La vue météo dans l'interface de visualisation de Shinken

Sommaire

Description

Le module **broker-module-livedata** permet de mettre à disposition sur le Broker une API HTTP permettant d'accéder aux informations d'un hôte, d'un cluster ou d'un check.

Cette API peut être utilisée pour accéder rapidement aux informations d'éléments supervisés pour ensuite les intégrer dans des outils externes (*outil de ticketing, récolte de données, etc...*).

Activation du module

Le module `broker-module-livedata` est un module qui peut être activé seulement sur le démon Broker.

- L'activation du module s'effectue en ajoutant le nom de ce module dans le fichier de configuration du démon Broker.
- Pour se faire, ouvrez le fichier de configuration du Broker à l'emplacement **/etc/shinken/brokers/nom_du_broker.cfg**, et ajouter le nom de votre module (dans notre exemple : "`broker-module-livedata`").

```
define broker {
    [...]
    modules                  Module 1, Module 2, Module 3, broker-module-livedata
    [...]
}
```

Pour prendre en compte le changement de configuration, il faut ensuite redémarrer l'Arbiter :

```
service shinken-arbiter restart
```

Configuration

La configuration du module se trouve par défaut dans le fichier suivant : **/etc/shinken/modules/broker-module-livedata.cfg**

Exemple de fichier de configuration

```
#===============================================================================
# broker module livedata
#===============================================================================
# Daemons that can load this module:
# - broker
# This module is an api getting information from the broker
#===============================================================================

define module {

    #======== Module identity =========
    # Module name. Must be unique
    module_name               broker-module-livedata

    # Module type (to load module code). Do not edit.
    module_type               broker_module_livedata

    #======== Listening address =========
    # host: IP address to listen to.
    #       note: 0.0.0.0 = all interfaces.
    host                      0.0.0.0
    # port to listen
    port                      50100

    # HTTPs part, enable if you want to set the visualisation interface listen in HTTPs mode
    # disabled by default. Set your own certificates.
    use_ssl                   0
    ssl_cert                  /etc/shinken/certs/server.cert
    ssl_key                   /etc/shinken/certs/server.key
    token                     change_me

    lang                      en

    #======== Broks getter in broker-module-livedata =========
    # These parameters allow some internal tuning in broks management in broker-module-livedata

    # Enable or disable late broks sets catchup
    # broker_module_livedata__broks_getter__activate_late_set_catchup                                1

    # Take extra broks sets to manage if more than this parameter sets are waiting
    # broker_module_livedata__broks_getter__nb_late_set_allowed_before_catchup                       10

    # Stop taking extra broks sets in catchup when we reach this number of broks
    # broker_module_livedata__broks_getter__catchup_broks_managed_by_module_in_a_catchup_loop        200000

    # Continue catchup if too much late broks sets remains after
    # broker_module_livedata__broks_getter__catchup_run_endless_until_nb_late_set_allowed_reached    0

    # Take the lock as soon as getter thread has some broks to manage
    # in order to attempt to reduce concurrent usage of CPU
    # broker_module_livedata__broks_getter__include_deserialisation_and_catchup_in_lock             0
}
```

Détails des sections composant le fichier de configuration

Identification du module

Il est possible de définir plusieurs instances de module de type "broker-module-livedata" dans votre architecture Shinken.

- Chaque instance devra avoir un nom unique.

Nom	Type	Unité	Défaut	Commentaire

`module_name`	Texte	---	**broker-module-livedata**	Nous vous conseillons de choisir un nom en fonction de l'utilisation du module pour que votre configuration soit simple à maintenir. Doit être unique.
`module_type`	Texte	---	**broker-module-livedata**	Ne peut être modifié.

Configuration de l'interface et du port d'écoute

```
#======== Listening address =========
# host: IP address to listen to.
# note: 0.0.0.0 = all interfaces.
host                        0.0.0.0
# port to listen
port                        50100
```

Il est possible de configurer l'interface réseau sur laquelle est mise à disposition l'API. Si par exemple l'API ne doit être accessible seulement via un réseau local, il est possible de n'écouter les requêtes que sur cette interface réseau.

Les paramètres suivants permettent de configurer l'accès à l'API :

Nom	Type	Unité	Défaut	Commentaire
`host`	Texte	Adresse IPv4	**0.0.0.0**	L'interface réseau sur laquelle le module "broker-module-livedata" va écouter.
`port`	Texte	Port réseau	**50100**	Port réseau sur lequel le module "broker-module-livedata" va écouter.

