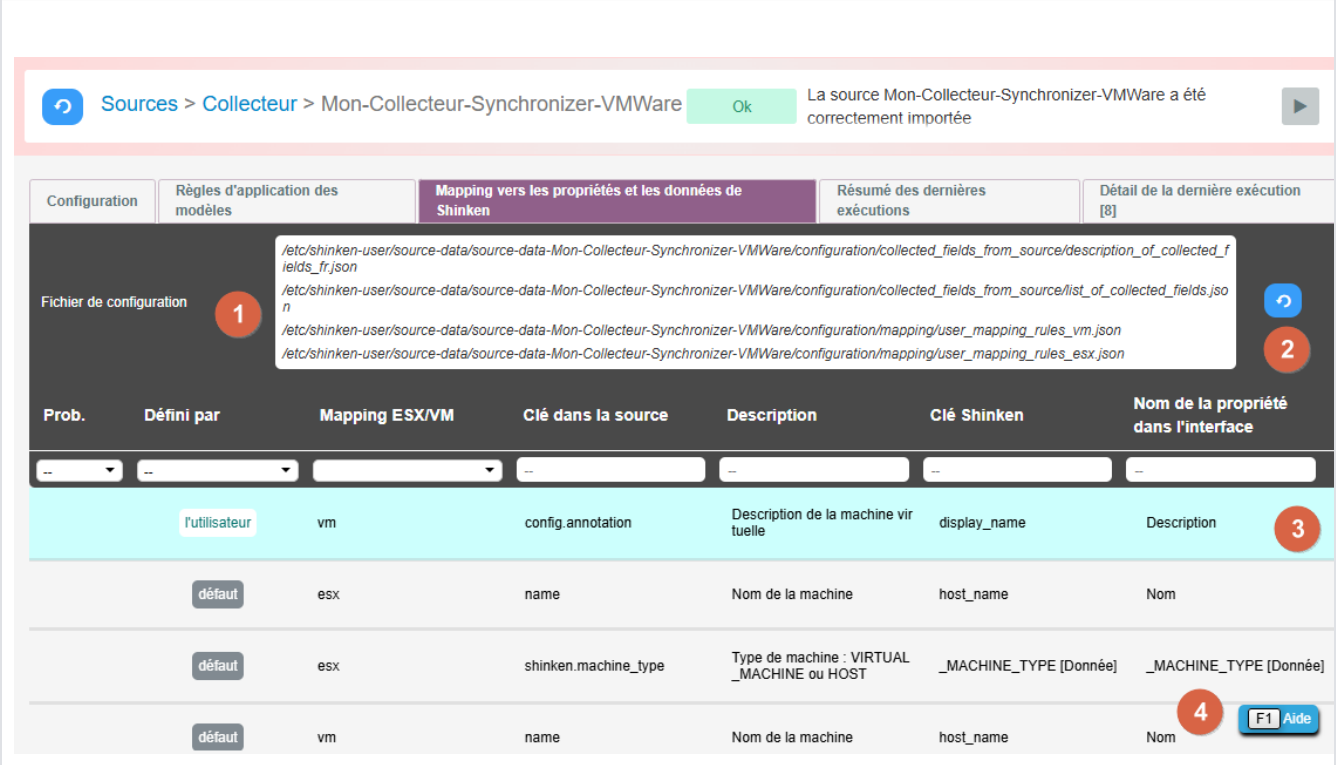
Pour visualiser la liste des mappings définis pour cette source, il faut se rendre dans l'onglet "**Mapping vers les propriétés de Shinken**".

Dans cet onglet, se trouve le chemin des fichiers de définition des règles mapping des utilisateurs pour l'ESXi et les Machines virtuelles du serveur ESXi **(1)**.

- ⚠️ Un message d'avertissement apparaîtra si un des fichiers n'existe pas.
- Pour résoudre ce problème, créer le fichier avec le chemin indiqué sur l'interface puis appuyer le bouton de rafraîchissement **(2)**.
- Ce fichier peut être vide. Dans ce cas les valeurs par défaut seront utilisées.

Dans l'entête de la liste, se trouve également le bouton de rafraîchissement de la liste des mappings **(2)**.

- Ce bouton permet de rafraîchir la liste des mappings sans devoir redémarrer le Synchronizer.
- Il faudra relancer un import pour réappliquer les nouveaux mapping sur les éléments importés.



La liste des mappings **(3)** comporte à la fois les mappings définis par l'utilisateur (*ligne bleue*) et les mappings par défaut (*ligne grise*).

