Broker - KEY - Module Visualisation UI - SLA Reader

Qu'est ce qu'un cluster ?

Son rôle principal est de permettre d'avoir dans un seul indicateur l'état agrégé d'autres états. Cet indicateur propose une vue unique pour des utilisateurs jouant différents rôles.

Rôles typiques:

- Service delivery Management
- Business Management
- Engineering
- IT support

Prenons l'exemple d'un rôle de service "delivery" pour une application ERP . Il est principalement constitué de:

- 2 bases de données, en haute disponibilité, donc avec une seule base active, le service est considéré comme rendu.
- 2 serveurs web, en partage de charge, donc avec un web serveur actif, le service est considéré comme rendu.
- 2 serveurs de répartition, encore en haute disponibilité.

Ces composants (Hôtes dans cet exemple) seront la base de ce service ERP .

Avec des règles métier, vous pouvez avoir une vue agrégée de l'indicateur représentant l'état du service ERP !

Shinken Enterprise vérifie chaque composant un par un pour l'analyse de problème source.

Accéder à la configuration du Cluster

L'accès à la configuration des Clusters se fait par le menu "Staging", disponible dans la barre de navigation.

? Unknown Attachment

Ce lien donne accès à la liste des Clusters et des modèles de Cluster, permettant d'en créer de nouveaux ainsi que de les modifier.

Le nom du cluster ne doit pas exister en tant que nom d'hôte.
Si tel est le cas, une erreur sera affichée à la sauvegarde.

Logique de notification

Un Cluster a la même logique de notification que les hôtes.

Lors de la création/édition d'un Cluster, les notifications peuvent être configurées via l'onglet Notifications de la même manière que pour un hôte.

La documentation sur les notifications fournit des informations détaillées sur le sujet: Logique de notification

Remarque Importante

La propriété "Décalage de la première notification" (**first_notification_delay**) est très importante pour la gestion des notifications.

En effet, cette propriété permet de définir combien de minutes Shinken doit attendre avant d'envoyer la première notification.

Ce temps additionnel peut être utilisé pour limiter une avalanche de notifications ; en effet, les clusters n'ont pas de gestion HARD/SOFT et leur état est donc susceptible de changer plus fréquemment.
Ce temps additionnel peut être mis à profit par les utilisateurs pour prendre en compte le cluster depuis l'interface de visualisation avant que la notification ne soit envoyée.

Par défaut la valeur est 0, ceci signifie que la première notification sera envoyée sans attendre.

Voici un exemple d'email reçu lors de décalage de la première notification :

? Unknown Attachment

Définition minimaliste d'un Cluster

Un Cluster va voir son état dépendre de ceux de ses éléments. Ces éléments peuvent être des Hôtes, des Checks, ou un mélange des deux. La définition des éléments du Cluster se fait par l'intermédiaire de la *bp_rule.*

La manière la plus simple de définir un Cluster est de lister un à un les Hôtes/Checks et de spécifier avec des opérateurs de logique booléenne le comportement du Cluster.

Deux opérateurs de bases sont disponibles pour définir l'agrégation des états: & et |

Elements avec des caractères spéciaux

Dans le cas où, des caractères spéciaux sont utilisés dans le nom des éléments, il est possible de les écrire sous la forme :

```
"hôte-1" & "hôte-2"
```

? Unknown Attachment

L'opérateur & semblable au ET classique renvoie l'état le plus bas des 2 éléments.

Par exemple, si un hôte *hote_a* est OK et qu'un hôte *hote_b* est Critique, renverra Critique.

```
hote_a & hote_b
```

L'opérateur | semblable au OU classique renvoie l'état le plus haut des 2 éléments.

Par exemple, si un hôte *hote_a* est OK et qu'un hôte *hote_b* est Critique, renverra OK.

```
hote_a | hote_b
```

Dans un cluster, on peut sélectionner 2 types d'éléments: des hôtes et des checks.

L'état d'un hôte est récupéré avec seulement le nom de l'hôte.

L'état d'un check est récupéré avec le nom de l'hôte, puis le nom du check, séparé par une virgule. Par exemple :

```
hote_a,service_a
```

Aussi, il est possible de configurer la priorité des opérateurs booléens avec des parenthèses.

Exemple de configuration d'une règle pour un cluster ERP

```
(srv-oracle-1 | srv-oracle-2) & (srv-http-1 | srv-http-2) & (srv-loadbalancer-1 | srv-loadbalancer-2)
```

Affichage des erreurs

Les erreurs de configuration de la propriété "Définition" sont affichées lors de la sauvegarde de l'élément.

