Actions de masse (Shinken admin)

Sommaire

Concept

Lorsqu'un utilisateur fait une demande de rapport SLA dans l'Interface de Visualisation, la génération peut avoir un impact non négligeable sur le fonctionnement de l'Interface.

? Unknown Attachment

- Une demande de rapport (*suivant les critères choisis*) peut nécessiter la récupération d'une grosse quantité de données :
  - Le traitement de ces données peut avoir coût CPU non négligeable qui a pour effet de ralentir le fonctionnement de l'Interface,
  - ainsi que le surcoût de consommation mémoire nécessaire au stockage de ces informations brut pour générer et le rapport en lui-même.
- Cela peut avoir comme effet :
  - de bloquer ou ralentir l'interface, mettant du temps à répondre aux autres utilisateurs,
  - et de pénaliser les autres programmes qui s'exécutent sur le même système, s'ils ont besoin de RAM au moment de la génération du rapport.

Le fait de déporter la génération des rapports dans le module `broker--module-report-builder` permet :

- de déporter le travail sur d'autre CPUs que celui de la WebUI (*Broker 1*),
- mais aussi de donner la possibilité de générer les rapports sur une autre machine, avec un Broker portant un ou X `broker--module-report-builder`.
  - dans ce cas, les resources CPU et RAM nécéssaire à la génération des rapports ne surcharge plus le serveur hébergeant la WebUI (*Broker 2*).

(*voir la page* Partage de charge lors de la génération des rapports SLA depuis le module WebUI)

Activation du module

Le module `broker--module-report-builder` peut uniquement être activé sur un démon de type Broker.

- L'activation du module s'effectue en ajoutant le nom de ce module dans le fichier de configuration du Broker.
  - Ouvrir le fichier de configuration du Broker à l'emplacement **/etc/shinken/brokers/nom_du_broker.cfg**,
  - trouver la ligne du paramètre "modules"
  - et ajouter le nom du module "broker--module-report-builder".

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

Redémarrer l'Arbiter pour prendre en compte le changement de configuration

```
service shinken-arbiter restart
```

Configuration

Exemple de fichier de configuration

```
# CFG_FORMAT_VERSION 1 ( SHINKEN : DON'T TOUCH THIS LINE )

#================================================================================
# broker--module-report-builder
#================================================================================
# Daemon that can load this module:
# - broker
# This module is used to generate reports on Broker
#================================================================================

define module {

    #  #
    #      MODULE IDENTITY      #
    #  #

    #  Module name [ Must be unique ]                                              [ MANDATORY ]
    #
    module_name                                     broker--module-report-builder-example

    #  Module type [ Do not edit ]                                                 [ MANDATORY ]
    #
    module_type                                     broker__module_report_builder

    #  #
    #      LISTENING PARAMETERS      #
    #  #

    #  IP address to listen to
    #
    #           Default : 0.0.0.0 ( all interfaces )
    #
    # broker__module_report_builder__listening_address    0.0.0.0

    #  Port to listen to
    #
    #           Default : 24100
    #
    # broker__module_report_builder__listening_port       24100

    #  #
    #      HTTPS PARAMETERS      #
    #  #

    #  Enable this parameter if you want to receive requests in HTTPs mode
    #
    #           Default : 0 => Disable
    #           ...     : 1 => Enable
    #
    # broker__module_report_builder__use_ssl              0

    #  Certificate file
    #
    #           Default : /etc/shinken/certs/server.cert
    #
    # broker__module_report_builder__ssl_cert             /etc/shinken/certs/server.cert

    #  Key file
    #
    #           Default : /etc/shinken/certs/server.key
    #
    # broker__module_report_builder__ssl_key              /etc/shinken/certs/server.key

    #  #
    #      AUTHENTICATION      #
    #  #

    #  Token used to authenticate on this module
    #
    #           Default : change_me
    #
```

```
    # broker__module_report_builder__token                change_me

    #  #
    #      MODULES      #
    #  #

    #  This parameter defines the name of SLA module to use for report generation
    #
    #           ...     : sla => [mandatory] read sla from this module definition
    #
    modules                                              sla

    #  #
    #      STATISTICS      #
    #  #

    #  Ranges definition for the performance check
    #
    #           Default : 5,10,20,30,60,120,300,600 ( seconds )
    #
    # broker__module_report_builder__exec_stats_ranges    5,10,20,30,60,120,300,600

    #  #
    #      BROKS GETTER PARAMETERS      #
    #  #

    #  These parameters allow some internal tuning in broks management in this module

    #  Late broks sets catchup
    #
    #           ...     : 0 => Disable
    #           Default : 1 => Enable
    #
    # broker__module_report_builder__broks_getter__activate_late_set_catchup 1

    #  Take extra broks sets to manage if more than this parameter sets are waiting
    #
    #           Default : 10
    #
    # broker__module_report_builder__broks_getter__nb_late_set_allowed_before_catchup 10

    #  Stop taking extra broks sets in catchup when we reach this number of broks
    #
    #           Default : 200000
    #
    # broker__module_report_builder__broks_getter__catchup_broks_managed_by_module_in_a_catchup_loop 200000

    #  Continue catchup if too many late broks sets remains after
    #
    #           ...     : 0 => Disable
    #           Default : 1 => Enable
    #
    # broker__module_report_builder__broks_getter__catchup_run_endless_until_nb_late_set_allowed_reached 1

    #  Take the lock as soon as getter thread has some broks to manage
    #
    #           Default : 0 => Disable
    #           ...     : 1 => Enable
    #
    # broker__module_report_builder__broks_getter__include_deserialisation_and_catchup_in_lock 0

}
```

