# Module event-manager-writer

**Sommaire**



## Concept

Le module de type `event_container` permet de disposer de la fonctionnalité du bac à événement ( *voir la page* Vue - Les Événements ).

Ce module peut s'accrocher :

- sur un Broker pour gérer l'écriture des événements en base de données ( *nous livrons par défaut un module nommé* `event-manager-writer` ) ;
- sur un module de type `webui`, pour gérer l'affichage des événements ( *nous livrons par défaut un module nommé* `event-manager-reader` ) ;

**Nous détaillerons ici la configuration du module quand il écrit les données.**

- La configuration du module quand il lit les données est détaillée sur la page Module event-manager-reader.

Pour que la fonctionnalité de bac à événement puisse afficher des événements à jour, il faut absolument que ce module soit activé sur un Broker pour écrire les évènements en base.

Il ne peut y avoir qu'un `event-manager-writer` par base MongoDB.

- Donc, pour avoir plusieurs modules qui écrivent sur le même serveur MongoDB, il faut les configurer pour écrire dans des bases différentes ( *voir le paramètre* ***database*** *ci-dessous* ).
- Cependant, il est fortement déconseillé de configurer plusieurs modules sur un même serveur MongoDB, en raison des ressources qu'un module peut requérir ( *bande passante disque et occupation disque* )

Il faut porter une attention particulière sur le volume de la base d'événements, dû au fonctionnement du module et des informations qu'il stocke.

FONCTIONNEMENT :

- Un élément ( *hôte, cluster, checks* ) peut avoir :
  - peu de changements d'état, consommant ainsi une place raisonnable,
  - avoir une oscillation de son état ( *alternance de pannes et de retour OK* ), et ainsi enregistrer beaucoup de changements d'état.
- Le souci est que la place prise par un changement d'état est **variable**, car chaque événement stocké contient le résultat court et le résultat long.
  - **Surveiller la taille de la base d'événement** avec le modèle d'hôte : Modèle shinken-broker-module-event-manager-writer ( *voir la page* Modèle shinken-broker-module-event-manager-writer ).
- Pour un élément donné, le changement de statut ( *OK, Attention, Critique, Inconnu* ) ou un changement de contexte ( *Flapping, Downtime, Acknowledged* ) créera une nouvelle entrée dans la liste des événements.

**A NOTER**

Un cas modifie une entrée déjà enregistrée : Quand le statut d'un élément passe de l'**état non confirmé** à l'**état confirmé**, sans autre changement de son statut ou de son contexte, l'événement le décrivant est mis à jour.

- La colonne **Confirmation de Statut** est modifiée pour indiquer que ce statut a été confirmé.
- la colonne **Date de confirmation de Statut** est modifiée pour indiquer à partir de quand le statut a été confirmé, la colonne **Date du changement** indiquant toujours quand le statut ( *en état non confirmé* ) a démarré.

## Activation du module

Les modules de type "`event_container`" vont gérer l'écriture des événements en base de données quand ils sont activés sur un démon de type "b`roker`", qu'on appellera le **démon parent**.

- L'activation du module s'effectue en ajoutant le **nom** du module dans la configuration du **démon parent**.
  - Pour cela, il faut ouvrir le fichier de configuration du **démon parent** ( *de type "`broker`"* ), et ajouter dans le paramètre **modules**, le nom du module de type "`event_container`".
- Il est possible de créer plusieurs modules de type "`event_container`" pour écrire les données en base de données.
  - Cela permet, **par exemple**, d'avoir des configurations différentes en fonction des royaumes.
- Tous les éléments supervisés doivent avoir un module de type "`event_container`" attaché à un Broker du royaume, ou d'un royaume parent, pour que les événements qu'ils génèrent soient enregistrés en base de données.

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

```
service-shinken-arbiter restart
```

### Exemple d'activation du module nommé "event-manager-writer" sur le démon "broker-master" ( configuration livrée par défaut par Shinken )

L'exemple suivant

- active le module "`event-manager-writer`" ,
- sur le démon "`broker-master`", dont la configuration est dans le fichier **/etc/shinken/brokers/broker-master.cfg**.