Dans chaque message d'erreur, les caractères provoquant l'erreur sont précisés, et la définition du cluster est affichée avec ces caractères affichés en rouge.

Les définitions suivantes sont détectées :

- guillemets vides ("")
- caractères interdits dans un contexte particulier (< >)

Erreurs non détectées

Une définition ne respectant pas les règles décrites ci-dessous ne provoquera pas de message d'erreur lors de la sauvegarde, mais dans l'interface de visualisation, ces clusters seront considérés comme vides.

? Unknown Attachment

Utiliser les négations

On peut également vouloir vérifier qu'un hôte ou un service renvoie un statut Critique. Cela peut être utile dans le cas de configuration avec des éléments actifs et passifs par exemple.

Le caractère **!** permet de prendre l'inverse de l'état de l'élément. Ainsi, on peut définir des règles comme la suivante:

```
(srv-oracle-1 & !srv-oracle-2)
```

Le cluster sera donc en statut OK si l'hôte srv-oracle-1 est OK et l'hôte srv-oracle-2 est Critical.

Groupement d'expressions

Parfois, on ne souhaite/peut pas spécifier précisément les hôtes contenus dans une règle. On peut dans les bp_rule grouper les éléments selon différents critères. On peut par exemple récupérer tous les hôtes appartenant à un certain groupe d'hôte, ayant un certain modèle d'hôte accroché, ou encore récupérer les hôtes dont le nom remplit certains critères. Ces groupements sont disponibles également pour les checks.

Pour cela, il est possible d'utiliser des expressions groupées suivant cette syntaxe :

```
flag:expression
```

Le flag est un simple caractère qualifiant le type d'expansion.
Les types supportés sont décrits dans les tables ci-dessous.

Flags pour les hôtes

Flag	Signification	Exemple	Equivalence
g	Hôtes appartenant au groupe d'hôtes	g:web	web-srv1 & web-srv2 & ...
r	Hôtes dont le nom matche la regex	r:^web	web-srv1 & web-srv2 & ...
t	Hôtes ayant le modèle d'hôte	t:http	web-srv1 & web-srv2 & ...
tr	Hôtes ayant le modèle d'hôte qui matche la regex	tr:^http	web-srv1 & web-srv2 & ...

Flags pour les checks

Flag	Signification	Exemple	Equivalence
r	Checks dont le nom match la regex	r:^HTTPS?	web-srv1,HTTP & db-srv2, HTTPS & ...
t	Checks ayant le modèle de check	t:http	web-srv1,HTTP & db-srv2, HTTPS & ...
tr	Checks ayant le modèle de check qui matche la regex	tr:^http	web-srv1,HTTP & db-srv2, HTTPS & ...

⚠ **Regex complexes**

Dans le cas où des caractères spéciaux doivent être utilisés dans le contenu de la regex, il est possible d'écrire la regex sous la forme:

```
r:/regex/
```

Les caractères spéciaux doivent être échappés avec un anti-slash.

Exemple d'expressions combinées

Le calcul du statut d'un élément groupé se fait avec l'opérateur ET (&). Par exemple, la règle suivante:

```
r:srv-oracle-.*
```

sera équivalente à

```
srv-oracle-1 & srv-oracle-2
```

Si vous souhaitez créer une règle incluant tous les web serveurs composant l'application frontend.

```
t:http,r:HTTPS?
```

qui est équivalent à:

```
web-srv1,HTTP & web-srv3,HTTPS
```

Vous devriez donc combiner l'expansion d'expression avec :

```
t:http,r:HTTPS? & db-srv1,MySQL
```

qui est équivalent à:

```
(web-srv1,HTTP & web-srv3,HTTPS) & db-srv1,MySQL
```

Ensembles d'éléments

On voudrait également pouvoir sélectionner les éléments qui remplissent plusieurs critères différents. C'est possible en définissant des ensembles.

Les ensembles sont définis en enveloppant une expression de crochets:

```
[ definition de l'ensemble ]
```

Dans les ensembles, les opérations booléennes **&** et **|** n'existent pas, et sont remplacées par **<and>** et **<or>**. Le tableau ci-dessous récapitule les significations des différents opérateurs dans les ensembles.

Opérateur	Sens de l'opération
<or>	Union
<and>	Intersection
<and not>	Exclusion (privé de)

Les ensembles peuvent être imbriqués pour permettre de définir des priorités d'évaluation.
Les opérateurs sont évalués dans l'ordre des sous-parties, puis de gauche à droite.