Détails des sections composant le fichier de configuration

Identification du module

```
#  #
#      MODULE IDENTITY      #
#  #

#  Module name [ Must be unique ]                                                    [ MANDATORY ]
#
module_name                                        broker--module-report-builder

#  Module type [ Do not edit ]                                                       [ MANDATORY ]
#
module_type                                        broker__module_report_builder
```

Il est possible de définir plusieurs instances de module de type "broker--module-report-builder" dans une architecture Shinken.

- Chaque instance devra avoir un nom unique.

Nom	Type	Unité	Défaut	Commentaire
`module_name`	Texte	---	**broker--module-report-builder**	**Il est conseillé de choisir un nom en fonction de l'utilisation qui va être faite du module pour que la configuration soit plus simple à maintenir.** Doit être unique.
`module_type`	Texte	---	**broker__module_report_builder**	Ne peut être modifié.

Paramètres réseau

```
#  #
#      LISTENING PARAMETERS      #
#  #

#  IP address to listen to
#
#           Default : 0.0.0.0 ( all interfaces )
#
# broker__module_report_builder__listening_address    0.0.0.0

#  Port to listen to
#
#           Default : 24100
#
# broker__module_report_builder__listening_port       24100
```

Nom	Type	Unité	Défaut	Commentaire
`broker__module_report_builder__listening_address`	Texte	---	**0.0.0.0**	Ce paramètre précise sur quelle interface réseau le module va se mettre en écoute pour recevoir des requêtes à traiter. Les valeurs possibles sont : ▪ **0.0.0.0** pour écouter sur toutes les interfaces réseau disponibles sur le système où s'exécute le Broker, ▪ **127.0.0.1** pour ne répondre qu'aux requêtes locales (*le module n'est pas accessible sur le réseau et ne répond qu'aux requêtes issues de la même machine*), ▪ l'adresse IP d'une des interfaces réseau du système où s'exécute le Broker.
`broker__module_report_builder__listening_port`	Entier	---	**24100**	Port d'écoute utilisé par le module pour attendre des requêtes.

Autorisation dans le pare feu

Si le module doit répondre à des requêtes provenant du réseau, et que le système sur lequel s'exécute le Broker dispose d'un pare feu, il faudra autoriser le trafic entrant sur le port d'écoute configuré.

Par exemple, si **firewalld** est actif sur le serveur sur lequel s'exécute le Broker les commandes suivantes peuvent être utilisée pour autoriser le trafic entrant sur le port 24100 :

```
firewall-cmd --add-port=24100/tcp
firewall-cmd --runtime-to-permanent
```

Chiffrement des échanges réseau

```
#  #
#      HTTPS PARAMETERS      #
#  #

#  Enable this parameter if you want to receive requests in HTTPs mode
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_report_builder__use_ssl             0

#  Certificate file
#
#           Default : /etc/shinken/certs/server.cert
#
# broker__module_report_builder__ssl_cert            /etc/shinken/certs/server.cert

#  Key file
#
#           Default : /etc/shinken/certs/server.key
#
# broker__module_report_builder__ssl_key             /etc/shinken/certs/server.key
```

