

Dupliquer des checks en fonction d'une liste de valeurs présentes dans la Donnée d'un hôte (duplicate_foreach)

Sommaire

[Introduction](#)
[Fonctionnement](#)
 [Les checks](#)
 [Cas d'erreur](#)
 [L'hôte](#)
 [Cas d'erreurs](#)
 [Syntaxe de la valeur "Duplicate for each":](#)
 [Exemple de donnée "Duplicate for each":](#)
 [Les valeurs](#)
 [Syntaxe des fichiers de configuration](#)

Introduction

Saisissez votre tâche ici, en utilisant "@" pour l'attribution à un utilisateur et "/" pour sélectionner une date d'échéance

La fonctionnalité de duplication des [Checks](#) par donnée de l'[Hôte](#) (ou "duplicate foreach") permet de dupliquer très simplement des checks sur un même hôte grâce à une donnée dédiée à cet effet.

- En effet, si un hôte héberge plusieurs services de même type (par exemple, plusieurs bases de données) il peut être intéressant de dupliquer autant de fois l'ensemble des checks de monitoring qu'il y a de base de données.
- Un modèle de donnée ne s'applique qu'à un hôte dans son ensemble et ne peut pas y être dupliqué. Pour pouvoir dupliquer les checks d'un hôte, il faut utiliser des checks ayant la propriété "Duplicate for each" (Dupliqué pour chaque valeur de la Donnée de l'hôte).
- Nous verrons dans ce chapitre la façon dont ils seront dupliqués sur cet hôte, et notamment comment obtenir des données spécifiques à chaque copie, qui seront récupérées dans cette [Variable](#) (propriété) de Duplicate for each ou via les arguments par défaut.

Fonctionnement

Les checks

Pour qu'un check soit dupliqué, il doit remplir plusieurs conditions. Prenons le cas d'une commande qui prend deux arguments tel que le check_dummy.

Commande d'exemple "cmd-dummy"

The screenshot shows the Nagios configuration interface for a command named 'cmd-dummy'. The interface has a yellow header bar with 'Staging > Commande'. Below this, there's a table with two columns: 'Propriété' and 'Valeur'. The table has two rows: 'Nom de la Commande' with the value 'cmd-dummy', and 'Ligne de Commande' with the value '/usr/lib64/nagios/plugins/check_dummy \$ARG1\$ "\$ARG2\$"'. The 'Ligne de Commande' row is highlighted in blue.

Propriété	Valeur
Nom de la Commande	cmd-dummy
Ligne de Commande	/usr/lib64/nagios/plugins/check_dummy \$ARG1\$ "\$ARG2\$"

Voici comment le check ou un modèle de check sera défini :

- **(1) Obligatoire** - Le nom du check doit contenir **\$KEY\$**.
Cette variable sera remplacée par les différentes valeurs de la donnée de Duplicate For Each de l'hôte, de manière à ne pas générer de collision de nom.
- **(2) Optionnel** - Le champ "Args" permet de passer des arguments séparés entre !
Attention, particularité, pour des arguments lors d'un Duplicate For Each, les Variables sont **\$VALUEn\$**, "n" allant de 1 à 16.
- **(3) Obligatoire** - Le champ "Duplicate for each key name" doit renseigner **le nom** de la donnée qui sera définie dans l'hôte. C'est la donnée qui servira à dupliquer le check.

- **(4)** Optionnel - Le champ "Duplicate for each default arguments" permet de spécifier les valeurs par défaut pour les Variables \$VALUEn\$.

Exemple de syntaxe (si la syntaxe n'est pas respectée, un message d'erreur s'affichera) :

```
$(ma_valeur_par_défaut_1)$$$(ma_valeur_par_défaut_2)$
```

Staging > Check appliqué à l'hôte

Check appliqué à l'hôte > check-dummy-\$KEY\$ 1

Général *	Propriété	Valeur	Venant des modèles
Données [0]	Période de DOWNTIME récurrente ?	Par défaut [Même comportement que son hôte] ▼	
Supervision *	Période de vérification ?	Par défaut [Même comportement que son hôte] ▼	
Notifications	Commande de vérification * ?	cmd-dummy ▼	☐ Pas d'héritage no-check [Modèle generic-service]
Expert	Args 2 ?	\$VALUE\$ \$VALUE2\$	
	Nb maximum de tentatives de confirmation du statut du check. *	2 [Dans le modèle generic-service]	☒ Hérite du modèle 2 [Modèle generic-service]
	Intervalle entre les vérifications (/minutes) ?	5 [Dans le modèle generic-service]	☒ Hérite du modèle 5 [Modèle generic-service]
	Intervalle de nouvelles tentatives de confirmations d'état (/minutes) *	1 [Dans le modèle generic-service]	☒ Hérite du modèle 1 [Modèle generic-service]
	Temps maximum d'exécution d'un check (/secondes) ?	Par défaut [Même comportement que son hôte]	
	Seuil d'alerte de l'utilisation CPU (/secondes) ?	Par défaut [Même comportement que son hôte]	
	Les commandes de vérifications sont ordonnancées et lancées par Shinken ?	☒ Vrai ☐ Faux	☒ Hérite du modèle Vrai [Modèle generic-service]
	Shinken accepte les états reçus depuis des outils externes pour cet hôte ?	☒ Vrai ☐ Faux	☒ Hérite du modèle Vrai [Modèle generic-service]
	Duplicate for Each 3 ?	STATUS	
	Valeur par défaut des arguments du Duplicate for each 4 ?	\$(0)\$ \$(Dummy OK)\$	

