Le Scheduler

Concept général

Les macros sont des objets spéciaux permettant d'effectuer des remplacements de données dans les différents éléments de Shinken Entreprise. Elles sont utilisées 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 utilise donc une macro. Dans Shinken Entreprise, on peut utiliser la macro $HOSTADDRESS$ qui va contenir l'adresse de l'hôte courant.

Ainsi, en utilisant cette macro dans notre commande, lorsque la commande sera utilisée lors de la vérification d'un hôte, l'adresse de l'hôte sera automatiquement remplacée dans la commande.

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 macros

Dans l'exemple précédent, une macro est utilisée pour permettre le remplacement dynamique d'une donnée locale à l'hôte dans une commande.

Il s'agit seulement d'un exemple des différents types de macros existantes. Ces macros peuvent être séparées en 3 grandes catégories:

- Les macros permettant un accès aux données locales
- Les macros spécifiques, permettant de transférer les données
- Les macros permettant un accès aux données globales

Les données locales

Les macros concernant les données locales permettent de faire référence à une donnée ou un attribut d'un objet particulier. Ces données locales peuvent être des attributs d'un élément, ou bien des données personnalisées.

Accès aux attributs d'un élément

Les macros concernant les données locales permettent d'accéder aux attributs d'un élément. Parmi tous les éléments de Shinken Entreprise, il est possible d'accéder aux attributs des hôtes, des checks et des utilisateurs.

Ce type de macro est déjà présent dans le premier exemple: $HOSTADDRESS$.

On accède ici à l'attribut "address" de l'objet HOST. On peut accéder aux autres attributs de l'hôte, mais aussi à ceux du check.

Remplacement des macros
? Unknown Attachment

Les macros permettent d'accéder aux attributs des hôtes, des checks ou bien des contacts. Pour cela, il faut commencer le nom de la macro par HOST, SERVICE ou CONTACT.

Par exemple:

- $HOSTADDRESS$
- $SERVICEDISPLAYNAME$
- $CONTACTEMAIL$

Le schéma ci-dessus explique le cas du remplacement des données 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 les macros pour accéder aux informations de l'utilisateur qui va recevoir la notification.

Accès aux données personnalisées d'un élément

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

Par exemple, si un hôte possède 2 interfaces réseau, les attributs par défaut de l'objet ne permettent pas de spécifier 2 adresses. Par contre, il est possible d'ajouter une donnée 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 une macro de la manière suivante:

- $_HOSTnomdeladonnéepersonnalisée$.
- $_SERVICEnomdeladonnéepersonnalisée$
- $_CONTACTnomdeladonnéepersonnalisée$

Remarque

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

Utilisation des données personnalisées

? Unknown Attachment

Les données globales

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

Ces données globales se définissent dans des fichiers de configuration, dont le détail sera expliqué dans la section qui traite l'utilisation des macros. 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 macro $MAGLOBALE$.

Remplacement des données globales

? Unknown Attachment

Macros spécifiques

Nous avons vu que les macros permettent de faire référence aux données locales ainsi qu'aux données globales. Il existe également des macros particulières, qui ne récupèrent pas les données globales et les données spécifiques d'un hôte mais qui permettent de transferer des données, notamment des arguments de commande.

C'est le cas des macros $ARGn$ et $VALUEn$.

Référence aux arguments d'un 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.

? Unknown Attachment

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

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

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

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

? Unknown Attachment

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

Récupération des arguments dans une commande

? Unknown Attachment

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

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

Cas du Duplicate Foreach

La fonctionnalité avancée Dupliquer pour chaque valeur de la Donnée de l'hôte 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 macro KEY, et les arguments sont accessible grâce aux macros $VALUEn$.

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

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

Exemple d'utilisation des données Duplicate Foreach

? Unknown Attachment

Utilisation des macros

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.

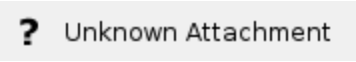
Exemple d'une objet définissant la donnée DONNEE_PERSONNALISEE

```
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'effectue grâce à l'onglet "Données".

Ajout d'une donnée dans l'interface de Configuration



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, checks, modèles de check, utilisateurs et modèles d'utilisateurs.

Utiliser des données locales

Lorsqu'on veut accéder à des données locales, il faut différencier l'utilisation de macros donnant accès aux données personnalisées et celles permettant l'accès à certains attributs de l'élément.

Attributs de l'élément

Macro	Fonction
$HOSTattribut$	Accède un attribut de l'hôte
$SERVICEattribut$	Accède à un attribut du check
$CONTACTattribut$	Accède à un attribut de l'utilisateur

Données personnalisées

Macro	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

TODO: Référence vers la liste des macros qui existe

Les données 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 globales doivent être entourées de $


$NOMDELAGLOBALE$=valeur
```

QESSISPASS si jamais une donnée globale a un nom réservé ???

