Les Sondes

Sommaire

Introduction

Shinken Enterprise s'appuie sur des programmes externes appelés "**sondes**" (*plugins dans l'univers de la supervision en anglais*) pour pouvoir superviser une large variété d'éléments.

- Le démon Poller exécute les sondes de supervision.
- Il devra donc pouvoir accéder à ces sondes directement sur leur système.

Un certain nombre de sondes sont incluses lorsque de l'installation de Shinken Enterprise, mais il est également possible d'utiliser celle de la communauté ou de développer les vôtres du moment qu'elle respecte les règles décrites dans cette page (*hérité de l'univers nagios*).

Si vous développez vos sondes :

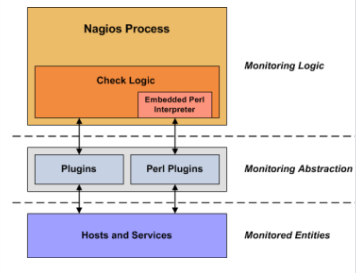
- Avec quelques compétences en scripting, il est par ailleurs possible de créer des sondes assez simplement, il suffira d'utiliser les codes retours correspondants.
- mais n'importe quel langage peut être utilisé, avec une préférence pour les langages compilés qui consomment peu de CPU
  - la quantité de temps machine consommé est l'indicateur clé pour les sondes,
  - car plus vos sondes consomment et plus, vous aurez besoin de CPU sur vos Pollers pour exécuter les sondes.

Qu'est ce qu'une sonde ou un plugin ?

Les sondes sont des **exécutables compilés** ou des **scripts** (*Python, Perl, Shell, etc*.) qui peuvent être lancés par une ligne de commande afin de vérifier le statut d'un hôte ou bien d'un check particulier sur un hôte.

- La sonde procède à la vérification de bon fonctionnement en retournant le résultat au format standard attendu
- Shinken Enterprise qui prendra alors les actions nécessaires en fonction de ce retour (*lancement d'événement, envoi de notifications, etc*).

- Shinken Enterprise lancera une sonde à chaque fois que ce sera nécessaire (en fonction de la configuration définie sur l'élément supervisé).



De la même manière que pour Nagios (*l'image ci-contre*), les sondes agissent à un niveau d'abstraction entre la logique de supervision présente dans les démons de Shinken Enterprise et les hôtes supervisés.

Le gros intérêt de l'approche "sonde" est que l'**on peut superviser à peu près tout type d'élément tant qu'il est joignable sur un réseau.**

- Si vous pouvez automatiser le processus de vérification d'un élément, alors vous pouvez le faire avec Shinken Enterprise.
- Il existe plusieurs milliers de sondes (*ou plugins*) créées pour superviser des ressources du type charge du processeur, utilisation disque, ping, etc…

Si vous souhaitez superviser quelque chose d'autre, référez-vous au paragraphe Créer sa sonde (*API*) et créez votre propre sonde, c'est aussi simple !

À noter que pour utiliser une sonde dans Shinken, vous devrez

- l'intégrer dans un objet Commande,
- lui-même utilisé par un check,
- et ce dernier devra être accroché sur un hôte ou un modèle d'hôte.

Quelles sondes sont disponibles ?

Shinken Enterprise est livré avec un ensemble de sondes par défaut.

- Ces sondes sont placées dans le répertoire **/var/lib/shinken/libexec**
  - Elles sont utilisées par des commandes Shinken déjà créées lors de l'installation de Shinken Enterprise.
  - Ne modifiez pas ces sondes présentes dans ce répertoire, car à la prochaine mise à jour, ces sondes seront mises à jour, écrasant toutes modifications que vous aurez réalisées.

- Ces sondes pré-installés permettent aux checks des modèles d'hôtes de fonctionner directement après l'installation ou la mise à jour de Shinken.

- Dans les commandes utilisant ces sondes consultables dans l'Interface de Configuration :
  - Les lignes de définition des commandes font référence aux données de remplacement **$PLUGINSDIR$**, **$USERPLUGINSDIR$,** ou encore $NAGIOSPLUGINS$.
    - Ces données sont globales et déclarées dans le fichier **/etc/shinken/resource.d/paths.cfg**.
    - Ces valeurs peuvent être modifiées,
      - mais il est important de comprendre qu'à la prochaine mise à jour, toute nouveauté dans ce fichier ne sera pas appliqué.
      - Le nouveau fichier **paths.cfg** laissé par la mise à jour se terminera par un paths.cfg**.new**.