Le check peut donc utiliser, dans sa configuration, **\$KEY\$**, et **\$VALUE1\$** jusqu'à **\$VALUE16\$**.

Ces variables seront remplacées de façon spécifique à chaque copie.

Plus d'information sur les variables [ici](#) avec le cas spécifique du "Duplicate For Each".

Cas d'erreur

Le champ "Duplicate for each key name" renseigné dans le paragraphe précédent (3) ne doit pas contenir le nom d'une donnée chiffrée.

Ces noms de données sont définis via la commande `shinken-protected-fields-data-manage`.

Si ce champ contient un des noms précédemment définis, un avertissement apparaîtra indiquant que les checks ne seront pas générés.

Temps maximum d'exécution d'un check <i>(secondes)</i>	Par défaut [Même comportement que son hôte]
Seuil d'alerte de l'utilisation CPU <i>(secondes)</i>	Par défaut [Même comportement que son hôte]
Les commandes de vérifications sont ordonnancées et lancées par Shinken	☒ Avertissements 1. PASSWORDSSS contient un mot-clé déclenchant le chiffrement ! Un mot-clé chiffré est considéré comme une donnée protégée ce qui ne permet pas d'effectuer la génération de check "Duplicate for Each".
Shinken accepte les états reçus depuis des outils externes pour cet hôte	☐ Vrai
1 Duplicate for Each Dupliqué pour chaque valeur de la Donnée de l'hôte	☒ PASSWORDSSS
Valeur par défaut des arguments du Duplicate for	

L'hôte

Pour que l'hôte duplique ses checks, il faut renseigner la valeur de la donnée "Duplicate for each". Ce champ est obligatoire, en cas d'oubli la donnée apparaît en erreur et aucun check n'est généré.

The screenshot shows the Nagios XI interface for host configuration. A modal error box is displayed with the following text:

Syntaxe invalide:
▲ Erreurs
1. La clé N° 1 est vide.

Chaque valeur Duplicate Foreach est présentée sous la forme suivante: CLE\$(ARG1)\$\$(ARG2)\$...\$(ARGn)\$
(CLE obligatoire, arguments facultatifs)
Les valeurs sont séparées entre elles par des virgules.

The background interface shows the 'Données locales' (Local Data) section for host 'srv-oracle-2'. It lists several checks: 'STATUS', 'host-critical', 'generic-host', and 'customer2'. The 'host-critical' check is highlighted, showing it is used by the 'STATUS' check.

Un hôte ayant des checks à dupliquer qui lui sont directement attachés ou provenant de modèles hérités, doit obligatoirement fournir des valeurs dans sa donnée de Duplicate For Each.

Lors de l'édition, chaque clé peut être spécifiée avec ses valeurs sur une ligne séparée.

The screenshot shows the 'Données locales' (Local Data) section for host 'srv-oracle-2'. The 'host-critical' check is selected, and the 'Duplicate for each' field is visible. The field contains the following multi-line value:

```
ok,  
warning$(Shinken)$,  
critical$(Shinken Entreprise)$$(testing)$
```

The interface also shows the 'Venant des modèles' (Inherited from models) section, which lists the 'host-critical' model and its associated checks.

Lorsque plusieurs checks utilisent la même donnée Duplicate For Each (ex: *BROKER_LIST* dans le screen ci-dessous), celle-ci n'apparaît qu'une fois et sa valeur est partagée pour tous les checks.