Ainsi, l'expression

```
[ t:pop <or> tr:imap.* <and not> [t:pop <and> tr:imap.* ]]
```

regroupe les hôtes ayant le modèle pop, ou un modèle commençant par imap, mais pas les deux.

La règle Xof

Dans certains cas, un cluster de N éléments nécessite d'avoir au moins X d'entre eux en état OK pour être considéré comme OK.

Pour spécifier ce type de comportement, il suffit d'utiliser l'opérateur "X of:".

La syntaxe de cet opérateur est la suivante:

```
Xof: expression
```

L'expression **X of:** peut être configurée avec différentes valeurs en fonction du besoin. Les valeurs supportées pour **X** sont les suivantes:

- Un nombre entier positif, soit **"au moins X hôtes doivent être up"**
- Un pourcentage positif, soit **"au moins X% d'hôtes doivent être up"**.
  On peut combiner des expressions groupées telles que **"95% de mes web front ends doivent être up"**..
- Un nombre entier négatif, soit **"au plus X hôtes doivent être down"**
- Un pourcentage négatif, soit **"au plus X% d'hôtes doivent être down"**.
  On peut combiner des expressions groupées telles que **"5% de mes web front ends doivent être down"**

```
(srv-oracle-1 | srv-oracle-2) & (srv-loadbalancer-1 | srv-loadbalancer-2) & 95% of: g:frontend
```

L'expression passée à l'opérateur Xof peut également être un ensemble. On peut par exemple avoir une règle du type:

```
90% of: [ t:linux <or> tr:windows.* <and not> [t:linux <and> tr:windows.* ]]
```

Valeur négative pour X

Les valeurs négatives sont interprétées pour l'évaluation comme *MAX - X*.

Cette possibilité est utile dans le cas où X est un entier et qu'on ne connaît pas le nombre d'éléments du Cluster. On pourrait par exemple vouloir dire qu'a plus X éléments doivent être en Critical.

Cependant, dans le cas d'un pourcentage, écrire **-20%of** est seulement une manière moins lisible d'écrire **80%of**, puisque les 2 expressions auront le même résultat. Il vaut donc mieux ne pas abuser de ce type de notation.

Gérer un état dégradé

L'opérateur Xof permet de gérer un état du cluster en se basant sur les états OK et Critical des éléments. Il renvoie également un statut OK ou Critique selon son seuil (valeur du paramètre X).
Mais, on pourrait avoir envie de voir notre cluster mis en Warning selon l'état des éléments du cluster.

Les opérateurs X,Y,Zof et Xof configurable permettent d'avoir un niveau d'information plus précise sur l'état du cluster.

L'opérateur X,Y,Zof (c.a.d. => OK,WARNING,CRITICALof)

L'opérateur X,Y,Zof agit de manière similaire à l'opérateur Xof.

Il dispose de 3 paramètres X,Y et Z qui agissent comme suivant:

- X: Nombre minimal de statuts OK pour que l'opérateur renvoie OK
- Y: Nombre minimal de statuts Warning pour que l'opérateur renvoie Warning
- Z: Nombre minimal de statuts Critical pour que l'opérateur renvoie Critical

L'évaluation de la règle Xof se fait dans l'ordre suivant:

- L'état du cluster peut-il être **Critique** ? (comparaison avec **Z**)
- L'état du cluster peut-il être **Warning** ? (comparaison ***nb_critiques*** + ***nb_warning*** **avec Y**)
- L'état du cluster peut-il être **OK** ? (comparaison avec **X**)
- **Si aucun cas n'est possible, le pire état est renvoyé**.

Evaluation du cas Warning

L'évaluation du cas Warning prend en compte le nombre d'éléments en Warning auquel on ajoute le nombre d'éléments en Critical.

La règle X,Y,Zof renvoie Warning si moins de Z éléments sont en Critique et *(Nombre d'éléments en Warning + Nombre d'éléments en Critique) > Y*

Pour illustrer, prenons l'exemple d'une business rule agissant sur 5 services A,B,C,D et E: X,Y,Zof: A|B|C|D|E

Exemple 1

A	B	C	D	E
Warning	OK	OK	OK	OK

Différentes valeurs des paramètres et résultat de l'opérateur:

- 4of: OK
- 5,1,1of: Warning
- 5,2,1of: OK

Exemple 2

A	B	C	D	E
Warning	Warning	OK	OK	OK

Différentes valeurs des paramètres et résultat de l'opérateur:

- 4of: Critical
- 3of: OK
- 4,1,1of: Warning

Example 3

A	B	C	D	E
Critical	Critical	OK	OK	OK

Différentes valeurs des paramètres et résultat de l'opérateur:

- 4of: Critical
- 3of: OK
- 4,1,1of: Critical

Example 4

A	B	C	D	E
Warning	Critical	OK	OK	OK

Différentes valeurs des paramètres et résultat de l'opérateur:

- 4of: Critical
- 4,1,1of: Critical

Example 5

A	B	C	D	E
Warning	Warning	Critical	OK	OK

Différentes valeurs des paramètres et résultat de l'opérateur:

- 2of: OK
- 4,1,1of: Critical

Example 6

A	B	C	D	E
Warning	Critical	Critical	OK	OK

Différentes valeurs des paramètres et résultat de l'opérateur:

- 2of: OK
- 2,4,4of: OK
- 4,1,1of: Critical
- 4,1,2of: Critical
- 4,1,3of: Warning

Configuration classiques

Par exemple, pour un nombre MAX d'éléments:

- Setup ON/OFF: MAXof <=> MAX,MAX,MAXof
- Warning dès qu'un problème apparaît, Critical si tous les éléments sont en critique: MAX,1,MAXof
- Pire état possible: MAX,1,1

L'opérateur Rule1|Rule2|... of configurable

L'opérateur **Rule1|Rule2|... of** permet d'avoir un contrôle plus fin sur le statut du cluster mais son comportement est fixe et ne peut pas être changé.

- Il permet de définir le comportement du Cluster précisément.
- L'utilisateur peut définir plusieurs comportements sous forme de propositions successives qui seront évaluées dans l'ordre fourni.

La syntaxe de cette commande est la suivante:

- **Rule1|Rule2|...|default->*return_state* of: elements**
- avec les règles de la forme: ***X*status->*status*** *(le status retourné)*
    - *La règle est **VALIDE** si le nombre de statut dans le cluster est **EGAL** ou **SUPERIEUR** à **X***

Remarque: Il faut que vous définissiez, de la règle des pires états vers les moins pires.

Les différentes règles sont évaluées dans l'ordre dans lequel elles apparaissent.

- Si une **règle est VALIDE**, l'état spécifié est **renvoyé** et **l'évaluation s'arrête**.
- Si **aucune règle N'EST VALIDE**, l'état spécifié dans "**default**" **est renvoyé**.
- Si **aucune valeur par défaut** n'est spécifiée, la valeur renvoyée est **UNKNOWN**.

Les statuts à fournir pour les règles doivent être écrits comme ceci :

- **OK**
- **Warning**
- **Critical**
- **Unknown**

Exemples

On se place dans le cas d'un cluster à 10 éléments :

- A, B, C, D, E, F, G, H, I et J.

Business rule:

- 1Critical->Warning|2Critical->Critical|30%Warning->Warning|50%Warning->Critical|default->OK of: A|B|C|D|E|F|G|H|I|J

A	B	C	D	E	F	G	H	I	J	Condition qui déclenche le résultat	Résultat de la bp_rule
OK	Warning	Critical	Warning	OK	Warning	OK	Warning	Critical	Warning	2Critical->Critical	Critical
OK	Warning	OK	Warning	OK	Warning	OK	Warning	Warning	Warning	50%Warning->Critical	Critical
OK	Warning	OK	Warning	OK	OK	OK	OK	OK	Warning	30%Warning->Warning	Warning
OK	OK	OK	OK	OK	Warning	Warning	OK	OK	OK	default->OK	OK

Logique de notification

Un Cluster a la même logique de notification que les hôtes.

Lors de la création/édition d'un Cluster, les notifications peuvent être configurées via l'onglet Notifications de la même manière que pour un hôte.

La documentation sur les notifications fournit des informations détaillées sur le sujet: Logique de notification

Remarque Importante

La propriété "Décalage de la première notification" (**first_notification_delay**) est très importante pour la gestion des notifications.

En effet, cette propriété permet de définir combien de minutes Shinken doit attendre avant d'envoyer la première notification.

Ce temps additionnel peut être utilisé pour limiter une avalanche de notifications ; en effet, les clusters n'ont pas de gestion HARD/SOFT et leur état est donc susceptible de changer plus fréquemment.
Ce temps additionnel peut être mis à profit par les utilisateurs pour prendre en compte le cluster depuis l'interface de visualisation avant que la notification ne soit envoyée.

Par défaut la valeur est 0, ceci signifie que la première notification sera envoyée sans attendre.

Voici un exemple d'email reçu lors de décalage de la première notification :