Nom	Type	Unité	Défaut	Commentaire
`broker__module_report_builder__use_ssl`	Booléen	---	**0**	Chiffrer les échanges en utilisant le protocole **https** au lieu de **http** (*1 pour activer, 0 pour désactiver*)
`broker__module_report_builder__ssl_cert`	Texte	---	**/etc/shinken/certs/server.cert**	Chemin du fichier contenant le certificat.
`broker__module_report_builder__ssl_key`	Texte	---	**/etc/shinken/certs/server.key**	Chemin du fichier contenant la clé du certificat.

Paramètre d'identification

Afin de sécuriser l'utilisation du module, un jeton d'identification est nécessaire et il doit être fourni avec chaque requête.

Ce jeton est défini via le paramètre suivant :

```
#  #
#      AUTHENTICATION      #
#  #

#  Token used to authenticate on this module
#
#           Default : change_me
#
# broker__module_report_builder__token                change_me
```

Nom	Type	Unité	Défaut	Commentaire
`broker__module_report_builder__token`	Texte	---	**change_me**	Chaîne de texte utilisée pour chaque requête au module

Modules

Cette section permet de définir quel module SLA utiliser pour la génération des rapports.

```
#  #
#      MODULES      #
#  #

#  This parameter defines the name of SLA module to use for report generation
#
#           ...     : sla => [mandatory] read sla from this module definition
#
modules                                             sla
```

Nom	Type	Unité	Défaut	Commentaire
`modules`	Texte	---	**sla**	Nom du module SLA à utiliser pour la génération des rapports. • ne peut être vide, • ne doit contenir qu'un seul nom de module SLA.

Statistiques (pour la supervision)

```
#  #
#      STATISTICS      #
#  #

#  Ranges definition for the performance check
#
#           Default : 5,10,20,30,60,120,300,600 ( seconds )
#
# broker__module_report_builder__exec_stats_ranges    5,10,20,30,60,120,300,600
```

Nom	Type	Unité	Défaut	Commentaire
`broker__module_report_builder__exec_stats_ranges`	Liste de nombres	Seconde	**5,10,20,30,60,120,300,600**	Regroupe le nombre de rapports effectués sur les dernières 24h selon leur durée d'exécution, à des fins statistiques pour la supervision.

Absorption des broks (*information de supervision venant des Schedulers, options internes*)

```
#  #
#      BROKS GETTER PARAMETERS      #
#  #

#  These parameters allow some internal tuning in broks management in this module

#  Late broks sets catchup
#
#           ...     : 0 => Disable
#           Default : 1 => Enable
#
# broker__module_report_builder__broks_getter__activate_late_set_catchup 1

#  Take extra broks sets to manage if more than this parameter sets are waiting
#
#           Default : 10
#
# broker__module_report_builder__broks_getter__nb_late_set_allowed_before_catchup 10

#  Stop taking extra broks sets in catchup when we reach this number of broks
#
#           Default : 200000
#
# broker__module_report_builder__broks_getter__catchup_broks_managed_by_module_in_a_catchup_loop 200000

#  Continue catchup if too many late broks sets remains after
#
#           ...     : 0 => Disable
#           Default : 1 => Enable
#
# broker__module_report_builder__broks_getter__catchup_run_endless_until_nb_late_set_allowed_reached 1

#  Take the lock as soon as getter thread has some broks to manage
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_report_builder__broks_getter__include_deserialisation_and_catchup_in_lock 0
```

Ces paramètres sont dédiés au fonctionnement interne au module, il est fortement recommandé de ne pas les modifier sans le support dédié.

Le fonctionnement du thread de récupération des **broks** 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 HTTP qu'il reçoit.

Principe de l'algorithme d'absorption des broks :

1. Attente de broks à traiter
2. Récupération de broks en retard (*fonctionnalité de rattrapage*)
3. Dé-sérialisation 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_report_builder__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_report_builder__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_report_builder__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_report_builder__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_report_builder__broks_getter__include_deserialisation_and_catchup_in_lock`	Booléen	---	**0**	Dans le cas où on veut disposer d'un maximum de temps CPU pour traiter les **broks** en retard, on peut 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