V1 - (PUT) /shinken/listener-rest/v1/hosts (création d'hôte)

Sommaire

Définition
Réponse
Exemple

Concept général

Shinken Entreprise permet d'effectuer des remplacements dynamiques de contenu dans les objets de configuration (*autrefois appelé "MACRO"*). Ces remplacements sont utilisés pour permettre une factorisation de la configuration, ainsi que pour fournir une grande flexibilité dans la création et modification d'une configuration de supervision.

Un exemple pour illustration :

On dispose d'une commande qui se charge de contacter un hôte pour déterminer si il est joignable ou non.

On veut donc que la commande récupère automatiquement l'adresse de l'hôte sans avoir à spécifier manuellement l'adresse pour chaque hôte.

Pour résoudre ce problème, on effectue un remplacement de contenu. Dans Shinken Entreprise, on peut utiliser la notation $HOSTADDRESS$ qui va contenir l'adresse de l'hôte courant.

Ainsi, en utilisant cette notation dans notre commande, lorsque celle-ci sera utilisée lors de la vérification d'un hôte, le mécanisme de remplacement dynamique va automatiquement remplacer la notation par l'adresse de l'hôte .

Commande avant remplacement

```
check_ping -H "$HOSTADDRESS$" (...autres paramètres)
```

Commande après remplacement

```
check_ping -H "192.168.1.12" (...autres paramètres)
```

Les différents types de Variables

Dans l'exemple précédent, la notation entre dollars (*$*) est utilisée pour permettre le remplacement dynamique d'une donnée locale à l'hôte dans une commande.

Il s'agit seulement d'un exemple parmi les différents types de contenus existant. Ces contenus peuvent être séparés en 3 grandes catégories :

- Les données locales
- Les globales
- Les opérateurs, permettant un transfert de données

Les Variables sur un élément

Les données locales désignent les données personnalisées et les propriétés d'un objet particulier.

Les Variables pour les Propriétés ($HOST..., $SERVICE..., $CONTACT...)

Parmi tous les éléments de Shinken Entreprise, il est possible d'accéder aux attributs des **hôtes**, des **checks** et des **utilisateurs**.

Dans le premier exemple, la notation entre dollars permet d'accéder à une propriété de l'hôte, par exemple l'adresse: $HOSTADDRESS$.

On accède ici à la propriété "address" de l'objet HOST. On peut accéder aux autres attributs de l'hôte, mais aussi à ceux d'un check ou d'un utilisateur.

Remplacement de données locales

? Unknown Attachment

Il est donc possible d'accéder aux propriétés des hôtes, des checks ou bien des utilisateurs. Pour cela, il faut commencer le nom de la Variable par HOST, SERVICE (*pour les checks*) ou CONTACT (*pour les utilisateurs*).

Par exemple:

- $**HOST**ADDRESS$
- $**SERVICE**DISPLAYNAME$
- $**CONTACT**EMAIL$

Le schéma ci-dessus explique le cas du remplacement des données locales pour les checks et les hôtes.

Dans le cas des utilisateurs, plusieurs utilisateurs peuvent être accrochés sur un hôte ou un check. Ils sont utilisés pour envoyer les notifications lorsque l'hôte ou le check passe dans un état d'erreur.

Dans ce cas, une notification est envoyée à chaque utilisateur. La commande utilisée pour envoyer la notification peut alors utiliser un remplacement dynamique de contenu pour accéder aux informations de l'utilisateur qui va recevoir la notification.

Les Variables pour les Données ($_HOST..., $_SERVICE..., $_CONTACT...)

Shinken Entreprise permet d'ajouter des données personnalisées sur certains éléments, comme les hôtes, les checks, les utilisateurs, et bien sûr les modèles d'hôtes, de checks et d'utilisateurs. Ces données permettent des créés des propriétés personnalisées lorsque les propriétés par défaut de l'objet ne suffisent pas à le décrire complètement.

Par exemple, si un hôte possède deux interfaces réseau, les propriétés par défaut de l'objet ne permettent pas de spécifier deux adresses. Par contre, il est possible d'ajouter une donnée personnalisée qu'on appelle par exemple "ADDRESS_2" qui pourra être utilisée lorsqu'on aura besoin d'avoir la deuxième adresse de l'hôte dans un check.

Ces données sont accessibles dans un objet en utilisant la notation suivante :

- $_**HOST**NOMDELADONNEPERSONNALISEE$.
- $_**SERVICE**NOMDELADONNEPERSONNALISEE$
- $_**CONTACT**NOMDELADONNEPERSONNALISEE$

Remarque

On remarque la présence d'un underscore (_) avant HOST, SERVICE ou CONTACT, ce qui permet de différencier l'accès à une propriété de l'élément et l'accès à une donnée personnalisée.