Utiliser des données globales

Remarques sur la définition et l'utilisation des macros

Inventaire des macros disponibles

Référence vers la doc Shinken readthedocs

A quoi servent les macros ?

L'une des caractéristiques qui rend Shinken Enterprise flexible, c'est sa capacité à utiliser des données dans la définition des Commandes. Ces données permettent de référencer des informations provenant des hôtes, des services, ou d'autres sources dans les commandes.

Remplacement de données

Avant d'exécuter une commande, Shinken Enterprise va remplacer toutes les données trouvées dans la définition de la commande avec leur valeurs correspondantes. Ce remplacement s'opère pour toute commande que Shinken Enterprise exécute : vérification d'hôte et de check, notification, exécution d'événements, etc .

Ce remplacement est récursif. Si une macro contient à son tour une macro, cette seconde macro sera résolue. Ce processus continue jusqu'à ce que la commande ne contienne plus de macro.

L'utilisation littérale du caractère '$' nécessitera l'utilisation de '$$'. C'est également le cas dans les règles de Clusters, qui peuvent aussi contenir des macros.

Le nom d'une donnée peut contenir uniquement des caractères alphanumériques (a-zA-Z0-9).

Exemple 1 : Adresse générique

Lorsque vous utilisez un hôte ou un service dans les données de définitions de commande , ils se réfèrent à des valeurs pour l'hôte ou le service pour lequel la commande est en cours d'exécution

Prenons un exemple. Imaginons que nous utilisons une définition de l'hôte "Linuxbox" qui est vérifié par une commande check_ping, qui est définie comme suit.

```
/var/lib/Shinken Enterprise/libexec/check_ping -H $HOSTADDRESS$ -w 100.0,90% -c 200.0,60%
```

Linuxbox a pour adresse 192.168.1.2.

La macros sera remplacée et la ligne de commande finale suivante sera exécutée :

```
/var/lib/Shinken Enterprise/libexec/check_ping -H 192.168.1.2 -w 100.0,90% -c 200.0,60%
```

Ce remplacement a lieu pour chaque exécution différente de la commande. Cette même commande peut donc servir à vérifier des hôtes différents, mais elle peut être rendue encore plus générique.

Exemple 2 : Argument de commande

Vous pouvez également passer des arguments dans une commande, ce qui permet de garder une définition de commande générique.

Les différents arguments sont séparés par le caractère '!'. On peut donc, dans l'hôte, définir les arguments de la commande check_ping comme étant:

Si vous devez utiliser le caractère (!) dans les arguments de votre commande, vous pouvez éviter son interprétation en le préfixant d'un anti-slash (\ ). si vous avez besoin de l'anti-slash dans une commande, il suffit de doubler l'anti-slash (\\ ).

```
200.0,80%!400.0,40%
```

Ces arguments deviennent alors disponibles dans la commande par les macros $ARGn$. $ARG1$ sera remplacé en "200.0,80%" et $ARG2$ en "400.0,40%". La définition de la commande peut alors être réécrite :

```
/var/lib/Shinken Enterprise/libexec/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$
```

Exemple 3 : Utilisation des données

Qu'en est-il si plusieurs hôtes partagent les mêmes arguments de commande ?

Le plus simple est de définir un modèle (voir la Logique des modèles) contenant ces données spécifiques.

Nous allons créer un modèle contenant des données. Pour cet exemple, nous les nommerons WARNINGPING et CRITICALPING, et contiendront ces valeurs.

> Dans l'interface de configuration, elles sont disponibles dans l'onglet de données du modèle. Dans un fichier d'import, il sera nécessaire de les préfixer avec '_'.

On spécifiera dans le modèle que la commande est appelée avec, en argument, ces données:

```
$_HOSTWARNINGPING$!$_HOSTCRITICALPING$
```

La commande n'a pas besoin d'être modifiée, de par l'application récursive des macros.

Cela ouvre une nouvelle possibilité: si, on imagine, que sur un hôte donné, le seuil doit être différent, il suffira alors de surcharger les données WARNINGPING et CRITICALPING dans l'hôte, et la commande lancée sera spécialisée pour cet hôte uniquement.

Il est possible d'effectuer de même pour les Checks, et donc d'appeler en argument $_SERVICEWARNINGPING$!$_SERVICECRITICALPING$ ce qui ira prendre les valeurs de la données WARNINGPING et CRITICALPING du Check.