La liste affiche pour chaque mapping une série de colonnes :

- Prob. => Affiche les problèmes ou les avertissements si le mapping n'a pas été correctement défini.
- Défini par => Par qui le mapping a été définie (*Par défaut ou par l'utilisateur*).
- Mapping ESXi/VM => Le type de mapping (*un mapping spécifique à l'ESXi ou un mapping spécifique aux VMs du serveur ESXi*).
- La clé dans la source => Le nom du champ dans l'API VMWare.
- Description => La description du champ dans la source.
- Clé Shinken => Nom de la propriété ou donnée Shinken vers laquelle la valeur sera copiée.
- Le nom de la propriété ou donnée Shinken dans l'interface.

Certaines des règles de mapping définies par "Défaut" ont une valeur définie dans les champs "**Clé de la source"** et la "**Description**", mais aucune valeur dans les champs " ***Clé Shinken*****"** et " ***Nom de propriété dans l'interface*** ".

- Cela signifie que ces mappings ne seront liés à aucune propriété Shinken.
- Elles sont volontairement listées ici pour être vus, et faciliter la création de son propre mapping (*afin de ne pas partir d'une feuille blanche*).

Ces mappings peuvent être surchargés dans l'un des fichiers de mapping utilisateur **(1)**.

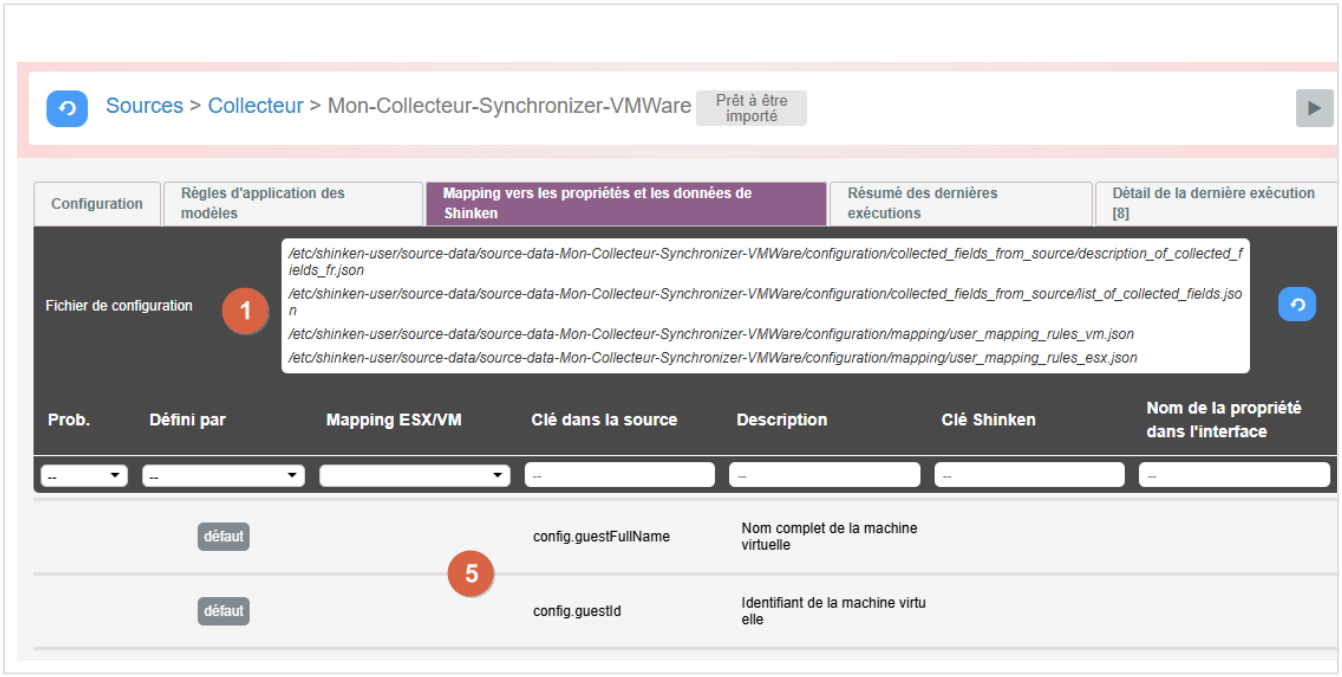
En bas à droite **(4)** se trouve le bouton pour afficher l'aide de la page (*Raccourci sur la touche F1*).