Remplacement des données personnalisées
? Unknown Attachment

Les Variables Globales

Il est possible en effectuant un remplacement de contenu d'accéder aux données locales à un hôte, un check ou un utilisateur. Il est aussi possible dans Shinken Entreprise de définir des globales accessibles partout dans Shinken et qui ne dépendent pas d'un élément particulier.

Ces globales se définissent dans des fichiers de configuration, dont le détail sera expliqué dans la section qui traite l'utilisation pratique des remplacements dynamiques de contenu. Elles sont accessibles simplement par leur nom, sans avoir besoin de le préfixer de _HOST, _SERVICE ou _CONTACT.

Par exemple, une globale nommée MAGLOBALE sera accessible avec la notation $MAGLOBALE$.

Remplacement des données globales
? Unknown Attachment

Les Variables spécifiques ($ARGn$ et $VALUEn$)

Nous avons vu que lorsque l'on remplace dynamiquement du contenu, il est possible de faire référence aux données locales ainsi qu'aux globales. Il existe également des opérateurs spéciaux, qui ne récupèrent pas les globales et les données locales d'un élément, mais qui permettent de transférer des données, notamment des arguments de commande.

Ces opérateurs spéciaux sont accessibles avec les notations $ARGn$ et $VALUEn$.

Référence aux arguments d'une commande

Shinken Entreprise permet aux hôtes et aux checks de spécifier des commandes qui seront utilisées pour la vérification de l'élément. Dans l'optique de rendre ces commandes les plus génériques possible, et de permettre de factoriser la configuration, des arguments peuvent être passés à ces commandes.

Ces arguments sont séparés par des points d'exclamation '**!**'.

Dans l'exemple, un check utilise la commande "ma_commande" et lui passe 3 arguments.

? Unknown Attachment

Dans la commande, on veut donc pouvoir récupérer la valeur des ces arguments pour pouvoir les utiliser dans le script.

Les notations $ARGn$ sont donc utilisées dans ce cas. La notation $ARGn$ permet simplement d'accéder à la valeur du n-ième argument.

Dans l'exemple, on utilise donc $ARG1$, $ARG2$ et $ARG3$ pour récupérer les valeurs du premier, deuxième et troisième argument.

? Unknown Attachment

Il est possible d'utiliser des Données comme argument.

- Exemple : 200!$_HOSTWARNING_LEVEL$
- Elle sera remplacée lors de l'évaluation de la commande.

Pour une utilisation avancée, et avec un certain niveau de maîtrise nécessaire, il est même possible d'utiliser une donnée pour définir plusieurs arguments.

- Il faut utiliser pour cela le séparateur : |-|
- Exemple :
  - Dans les arguments : VERBOSE!$_HOSTLEVELS$
  - Dans l'hôte la donnée LEVELS vaut 200|-|400
  - La commande script.py $ARG1$ $ARG2$ $ARG3$ deviendra script.py VERBOSE 200 400

Récupération des arguments dans une commande

? Unknown Attachment

Comme dans Nagios, il est possible d'utiliser jusqu'à 32 arguments.

Ainsi, les notations $ARG1$ à $ARG32$ sont utilisables.

Cas du Duplicate Foreach

La fonctionnalité avancée Dupliquer des checks en fonction d'une liste de valeurs présentes dans la Donnée d'un hôte (duplicate_foreach) permet d'appliquer plusieurs fois le même check sur un hôte avec des paramètres différents, selon la valeur d'une donnée personnalisée sur l'hôte.

Sur chaque check utilisant la fonctionnalité Duplicate Foreach, est affecté une clé, et optionnellement des paramètres.

Exemple de donnée Duplicate Foreach

```
check1$(valeur1)$$(valeur2)$$(valeur3)$
```

La valeur de la clé est accessible avec la notation KEY, et les arguments sont accessible grâce aux notations $VALUEn$.

Le tableau suivant récapitule les notations permettant d'accéder aux valeurs de la donnée Duplicate Foreach:

Variable	Valeur
KEY	check1
$VALUE1$	valeur1
$VALUE2$	valeur2
$VALUE3$	valeur3

Exemple d'utilisation des données Duplicate Foreach

? Unknown Attachment

Il possible d'utiliser 16 valeurs différentes. Ainsi, les notations $VALUE1$ jusqu'à $VALUE16$ sont valides.