Données locales & héritées d'un modèle

Nom	Valeur
BROKER_LIST	broker-master\$(\$HOSTBROKER_PORTS)\$ [Dans le modèle shinken-broker]

Utilisé par les checks [4 / 16]

- Broker - \$KEY\$ - Alive
- Broker - \$KEY\$ - Performance API Connection
- Broker - \$KEY\$ - Performance Modules Queues
- Broker Daemon Module SLA - \$KEY\$

Non utilisé par les checks [12 / 16]

- Synchronizer - \$KEY\$ - Alive
- Synchronizer - \$KEY\$ - Performance API Connection
- Receiver - \$KEY\$ - Alive
- Receiver - \$KEY\$ - Performance API Connection
- Reactionner - \$KEY\$ - Performance
- Reactionner - \$KEY\$ - Running Well
- Poller - \$KEY\$ - Performance
- Poller - \$KEY\$ - Running Well
- Arbiter - \$KEY\$ - Alive
- Arbiter - \$KEY\$ - Performance
- Scheduler - \$KEY\$ - Performance
- Scheduler - \$KEY\$ - Running Well

(Duplicate Foreach)

- shinken-arbiter [1]
- shinken-broker [1]
- shinken-poller [1]
- shinken-reactionner [1]
- shinken-receiver [1]
- shinken-scheduler [1]
- shinken-synchronizer [1]
- shinken-broker-module-sla [0]

Cas d'erreurs

Si la donnée définie est une donnée chiffrée (voir [Cas d'erreur](#)), un avertissement apparaîtra dans la liste des données.

Général

- Données [3 / 7]
- Droits de l'utilisateur
- Supervision
- Checks [3]
- Notifications
- Expert

Données locales

Nom	Valeur
PASSWORDSSS	

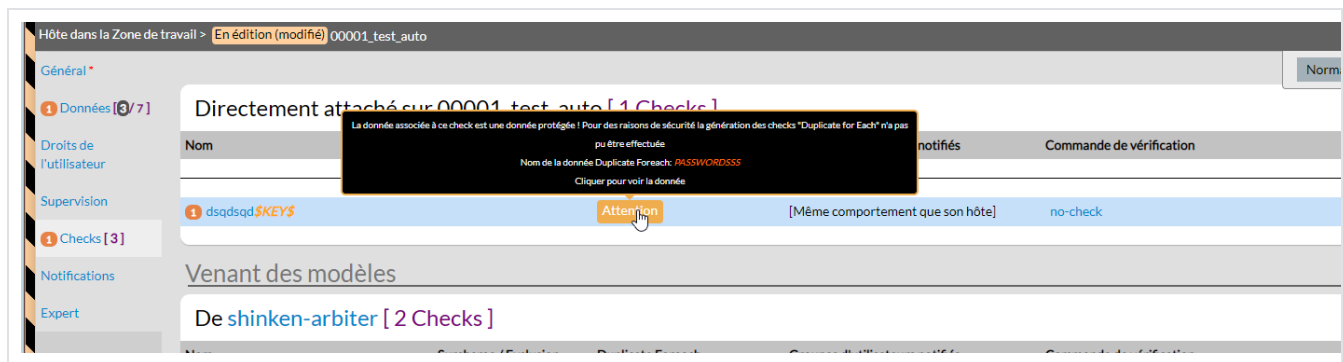
Utilisé par le check [1 / 3]

Non utilisé par les checks [2 / 3]

⚠ Avertissements

1. Cette donnée est protégée !
 Pour des raisons de sécurité la génération des checks "Duplicate for Each" a été désactivée dans l'interface de configuration

Les checks ne seront alors pas générés et un message "Attention" apparaîtra dans la liste des checks.



Syntaxe de la valeur "Duplicate for each":

La valeur de la donnée Duplicate For Each a une syntaxe précise:

```
ma_cle1$(mon_arg1_cle1)$$(mon_arg2_cle1)$
```

On définit le nom de la clé et les arguments sont passés entre \$(et)\$ de manière consécutive. Les arguments sont facultatifs (s'ils ne sont pas renseignés, les arguments par défaut pourront être utilisés).

Pour renseigner plusieurs clés, on reprend la même syntaxe séparée par des virgules:

```
Exemple 1 : ma_cle1$(mon_arg1_cle1)$$(mon_arg2_cle1)$, ma_cle2$(mon_arg1_cle2)$$(mon_arg2_cle2)$,
ma_cle3$(mon_arg1_cle3)$$(mon_arg2_cle3)$
Exemple 2 : ma_cle1, ma_cle2$(mon_arg1_cle2)$$(mon_arg2_cle2)$, ma_cle3$(mon_arg1_cle3)$
```

Dès que la saisie du champ est terminée, le nombre de checks de l'hôte est mis à jour.

Exemple de donnée "Duplicate for each":

```
ok,warning$(1)$$(Dummy WARNING)$,critical$(2)$$(Dummy CRITICAL)$,unknown$(3)$$(Dummy UNKNOWN)$
```

Dans cet exemple, la première valeur/clé "ok" n'a aucun argument défini, la valeur par défaut des arguments sera alors utilisée, à savoir, 2 arguments dans notre exemple :

```
$(0)$$(Dummy OK)$
```

Pour la valeur/clé "warning", cette clé a deux arguments "1" et "Dummy WARNING".

Plus de détails sont dans la prochaine section "Les valeurs".