  - Ces données représentent les chemins des répertoires des sondes :

Variable	Chemin	Description
$PLUGINSDIR$	/var/lib/shinken /libexec	Ce répertoire contient des sondes adaptées et utilisées spécifiquement pour Shinken Enterprise. Attention, les sondes de ce répertoire seront régulièrement mises à jour, ne pas les modifier.
$USERPLUGINSDIR$	/var/lib/shinken-user /libexec	Ce répertoire contient vos propres sondes. Ce répertoire ne sera jamais modifié lors de mises à jour.
$NAGIOSPLUGINS$	/var/lib/shinken-user /libexec	Ce répertoire contient des sondes issues de Nagios

  - Dans les répertoires **$NAGIOSPLUGINS$** et **$PLUGINSDIR$**, il y a des sondes pour surveiller tout type d'éléments et services.
    - Elles utilisent des protocoles standard comme : WMI, SNMP, SSH, TCP, UDP, ICMP, LDAP et plus encore.
    - Cela permet de surveiller à peu près tout :
      - Unix/Linux, Windows ;
      - Routeurs, Switches:
      - services réseau : "HTTP", "POP3", "IMAP", "FTP", "SSH", "DHCP" ;
      - Charge CPU, utilisation disque, utilisation mémoire… ;
      - Applications, Bases de données, logs et plus.

- Il est possible de créer **ses propres variables globales** :
  - Ces données doivent être placées dans des fichiers de configuration que vous aurez créé dans le répertoire **/etc/shinken/resource.d/**.
    - Pour plus de rangement, il est possible de créer des répertoires et d'y placer les fichiers créés.

Droits et accès

Les sondes sont utilisées par les commandes Shinken. Commandes qui peuvent être rattachées directement à des hôtes et des checks (*via leurs commandes de vérification*), ou encore à des méthodes de notifications (*voir la page* Créer une méthode de notification spécifique - bonnes pratiques) ou commandes lancées par le gestionnaire d'événements (*voir la page* Gestionnaire d'événements).

- Suivant le cas, elles seront donc exécutées par le **Poller** (*pour les hôtes et checks*) ou par le **Reactionner** (*pour les notifications et les gestionnaires d'évènements*).
- Le démon correspondant exécutera alors la commande de son répertoire de sonde.
    - Les processus Shinken des démons, lancés en tant qu'utilisateur Unix "**shinken**", utiliseront donc l'utilisateur "**shinken**" pour l'exécution des sondes.

- Pour la bonne exécution des sondes, il est donc important de s'assurer que l'utilisateur **shinken** puisse correctement accéder en exécution aux scripts/sondes.
    - Pour cela, si besoin, rajoutez les droits UNIX correspondants :

      ```
      chmod 755 ma_sonde.py
      ```

      ou

      ```
      chmod +x ma_sonde.py
      ```

- Les droits propriétaires des scripts peuvent rester root/root, il n'est pas nécessaire de changer le "propriétaire" (*ownership*) du fichier à l'utilisateur **"shinken"** via la commande chown.

Comment utiliser une sonde ?

La plupart des sondes possèdent une option pour afficher l'aide dans la ligne de commande : "-h" ou "--help".

Comme vu précédemment, les scripts sont exécutés via l'utilisateur UNIX **shinken**, il faut donc effectuer les tests à partir de cet utilisateur :

```
su - shinken
```

Par exemple, pour savoir comment fonctionne la sonde **check_http** (*sonde du répertoire $NAGIOSPLUGINS$ qui vérifie le statut du port HTTP d'un serveur*) ou pour savoir quelles options sont possibles, il est possible de lancer la commande suivante :

```
/usr/lib64/nagios/plugins/check_http --help
```

Voici un extrait de l'aide de la commande :

```
Options:
 -h, --help
    Print detailed help screen
 -V, --version
    Print version information
 --extra-opts=[section][@file]
    Read options from an ini file. See
    https://www.nagios-plugins.org/doc/extra-opts.html
    for usage and examples.
 -H, --hostname=ADDRESS
    Host name argument for servers using host headers (virtual host)
    Append a port to include it in the header (eg: example.com:5000)
```

On peut lire que l'option "-H" permet de préciser le nom de l'hôte à tester, donc pour tester la réponse http de www.google.fr, on peut faire la commande suivante :

```
/usr/lib64/nagios/plugins/check_http -H www.google.fr
```

Qui donne le retoure suivante :

```
HTTP OK: HTTP/1.1 200 OK - 11837 bytes in 0.313 second response time |time=0.312690s;;;0.000000
size=11837B;;;0
```

Ici la ligne de retour contient deux parties séparées par le symbole pipe **|**.