Utilisation pratique des remplacements dynamiques de données

Les données locales

Définir des données personnalisées

Les données locales peuvent être définies sur les hôtes, checks, utilisateurs et leurs modèles respectifs de 2 manières:

- Par fichier de configuration
- Par l'interface de configuration

Dans un fichier de configuration, les données sont définies en préfixant un _ à leur nom. Le nom d'une donnée peut contenir seulement des caractères alphanumériques (A-Z0-9), des tirets (-) ou underscore (_). Aussi, le nom d'une donnée sera toujours en majuscules.

Exemple d'une objet définissant la donnée DONNEE_PERSONNALISEE

```
define host {
    host_name    mon_hote
    address                192.168.0.12


    _DONNEE_PERSONNALISEE      valeur_de_la_donnée
}
```

Dans l'interface de configuration, l'ajout et la modification de données personnalisées s'effectuent grâce à l'onglet "Données".

Ajout d'une donnée dans l'interface de Configuration

? Unknown Attachment

Cette capture d'écran montre l'édition de données personnalisées dans le cas d'un hôte. Les mêmes manipulations sur les données personnalisées sont possibles pour les modèles d'hôtes, clusters, modèles de clusters, checks, modèles de check, utilisateurs et modèles d'utilisateurs.

Dans un fichier de configuration, une donnée personnalisée est définie précédée d'un underscore (_).

Dans l'interface de configuration, cet underscore ne doit pas être spécifié, car il s'agit seulement d'un moyen dans les fichiers de différencier une donnée personnalisée d'une propriété. La déclaration d'une donnée personnalisée depuis l'interface se fait seulement en spécifiant le nom de la donnée et sa valeur.

Utiliser des données locales

Lorsqu'on veut accéder à des données locales, il faut différencier l'utilisation de la notation entre dollars ($) donnant accès aux données personnalisées et celles permettant l'accès à certains attributs de l'élément.

Attributs de l'élément

Variable	Fonction
$HOSTPROPRIETE$	Accède une propriété de l'hôte
$SERVICEPROPRIETE$	Accède à une propriété du check
$CONTACTPROPRIETE$	Accède à une propriété de l'utilisateur

Données personnalisées

Variable	Fonction
$_HOSTDONNEE$	Accède à la donnée personnalisée "DONNEE" de l'hôte
$_SERVICEDONNEE$	Accède à la donnée personnalisée "DONNEE" du check
$_CONTACTDONNEE$	Accède à la donnée personnalisée "DONNEE" de l'utilisateur

La liste des propriétés disponibles pour chaque élément (*hôtes, checks et utilisateurs*) est présente dans la section Propriétés accessibles dans les remplacements de contenu

Les globales

Définir des données globales

Les données globales peuvent être définies uniquement par fichiers de configuration.

Par défaut, un certain de nombre de globales sont définies dans le dossier **/etc/shinken/resource.d**, dans lequel sont présents tous les fichiers qui déclarent des globales. Au démarrage de Shinken, ces fichiers sont donc chargés et les globales qui y sont définies sont alors disponibles.

La syntaxe pour la déclaration des globales est la suivante:

Syntaxe de déclaration des globales

```
# Commentaire: les lignes commencant par # seront ignorées
# Les noms de globales doivent être entourés de $


$NOMDELAGLOBALE$=valeur
```

Comme pour les données locales, les noms de globaux ne peuvent contenir que des caractères alphanumériques (A-Z0-9), des tirets (-) et des underscore (_). Comme pour les données locales, le nom d'une globale sera toujours en majuscules.

Pour permettre à l'utilisateur de faire ses propres packs et faciliter l'import d'une configuration externe, il est possible de déclarer des globales dans une source. Pour cela, il faut placer les fichiers .cfg dans un dossier **source_data** de la source.

Les fichiers de déclaration de globaux seront donc copiés dans **/etc/shinken/resource.d/** et disponibles comme les autres globales.

Utiliser des données globales

Les globales sont accessibles en spécifiant seulement le nom de la globale entourée par des dollars ($).

La globale "MAGLOBALE" est donc accessible par la notation $MAGLOBALE$.

⚠ Parce que les globales sont définies dans les fichiers de configuration, un ajout ou modification d'une globale dans ces fichiers nécessite un redémarrage de Shinken Entreprise pour que les modifications soient prises en compte.

Remarques sur la notation entre dollars ($)

ⓘ Quelle que soit l'utilisation d'une valeur entre dollars, cette valeur doit toujours être en majuscule. Si à l'import des fichiers CFG ou lors de la modification sur l'interface de configuration, une valeur entre dollars comporte des minuscules, celles-ci seront converties en majuscules et un avertissement sera affiché.

