

Dupliquer pour chaque valeur de la Donnée de l'hôte (duplicate_foreach)

Introduction

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

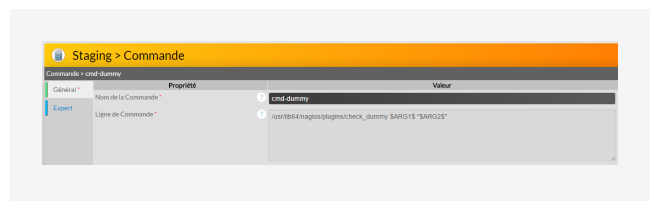
- En effet, si un hôte héberge plusieurs services de même type (par exemple, plusieurs bases de données) il peut être intéressant de dupliquer autant de fois l'ensemble des checks de monitoring qu'il y a de base de données.
- Un template donné ne s'applique qu'à un hôte dans son ensemble et ne peut pas 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 [Macro](#) (donnée) 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"



Voici comment le check ou check template sera défini :

- (1) Obligatoire** - Le nom du check doit contenir **\$KEY\$**. Cette macro 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 macros 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 macros \$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)$
```

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

Ces macros seront remplacées de façon spécifique à chaque copie.

Plus d'information sur les macros [ici](#) avec le cas spécifique du "Duplicate for each".

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.

Source	Nom	Valeur	Venant des modèles
Locale [0]			
testpack-host-tpl [0]	STATE_LOOP_LIST	simple.custom\$(1:2)\$ [Dans le modèle testpack-host-tpl]	Hérite du template simple.custom\$(1:2)\$
shinken-full [35]		Utilisé par les checks [1 / 8]	Non utilisé par les checks [7 / 8]
linux [1/16]	STATUS	ok.warning\$(1)\$(Dummy WARNING)\$.critical\$(2)\$(Dummy CRITICAL)\$.unknown\$(3)\$(Dummy UNKNOWN)\$	Hérite du template ok.warning\$(1)\$(Dummy WARNING)\$.critical\$(2)\$(Dummy CRITICAL)\$.unknown\$(3)\$(Dummy UNKNOWN)\$
enable-notifications-5mn [0]			
(Duplicate Foreach)			
testpack-host-tpl [2]			

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.

Modèle d'hôte > testpack-host-tpl

Données locales & héritées d'un modèle

Source	Nom	Valeur
(Duplicate Foreach)	STATE_LOOP_LIST	simple.custom\$(1:2)\$
Locale [1/2]	STATUS	

Système invalide

Erreurs

- 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.

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".



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

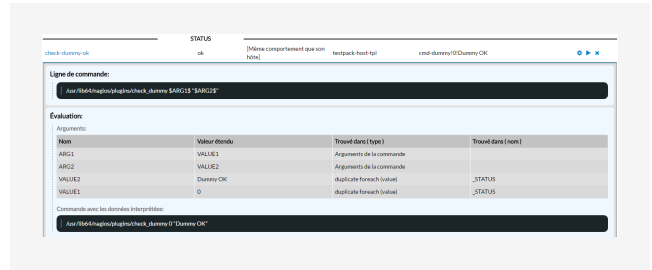
Les checks obtenus ont des configurations identiques. Pour pouvoir leur donner des données spécifiques, il faudra utiliser les valeurs.

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 [Macros](#) spéciales \$VALUEn\$ qui correspondent aux différents \$ARGn\$)

Nom	Duplication	Groupe d'instance	Modèle d'hôte	Commande	Essayer ce check
check-dummy-ok	2	ok	testpack-host-tp1	cmd=runner(1)DummyOK	+
check-dummy-warning	2	warning	testpack-host-tp1	cmd=runner(1)DummyWARNING	+
check-dummy-critical	2	critical	testpack-host-tp1	cmd=runner(2)DummyCRITICAL	+
check-dummy-unknown	2	unknown	testpack-host-tp1	cmd=runner(3)DummyUNKNOWN	+

Evaluation de commande



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".

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	Macro	Valeur
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)$
}
```