Le chapitre suivant présente comment se compose le retour d'une sonde pour la création des scripts de sonde.

Créer sa sonde (API)

Voici comment créer des sondes ! Ces sondes seront à placer dans le répertoire **$USERPLUGINSDIR$**.

Concept de sonde

Les scripts et les exécutables doivent faire au moins deux choses :

- sortir avec une valeur valide de retour (*"code retour"*)
- retourner au moins une ligne de texte sur la sortie standard (*STDOUT*)

Le fonctionnement interne de la sonde importe peu à Shinken Enterprise, c'est l'interface entre eux deux qui compte.

La sonde peut vérifier le statut d'un port TCP, lancer une requête sur une base de données, ou faire tout ce qui est nécessaire pour vérifier un élément sur un réseau LAN ou WAN ou encore sur Internet.

Les détails dépendront de ce qui doit être vérifié.

Code retour

Shinken Enterprise détermine le statut d'un hôte ou d'un check en évaluant le code retour de la sonde.

La table suivante montre la liste des codes retour valides, avec leur statut correspondant, suivante si la commande est attachée à un check ou à un hôte :

Code retour	Statut du Check	Statut de l'hôte
0	OK	UP
1	WARNING	DOWN
2	CRITICAL	DOWN
3	UNKNOWN	DOWN

Format de sortie des données de sonde

Au minimum, la sonde doit retourner une ligne de texte sur la sortie standard (*STDOUT*).

La sonde peut également retourner en option des données de performance pour la métrologie.

Le format standard de sortie d'une sonde est donc le suivant :

TEXT OUTPUT | OPTIONAL_PERFDATA

Les données de performance sont optionnelles. Si une sonde retourne des données de performance, elles doivent être séparées du reste du texte par le symbole pipe **|** .

Exemples de retour de sonde

Voici quelques exemples possibles.

Cas 1: une ligne de retour (*texte seulement*)

Imaginons une sonde qui retourne une ligne telle que décrite en dessous :

```
DISK OK - free space: /var 3326 MB (56%);
```

Si cette sonde est utilisée pour réaliser une vérification de service, la totalité de la ligne retour sera stockée dans **$SERVICEOUTPUT$** .

Cas 2: une ligne de retour (*texte et données de performance*)

La sonde peut retourner en option des données de performance utilisables par Graphite. Dans ce cas, les données de performance doivent être séparées par le symbole pipe " **|** " :

```
DISK OK - used space: /var 3326 MB (56%); | /var=3326
```

Si cette sonde est utilisée pour vérifier un service, la partie à gauche du symbole sera stockée dans **$SERVICEOUTPUT$** et la partie à droite du séparateur sera stockée dans **$SERVICEPERFDATA$**

Cas 3: plusieurs lignes de retour (*texte et données de performance*)

En plus de retourner des données de performances, la sonde peut retourner un résultat long à l'aide d'un saut de ligne.
À ce moment-là, le retour de la sonde sera divisé en trois parties :

- le résultat,
- le résultat long,
- les métriques.

```
DISK OK
- used space: /var 3326 MB (56%); | /var=3326
```

Exemples de retour de sonde avec HTML/CSS

Il est tout à fait possible de retourner du HTML/CSS dans le retour des sondes.

De cette manière, il est possible d'insérer du style et donc d'améliorer le retour des sondes affiché dans l'Interface de Configuration (*évaluation des checks*), dans l'Interface de Visualisation (*liste et panneau de détails*) et dans les mails de notification.