? Unknown Attachment

Boucle de référencement dans les remplacements de contenu

Il est possible, lors de la définition d'une donnée personnalisée, de référencer une autre valeur accessible par une notation entre dollars.
Il est alors possible d'effectuer une boucle de référencement sans le vouloir.
Si c'est le cas, cette erreur est indiquée dans l'onglet Checks de la page d'édition d'un hôte.

Propriétés accessibles dans les remplacements de contenu

Propriétés des hôtes

Syntaxe	Description
$HOSTNAME$	Nom de l'hôte (*propriété host_name*)
$HOSTDISPLAYNAME$	Nom d'affichage de l'hôte (*propriété display_name*)
$HOSTADDRESS$	Adresse de l'hôte (*propriété address*)
$HOSTSTATE$	Etat courant de l'hôte (*UP, DOWN, ou UNREACHABLE*)
$HOSTSTATETYPE$	Type d'état permettant la confirmation du statut d'un hôte (*SOFT ou HARD*)
$HOSTSTATEID$	Numéro correspondant à l'état courant de l'hôte (*0=UP, 1=DOWN, ou 2=UNREACHABLE*)
$LASTHOSTSTATE$	Etat précédent de l'hôte (*UP, DOWN, ou UNREACHABLE*)
$LASTHOSTSTATEID$	Numéro correspondant à l'état précédent de l'hôte (*0=UP, 1=DOWN, ou 2=UNREACHABLE*)
$HOSTGROUPNAME$	Nom du groupe d'hôte auquel appartient l'hôte. Si il appartient à plusieurs groupes d'hôtes, un seul sera retourné
$HOSTGROUPNAMES$	Liste des groupes d'hôtes auxquels appartient l'hôte, séparés par des virgules
$LASTHOSTCHECK$	Date au format timestamp de la dernière vérification de l'hôte
$LASTHOSTSTATECHANGE$	Date au format timestamp du dernier changement d'état de l'hôte
$LASTHOSTUP$	Date au format timestamp du dernier état UP de l'hôte
$LASTHOSTDOWN$	Date au format timestamp du dernier état DOWN de l'hôte
$LASTHOSTUNREACHABLE$	Date au format timestamp du dernier état UNREACHABLE de l'hôte
$HOSTOUTPUT$	Résultat de la dernière vérification de l'hôte
$LONGHOSTOUTPUT$	Résultat long de la dernière vérification de l'hôte
$HOSTPERFDATA$	Données de performances renvoyées par la dernière vérification de l'hôte
$HOSTCHECKCOMMAND$	Nom de la commande utilisée pour la vérification de l'hôte (*avec les paramètres*)
$HOSTNOTESURL$	URL externe de l'hôte (*propriété notes_url*)
$HOSTBUSINESSIMPACT$	Nombre entre 0 et 5 indiquant l'impact métier de l'hôte
$HOSTFIRSTNOTIFICATIONDELAY$	Nombre de minutes à attendre avant d'envoyer la **première** notification pour un hôte
$HOSTNOTIFICATIONNUMBER$	N° d'ordre d'une notification pour un événement donné
$HOSTTHRESHOLDSDISPLAY$	Affichage des seuils, tel qu'il est paramétré sur l'hôte

On accède en général aux propriétés de l'hôte avec la notation entre dollars commençant par HOST (ex $HOSTADDRESS$). Dans le tableau, certaines entrées ne commencent pas par HOST sont présentes, mais font quand même référence à une propriété de l'hôte.