Sécurisation de la communication avec le module

```
# HTTPs part, enable if you want to set the visualisation interface listen in HTTPs mode
# disabled by default. Set your own certificates.
use_ssl                         0
ssl_cert                         /etc/shinken/certs/server.cert
ssl_key                         /etc/shinken/certs/server.key
token                                 change_me
```

L'API du module est accessible via HTTP. Il est recommandé d'utiliser le protocole HTTPS pour chiffrer la communication avec le module, et donc chiffrer le token d'accès.

Si pour des raisons de sécurité, cette API doit être accessible via HTTPS, il est possible de configurer les certificats avec les paramètres suivants :

Nom	Type	Unité	Défaut	Commentaire
`use_ssl`	Booléen	---	**0**	Permet d'activer ou non l'utilisation du protocole HTTPS. • **0** : Désactivé (*utilise HTTP*) • **1** : Activé (*utilise HTTPS*)
`ssl_cert`	Texte	Chemin	**/etc/shinken/certs/server.cert**	Chemin vers le certificat SSL utilisé par le protocole HTTPS.
`ssl_key`	Texte	Chemin	**/etc/shinken/certs/server.key**	Chemin vers la clé SSL utilisée par le protocole HTTPS.

`token`	Texte	---	**change_me**	Token utilisé lors des requêtes avec le module pour s'authentifier. Si `use_ssl` est à `0`, alors ce token est lisible en clair (*non chiffré*) dans les requêtes faites au module.

Absorption des broks (éléments de supervision)

Le fonctionnement du thread de récupération des **broks** de ce module peut être configuré via certains paramètres, afin de modifier son "agressivité".

Pendant la mise à jour des données de supervision, le module ne peut pas répondre aux requêtes via son API.

Une mauvaise configuration de ces paramètres peut compromettre le bon fonctionnement du module, se rapprocher du support si vous avez le moindre doute.

Principe de l'algorithme d'absorption des broks :

1. Attente de broks à traiter
2. Récupération de broks en retard (*fonctionnalité de rattrapage*)
3. Désérialisation des broks
4. Entrée en session critique (*les requêtes à l'API sont bloquées*)
5. Traitement des broks
6. Libérer la session critique et attendre de nouveaux broks, **ou** continuer l'absorption de broks (*en cas de retard important, on repart à l'étape 1. en restant sur la session critique*)

Nom	Type	Unité	Défaut	Commentaire
`broker_module_livedata__broks_getter__activate_late_set_catchup`	Booléen	---	**1**	Utilisation de la fonctionnalité de rattrapage pour absorber des **broks** en retard : • **1** : Activé • **0** : Désactivé
`broker_module_livedata__broks_getter__nb_late_set_allowed_before_catchup`	Nombre	Nombre de broks set	**10**	Nombre de **brok set** en attente toléré. Au-dessus de ce nombre, les **brok set** sont immédiatement récupérés par l'algorithme de rattrapage pour être traités immédiatement.
`broker_module_livedata__broks_getter__catchup_broks_managed_by_module_in_a_catchup_loop`	Nombre	Nombre de broks	**200000**	Nombre maximal de **broks** que l'algorithme de rattrapage récupère avant de lancer le traitement. Ce paramètre permet de borner la consommation mémoire et le temps d'exécution d'un tour de boucle de traitement.
`broker_module_livedata__broks_getter__catchup_run_endless_until_nb_late_set_allowed_reached`	Booléen	---	**1**	Après traitement des **broks**, si le nombre de **brok set** en retard est trop élevé, • **1** : continuer le rattrapage et absorber des **broks** en retard en restant sur la session critique (*"avec le lock"*) • **0** : arrêter l'absorption de **brok** et libérer la session critique (*rendre le lock*)
`broker_module_livedata__broks_getter__include_deserialization_and_catchup_in_lock`	Booléen	---	**0**	Dans le cas où vous voulez disposer d'un maximum de temps CPU pour traiter les **broks** en retard, vous pouvez activer ce paramètre afin de bloquer les requêtes à l'API dès la phase 2 (*Récupération de broks en retard*) puis une fois les broks rattrapés passés en Phase 5 (*Traitement des broks*). Deux valeurs possibles pour ce paramètre : • **1** : Activé • **0** : Désactivé