Voici un exemple :

```
<span style="color:#e48c19;font-weight: bold;">[WARNING]</span> 1 disk uses more than 95% of his total disk
space :<br/><li>/ (97%)</li>
<style type="text/css"> .disks-table, .disks-table td, .disks-table th { border: 1px solid #000000 !
important; border-collapse: collapse !important; color: #000000 !important; } .disks-table { width: 90% !
important; } .disks-table-th { background-color: #E8E7E7 !important; width: auto !important; max-width: 20% !
important; padding: 2px !important; word-break: break-word !important; background-color: #E8E7E7 !important;
text-align: center;}.disks-table-td { padding: 2px !important; width: auto !important; max-width: 20% !
important; font-weight: normal !important; word-break: break-word !important; background-color: #FFFFFF !
important; } .disks-host-command { font-style: italic !important; color: #7F7F7F !important; } .disks-table-
center { text-align: center; } </style><br/>More than 95% of total disk space used :<br/> <table class="
disks-table"><tr><th class="disks-table-th">Disk</th><th class="disks-table-th">Usage</th><th class="disks-
table-th">Used</th><th class="disks-table-th">Total</th></tr>
 <tr> <td class="disks-table-td">/</td> <td class="disks-table-td">97%</td><td class="disks-table-td">68.9GB<
/td><td class="disks-table-td">75.2GB</td></tr>
</table><br> | "/boot_used_pct"=16%;95%;98%;0%;100% "/boot_used"=0.069993019104GB;0.441808128357;
0.455759963989;0;0.465061187744 "/_used_pct"=97%;95%;98%;0%;100% "/_used"=68.8651046753GB;71.4608747482;
73.7175339508;0;75.2219734192
```

Attention aux différentes méthodes de notification, la plupart des clients mail pourront interpréter le HTML, mais par exemple une notification par SMS ne sera pas interpréter sur le téléphone.

Exemple de retour de sonde en HTML depuis l'Interface de Configuration :

Exemple de retour de sonde en HTML depuis l'Interface de Visualisation :

Les données de performance

Dans la sortie d'une sonde, les données de performance se trouvent après le caractère | , selon le format : **TEXT OUTPUT | OPTIONAL_PERFDATA** . Ces données sont facultatives. Une sonde peut retourner une ou plusieurs données de performance.

Les données de performance sont sauvegardées dans la base de données Graphite (*voir la page* Base de métrologie (Graphite))

Les données de performance sont accessibles depuis le widget Graphique (*voir la page* Widget Graphique), l'onglet Graphiques (*voir la page* Onglet Graphiques) et les outils tiers capables de traiter les données de Graphite. Shinken assure la compatibilité avec Grafana (*voir la page* Grafana - v8.3.2)

Syntaxe des données de performance

La syntaxe de Shinken se rapproche de la syntaxe des données de performance utilisée par Nagios (*voir la documentation de nagios* https://nagios-plugins.org/doc/guidelines.html#AEN200). Contrairement à Nagios, Shinken ne gère pas les intervalles pour les seuils critique et d'avertissement. Par conséquent, les intervalles de seuils d'une sonde compatible Nagios ne seront pas pris en charge par Shinken.

La syntaxe d'une donnée de performance est la suivante : **'label'=valueUNIT;warning;critical;minimumvalue;maximalvalue**

Les différents champs sont séparés par un point virgule (;).

Exemple

```
Ok: | 'metric'=14;90;9
```

Nom	Explication	Règle syntaxique	Commentaire
label	Nom de la métrique	**Caractères interdits :** le signe égal (=) la barre verticale (|) et les guillemets (' ou ") **Caractères transformés :** Les caractères non alphabétiques ou numériques sont automatiquement convertis en un underscore (_) par Shinken. **Caractères non modifiés :** Les caractères alphabétiques et les nombres sont conservés pas Shinken	Lorsque, après la conversion des caractères, deux noms de métriques sont identiques, ils sont fusionnés et leurs données sont mélangées. (*Si metric~ et metric! sont converties en metric_, leurs données sont mélangées.*)
value	Valeur de la métrique	Accepte les entiers ou réels, et peut être précédé d'un signe plus ou moins (+ ou -). Il est possible d'écrire les puissances de 10 à l'aide du symbole 'e' ou 'E' (par exemple 1e3 pour 1000)	
UNIT	Unité	Accepte les caractères alphabétiques, les chiffres et le caractère %	Il faut préciser l'unité que pour la value. Pour l'instant cette valeur n'est pas sauvegardée par Shinken.
warning	Seuil de warning affiché	Accepte les entiers ou réels, et peut être précédé d'un signe plus ou moins (+ ou -). Il est possible d'écrire les puissances de 10 à l'aide du symbole 'e' ou 'E' (par exemple 1e3 pour 1000)	S'affiche dans les graphiques des interfaces.
critical	Seuil de critique affiché	Accepte les entiers ou réels, et peut être précédé d'un signe plus ou moins (+ ou -). Il est possible d'écrire les puissances de 10 à l'aide du symbole 'e' ou 'E' (par exemple 1e3 pour 1000)	S'affiche dans les graphiques des interfaces.
minimumvalue	Valeur minimale que pourra retourner la sonde	Accepte les entiers ou réels, et peut être précédé d'un signe plus ou moins (+ ou -). Il est possible d'écrire les puissances de 10 à l'aide du symbole 'e' ou 'E' (par exemple 1e3 pour 1000)	Pour l'instant cette valeur n'est pas sauvegardée par Shinken.
maximalvalue	Valeur maximale que pourra retourner la sonde	Accepte les entiers ou réels, et peut être précédé d'un signe plus ou moins (+ ou -). Il est possible d'écrire les puissances de 10 à l'aide du symbole 'e' ou 'E' (par exemple 1e3 pour 1000)	Pour l'instant cette valeur n'est pas sauvegardée par Shinken.