Exemple

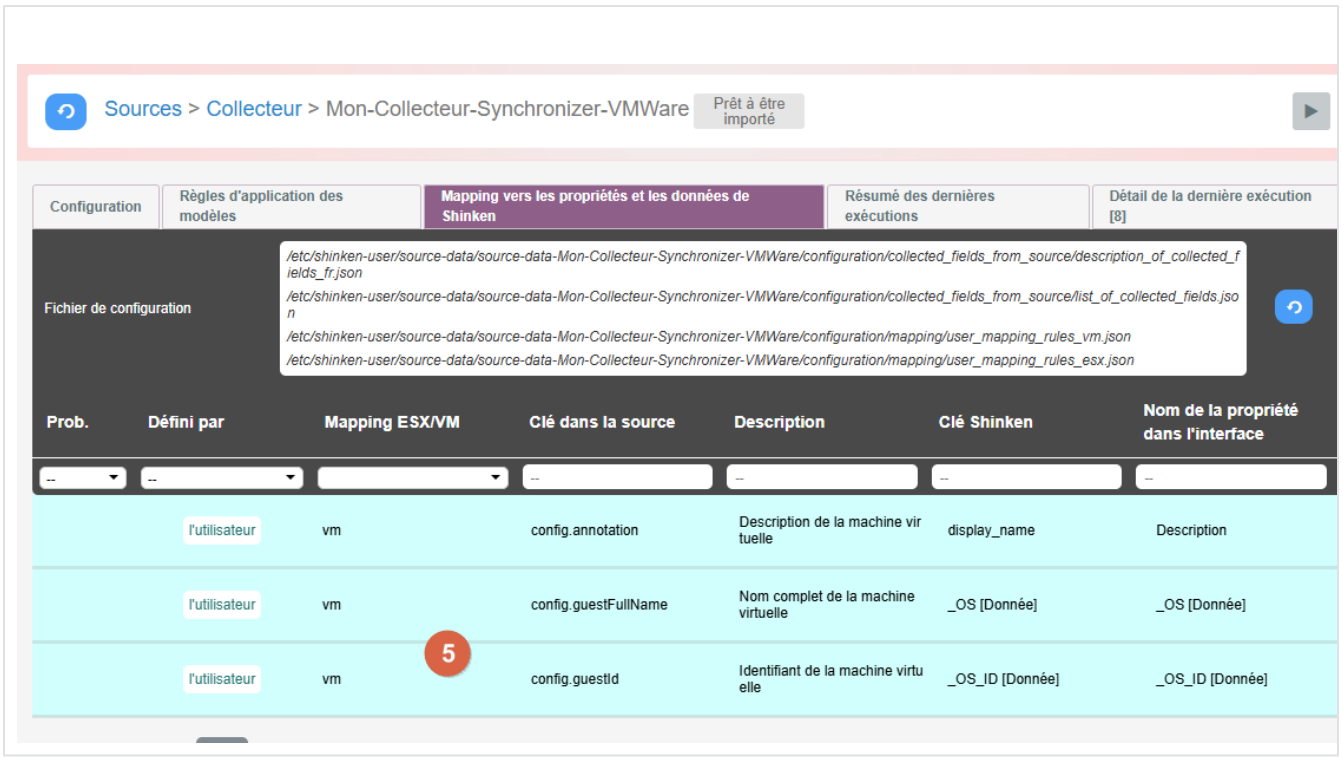
Les exemples suivants surchargent les règles de mapping de " ***config.guestFullName"*** et " ***config.guestId*** " qui ne sont pas liées à une propriété Shinken.

Pour ce faire, il suffit de créer, dans l'un des fichiers de mapping utilisateur **(1)** , une règle possédant la même clé que celle à surcharger **(5)** .

Avant :



Après :


Sources > Collecteur > Mon-Collecteur-Synchronizer-VMWare
Prêt à être importé
Configuration
Règles d'application des modèles
Mapping vers les propriétés et les données de Shinken
Résumé des dernières exécutions
Détail de la dernière exécution [8]
Fichier de configuration
/etc/shinken-user/source-data/source-data-Mon-Collecteur-Synchronizer-VMWare/configuration/collected_fields_from_source/description_of_collected_fields_fr.json
/etc/shinken-user/source-data/source-data-Mon-Collecteur-Synchronizer-VMWare/configuration/collected_fields_from_source/list_of_collected_fields.json
/etc/shinken-user/source-data/source-data-Mon-Collecteur-Synchronizer-VMWare/configuration/mapping/user_mapping_rules_vm.json
/etc/shinken-user/source-data/source-data-Mon-Collecteur-Synchronizer-VMWare/configuration/mapping/user_mapping_rules_esx.json
Prob.
Défini par
Mapping ESX/VM
Clé dans la source
Description
Clé Shinken
Nom de la propriété dans l'interface
l'utilisateur
vm
config.annotation
Description de la machine virtuelle
display_name
Description
l'utilisateur
vm
config.guestFullName
Nom complet de la machine virtuelle
_OS [Donnée]
_OS [Donnée]
l'utilisateur
vm
config.guestId
Identifiant de la machine virtuelle
_OS_ID [Donnée]
_OS_ID [Donnée]
5