Hôte dans la Zone de travail > Serveur-Duplicate

Général *

Données [1 / 2]

Droits de l'utilisateur

Supervision

Checks [12]

Notifications

Expert

Données locales & héritées d'un modèle

Locale

Locale [0]

Venant des modèles

testpack-host-tpl [0]

(Duplicate Foreach)

testpack-host-tpl [1 / 2]

Nom	Valeur
STATE_LOOP_LIST	simple,custom\$(1:2)\$ [Dans le modèle testpack-host-tpl]
Utilisé par les checks [1 / 8]	Non utilisé par les checks [7 / 8]
STATUS	ok,warning\$(1)\$\$(Dummy WARNING)\$,critical\$(2)\$\$(Dummy CRITICAL)\$,unk
Utilisé par les checks [1 / 8]	Non utilisé par les checks [7 / 8]

L'onglet check permet de vérifier le résultat de la duplication.

Les checks obtenus ont des configurations identiques. Pour pouvoir leur donner des données spécifiques, il faudra utiliser les valeurs.

On voit ici que le check a bien été dupliqué pour les 4 valeurs de données "Duplicate for each".

Pour chaque duplication, les arguments sont bien récupérés pour le passage dans la commande. (Via les Variables spéciales \$VALUEn\$ qui correspondent aux différents \$ARGn\$)

Hôte dans la Zone de travail > Serveur-Duplicate

Général *

Données [1 / 2]

Droits de l'utilisateur

Supervision

Checks [12]

Notifications

Expert

De testpack-host-tpl [12 Checks]

Nom	Duplicate Foreach	Groupe d'utilisateur	Modèle d'hôte	Command	Essayer ce check
check-dummy-ok	ok	[Même comportement que son hôte]	testpack-host-tpl	cmd-dummy!0!Dummy OK	
check-dummy-warning	warning	[Même comportement que son hôte]	testpack-host-tpl	cmd-dummy!1!Dummy WARNING	
check-dummy-critical	critical	[Même comportement que son hôte]	testpack-host-tpl	cmd-dummy!2!Dummy CRITICAL	
check-dummy-unknown	unknown	[Même comportement que son hôte]	testpack-host-tpl	cmd-dummy!3!Dummy UNKNOWN	

Evaluation de commande

check-dummy-ok

STATUS

ok

[Même comportement que son hôte]

testpack-host-tpl

cmd-dummy!0!Dummy OK

Ligne de commande:

```
/usr/lib64/nagios/plugins/check_dummy $ARG1$ "$ARG2$"
```

Évaluation:

Arguments:

Nom	Valeur étendu	Trouvé dans (type)	Trouvé dans (nom)
ARG1	VALUE1	Arguments de la commande	
ARG2	VALUE2	Arguments de la commande	
VALUE2	Dummy OK	duplicate foreach (value)	._STATUS
VALUE1	0	duplicate foreach (value)	._STATUS

Commande avec les données interprétées:

```
/usr/lib64/nagios/plugins/check_dummy 0 "Dummy OK"
```

Les valeurs

Pour que chaque check puisse exécuter une vérification différente sur l'hôte, il est possible de paramétrer spécifiquement chaque copie.

Pour cela, l'hôte peut passer des arguments de duplication à chaque copie sous la forme de valeurs dans la donnée de "Duplicate For Each".

Check	Variables	Valeur
-------	-----------	--------

Par exemple, pour une commande qui demande deux arguments, la syntaxe de la donnée de "Duplicate for each" est la suivante :

```
red$(IndianRed)$$(CD5C5C)$, green$(ForestGreen)$,
blue
```

 **Remarque:** Les arguments manquants seront remplacés par les valeurs par défaut.

Check-Duplicate-red	\$KEY\$	red
	\$VALUE1\$	IndianRed
	\$VALUE2\$	CD5C5C
Check-Duplicate-green	\$KEY\$	green
	\$VALUE1\$	ForestGreen
	\$VALUE2\$	default_value2
Check-Duplicate-blue	\$KEY\$	blue
	\$VALUE1\$	default_value1
	\$VALUE2\$	default_value2

Syntaxe des fichiers de configuration

Voici dans notre exemple, la définition d'un check Duplicate For Each qui va s'appliquer à un modèle d'hôte:

```
define service {
    service_description    check-dummy-$KEY$
    register                0
    host_name              testpack-host-tpl
    use                    check-dummy-tpl
    duplicate_foreach      _STATUS
    default_value          $(0)$$(Dummy OK)$
}

define host {
    name                  testpack-host-tpl
    check_command          check-host-alive
    check_interval         1
    retry_interval         1
    register               0
    _STATUS                ok,warning$(1)$$(Dummy WARNING)$,critical$(2)$$(Dummy CRITICAL)$,unknown$(3)$$(Dummy UNKNOWN)$
}
}
```