Propriétés des checks

Syntaxe	Description	Synchronizer	Scheduler
$SERVICEDESC$	Nom/description du check	X	X
$SERVICEDISPLAYNAME$	Nom d'affichage du check (*propriété display_name*)	X	X
$SERVICESTATE$	Etat courant du check (*OK, WARNING, UNKNOWN, CRITICAL*)		X
$SERVICESTATETYPE$	Type d'état permettant la confirmation du statut d'un check (*SOFT ou HARD*)		X
$SERVICESTATEID$	Numéro correspondant à l'état courant du check (*0=UP, 1=DOWN, ou 2=UNREACHABLE*)		
$LASTSERVICESTATE$	Etat précédent du check (*OK, WARNING, UNKNOWN, CRITICAL*)		
$LASTSERVICESTATEID$	Numéro correspondant à l'état précédent du check (*0=UP, 1=DOWN, ou 2=UNREACHABLE*)		
$SERVICEISVOLATILE$	Booléen indiquant si le check est volatile (*0=Non volatile, 1=Volatile*)		
$LASTSERVICECHECK$	Date au format timestamp de la dernière exécution du check		
$LASTSERVICESTATECHANGE$	Date au format timestamp du dernier changement d'état du check		
$LASTSERVICEOK$	Date au format timestamp du dernier état OK du check		
$LASTSERVICEWARNING$	Date au format timestamp du dernier état WARNING du check		
$LASTSERVICEUNKNOWN$	Date au format timestamp du dernier état UNKNOWN du check		
$LASTSERVICECRITICAL$	Date au format timestamp du dernier état CRITICAL du check		
$SERVICEOUTPUT$	Résultat de la dernière vérification du check		
$LONGSERVICEOUTPUT$	Résultat long de la dernière vérification du check		
$SERVICEPERFDATA$	Données de performances renvoyées par la dernière exécution du check		
$SERVICECHECKCOMMAND$	Nom de la commande utilisée pour l'exécution du check (*avec les paramètres*)		
$SERVICENOTESURL$	URL externe du check (*propriété notes_url*)		
$SERVICEBUSINESSIMPACT$	Nombre entre 0 et 5 indiquant l'impact métier du check		
$SERVICEFIRSTNOTIFICATIONDELAY$	Nombre de minutes à attendre avant d'envoyer la **première** notification pour un service		
$SERVICENOTIFICATIONNUMBER$	N° d'ordre d'une notification pour un événement donné		
$SERVICETHRESHOLDSDIPLAY$	Affichage des seuils, tel qu'il est paramétré sur le check		

On accède en général aux propriétés du check avec la notation entre dollars commençant par SERVICE (*ex $SERVICEDESC$*). Dans le tableau, certaines entrées ne commencent pas par HOST sont présentes, mais font quand même référence à une propriété du check.

Propriétés des utilisateurs

Syntaxe	Description
$CONTACTNAME$	Nom de l'utilisateur (*propriété contact_name*)
$CONTACTEMAIL$	Adresse mail de l'utilisateur (*propriété email*)
$CONTACTPAGER$	Numéro de téléphone de l'utilisateur (*propriété pager*)

Globales prédéfinies

Certaines globales présentes ci-dessous sont définies par Shinken et accessibles comme n'importe quelle autre globale.

Syntaxe	Description

$LONGDATETIME$	Heure/date courante au format long (par ex Fri Oct 13 00:30:28 CDT 2000)
$SHORTDATETIME$	Heure/date courante au format court (par ex 10-13-2000 00:30:28)
$DATE$	Date courante (par ex 10-13-2000)
$TIME$	Heure courante (par ex 00:30:28)
$TIMET$	Heure courante au format timestamp

Limites lors de l'évaluation des variables dynamiques pour générer une ligne de commande

Il peut arriver que le résultat de certaines variables dynamiques nécessite l'évaluation d'autres variables dynamiques pour être obtenu.

Pour éviter tout emballement récursif (*exemple:* ***VARIABLE_1*** *nécessite* ***VARIABLE_2*** *qui nécessite* ***VARIABLE_3*** *qui nécessite* ***VARIABLE_1***), les limites suivantes sont appliquées lors de la résolution des variables dynamiques :

- il ne peut pas y avoir plus de 32 niveaux d'imbrication de variables, au delà de ce niveau, les variables ne sont plus résolues
- il ne peut pas y avoir plus de 255 variables à résoudre sur la ligne de commande, au delà de ce nombre, les variables ne sont plus résolues
- la ligne de commande générée après résolution des variables dynamiques ne peut pas excéder 65000 caractères.

Si un dépassement se produit, la résolution des variables est interrompue, et la ligne de commande est tronquée pour être remplacée par une commande remontant cette information.

Différences de comportement entre le Synchronizer (configuration) et le Scheduler (production)

Certaines données n'étant disponibles qu'en production, le Synchronizer se contente de simuler leur existence lors des essais de check (*exemple:* ***$HOSTPERFDATA$, $SERVICEPERFDATA$*** *...*)

En conséquence, certaines problématiques que l'ont peut rencontrer en production ne peuvent pas être testées ou observées depuis l'essai de check, , comme par exemple une variable dynamique qui se substitue par une multitude de variables dynamiques.

En particulier, **pendant les résolutions** des variables dynamiques la différence suivante peut être observée :

- le Scheduler, en production, gère le nombre de variables dynamiques présentes sur la ligne de commande, à chaque étape de substitution
- le Synchronizer, lors de l'essai de check, ne gère pas le nombre de variables présentes sur la ligne de commande lors des substitutions, et il n'applique pas la règle limitant leur nombre à 255 à chaque étape de substitution.