A quoi cela sert?

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 Variable de Donnée contient à son tour une autre variable, cette seconde variable sera résolue. Ce processus continue jusqu'à ce que la commande ne contienne plus de variable.

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

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 Variable 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:

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

Ces arguments deviennent alors disponibles dans la commande par les Variables $ARGn$. $ARG1$ 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 $ARG2$
```

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 (\\ ).

Il est possible d'utiliser des Données comme argument.

Exemple : 200.0,80%!%_HOSTWARNING_LEVEL%

Il est même possible de définir plusieurs argument dans une donnée avec le séparateur : |-|

Exemple :

- Dans les arguments : %_HOSTLEVELS%
- Dans l'hôte la valeur _LEVELS 200.0|-|400.0,40%
- La commande check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ deviendra check_ping -H $HOSTADDRESS$ -w 200.0,80% -c 400.0,40%

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 template (voir la Logique des templates) contenant ces données spécifiques.

Nous allons créer un template 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 template. Dans un fichier d'import, il sera nécessaire de les préfixer avec '_'.

On spécifiera dans le template 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 Variables.

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.