Pour renvoyer plusieurs données de performance, les données doivent se suivre, séparées par des espaces.

Exemple :

```
Ok: | 'metric_1'=14;90;95 'metric_2'=44;90;95
```

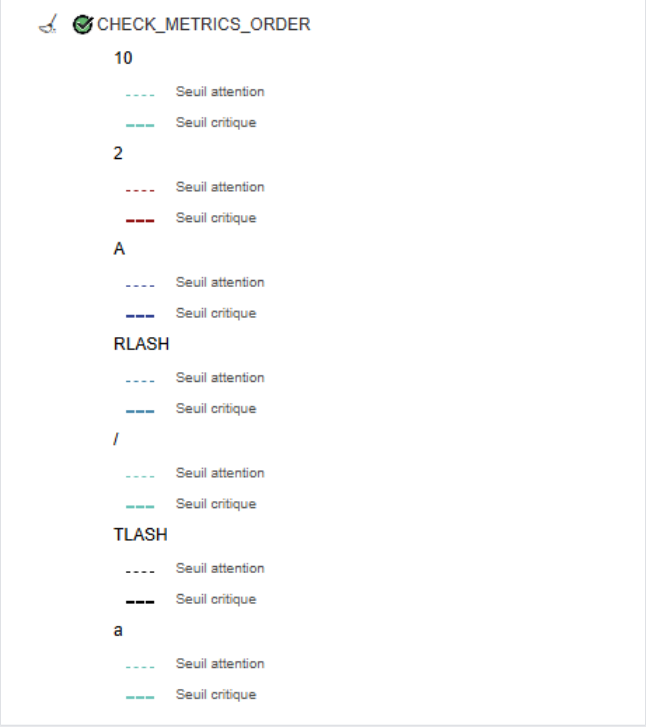
Si le nom de la métrique (*label*) change, une nouvelle métrique sera sauvegardée. L'ancienne métrique est conservée dans la base de données, ce qui fera apparaître à la fois l'ancienne et la nouvelle métrique dans les graphiques.

Affichage dans les interfaces

Dans l'Interface de Visualisation, l'ordre des métriques est constant et suit l'ordre lexicographique. Une particularité : le caractère slash (/) est traité comme la chaîne de caractères SLASH.

Grafana préserve le même ordre, mais le slash (/) est traduit en SLASH dans l'interface (*le label /var est traduit en SLASHvar*).

Exemple de l'ordre d'affichage dans l'Interface de Visualisation (*le / étant traité comme la chaine de caractère SLASH se trouve lexicographiquement entre RLASH et TLASH*) :



Quelques exemples

```
Ok: | 'metric~'=14;90;95 'metric!'=44;90;95
```

- Les deux métriques sont fusionnés en "metric_" et les données sont mélangées. Un seul graphique sera visible depuis l'Interface de Visualisation

```
Ok: | 'shinken-broker [ master ]'=14;90;95
```

- Le label sera transformé en shinken-broker___master___

```
Ok: | 'shinken-broker [ master ]'=45.5B;70;80
```

- Il est possible de préciser l'unité de la métrique

```
Ok: | 'shinken-broker [ master ]'=-7;-25;-5
```

- Il est possible d'avoir des métriques négatives

Taille maximum du retour d'une sonde

Shinken Enterprise va seulement lire les premiers 64 KB de données qu'une sonde retourne. Cela évite de surcharger la base de données et le flux d'échange entre les démons.

Exemple de sonde écrite en python

Voici un exemple simple et basique d'une sonde écrite en Python afin de vérifier la présence du fichier /tmp/test_file :

```
#!/usr/bin/env python

import os
import sys

# Shinken return values
shinkenRetValOk = 0
shinkenRetValWarn = 1
shinkenRetValCritical = 2

try:
    os.stat("/tmp/test_file")

except:
    print "CRITICAL! Unable to open test_file"
    sys.exit(shinkenRetValCritical)

print "OK! The file /tmp/test_file exists"
sys.exit(shinkenRetValOk)
```

Exemple de sonde écrite en VBS pour un Poller Windows

Voici un exemple simple et basique d'une sonde check_file_exists.vbs écrite en VBS afin de vérifier la présence d'un fichier passé en argument (*avec retour d'une donnée de performance*):

```
Dim strfilepath
Dim wsh
Dim Perf_Data

'############################################################'
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set wsh = CreateObject("WScript.Shell")
'############################################################'

If Wscript.Arguments.Count = 1 Then
strfilepath = Wscript.Arguments(0)

If (objFSO.FileExists(strfilepath)) Then
        Perf_Data = "|'path_file_existance'=1"
        Wscript.Echo "OK: the file with path " & strfilepath & " exists" & Perf_Data
        Wscript.Quit(0)
else
        Perf_Data = "|'path_file_existance'=0"
        Wscript.Echo "CRITICAL: the file with path " & strfilepath & " does NOT exist" & Perf_Data
        Wscript.Quit(2)
End If

else
        Wscript.Echo "UNKNOWN"
        Wscript.Quit(3)
End If
```

Pour utiliser cette sonde sur un Poller sur Windows, la ligne de commande devra être définie comme tel :

```
C:\WINDOWS\SysWOW64\cscript.exe //nologo $WINPLUGINSDIR$\check_file_exists.vbs "$_HOSTFILEPATH$"
```

Récupérer des sondes

Il est possible de dupliquer les sondes présente dans les répertoires **$NAGIOSPLUGINS$** et **$PLUGINSDIR$** et les placer dans le répertoire **$USERPLUGINSDIR$** pour les personnaliser.

Beaucoup d'autres sondes sont disponibles dans la communauté Nagios, et plus particulièrement sur les sites suivants :

- Shinken Monitoring Plugins : http://www.shinken.io/
- Monitoring Plugins Project : https://www.monitoring-plugins.org
- Nagios Plugins Exchange : https://exchange.nagios.org/

Pour plus d'informations pour l'installation de pack Shinken IO, voir la page Installer un pack de supervision.

Mise en place d'une sonde dans Shinken Enterprise

Une fois la sonde réalisée ou téléchargée, il faut maintenant la mettre en place dans Shinken afin de pouvoir facilement l'utiliser via des checks appliqués à des modèles d'hôtes. Il est préférable de passer par des modèles d'hôtes et des checks dédiés à ces modèles pour pouvoir réutiliser ce check le plus simplement possible.

Mise en place de la sonde

- Créer ou télécharger une sonde via la commande wget depuis les serveurs avec les démons Poller.
- Placer la sonde dans **$USERPLUGINSDIR$** (*/var/lib/shinken-user/libexec/*) des démons Shinken Poller.

Droits d'exécution

- Donner les droits d'exécution : chmod +x monscript.py

Création de la commande Shinken

- Ouvrir l'Interface de Configuration et créer une commande "cmd_monscript" avec la ligne de commande suivante : **$USERPLUGINSDIR$\monscript.py**
- Si la sonde prend des arguments, il faut ajuster la ligne de commande. Il est possible de s'inspirer d'autres commandes.

Création d'un modèle d'hôte

- Créer un modèle d'hôte : "monmodele"

Création d'un check dédié

- Pour appliquer systématiquement le script lors de l'application de "monmodele" à un hôte, il faut créer un "check dédié à modèle d'hôte", par exemple check_monscript (*cf: le check dédié "http"*).
- Dans ce check dédié, mettre "generic-service" dans la propriété "Modèle de Check hérité" et mettre le modèle "monmodèle" dans la propriété "Attaché sur les modèles d'hôte"
- Faire pointer la commande de vérification vers la commande cmd_monscript

Application du modèle et évaluation

- Appliquer le modèle "monmodèle" à des hôtes, le check doit bien être appliqué, il est possible d'essayer ce check depuis l'onglet "Checks"