Modification dans le fichier du module **/etc/shinken/brokers/broker-master.cfg** :

```
define broker {
    [...]
    modules                  Module 1, Module 2, Module 3, event-manager-writer
    [...]
}
```

Puis redémarrage de l'Arbiter

```
service-shinken-arbiter restart
```

### Créer un nouveau module de type event_container pour l'enregistrement des événements

Pour pouvoir configurer un module de type "`event_container`" pour l'enregistrement des événements, il faut créer un nouveau fichier de configuration grâce au fichier d'exemple fourni par défaut.

- Pour commencer, il faut choisir le nom du nouveau module.
  - Pour l'exemple, on l'appelle "`Mon-Module-event-manager-writer`".
  - Remplacer dans l'exemple le mot "`Mon-Module-event-manager-writer`" par le nom qui a été choisi.
- Puis il faut créer le fichier de configuration :
  - Copier le fichier de définition du module d'exemple : **/etc/shinken-user-example/configuration/daemons/brokers/modules /event_manager_writer/event_manager_writer-example.cfg** dans le répertoire de définition des modules **/etc/shinken/modules/** .
    ( *Exemple : /etc/shinken/modules/broker__Mon-Module-`event-manager-writer`.cfg* )

```
cp /etc/shinken-user-example/configuration/daemons/brokers/modules/event_manager_writer
/event_manager_writer-example.cfg /etc/shinken/modules/broker__Mon-Module-event-manager-writer.
cfg
```

- Ensuite, il faut modifier le fichier nouvellement créé pour configurer le nouveau module.
  - Il faut vérifier que le fichier appartienne à l'utilisateur shinken et qu'il possède le droit d'édition. Si ce n'est pas le cas, il faut effectuer les commandes suivantes :

```
chown shinken:shinken /etc/shinken/modules/broker__Mon-Module-event-manager-writer.cfg
chmod u+w /etc/shinken/modules/broker__Mon-Module-event-manager-writer.cfg
```

  - On change le nom du module en "`Mon-Module-event-manager-writer`" dans le fichier **/etc/shinken/modules/broker__Mon-Module-event-manager-writer.cfg**

```
...
        #  Module name [ Must be unique ]
[ MANDATORY ]
    #


    module_name                                          Mon-Module-event-manager-writer
...
```

- Ensuite, il faut ajouter le nouveau module dans la liste des module du Broker correspondant.
  - Dans notre exemple, on ajoute le module "`Mon-Module-event-manager-writer`" au démon "`broker-master`" défini dans le fichier **/etc/shinken/brokers/broker-master.cfg**

```
define broker {
        [...]
        modules                                              Module 1, Module 2, Module 3,
Mon-Module-event-manager-writer
    [...]
}
```

- Puis pour finir, il faut redémarrer l'Arbiter pour que le Broker puisse prendre en compte ce nouveau module.

```
service-shinken-arbiter restart
```

## Configuration

La configuration du module que Shinken livre par défaut se trouve dans le fichier **/etc/shinken/modules/event_manager_writer.cfg**.

- un exemple dans **/etc/shinken-user-example/configuration/daemons/brokers/modules/event_manager_writer/event_manager_writer-example.cfg**.

### Exemple de fichier de configuration

```
#===============================================================================
# event manager
#===============================================================================
# Daemons that can load this module:
#  - broker (to save events information into a mongodb database)
# This module compute and save event for event manager
#===============================================================================

define module {

    #  #
    #      MODULE IDENTITY      #
    #  #
```

```
#  Module name [ Must be unique ]                                                    [ MANDATORY ]
#
module_name                                        event-manager-writer

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

#  #
#      MODULE OPTIONS      #
#  #

#  Number of day the events are kept in database
#
#           Default : 30 ( days )
#
# day_keep_data                                      30

#  #
#      DATABASE CONNECTION      #
#  #

#   MongoDB parameters   #

#  MongoDB uri definition . You can find the mongodb uri syntax at
#  https://docs.mongodb.com/manual/reference/connection-string/
#
#           Default : mongodb://localhost/?w=1&fsync=false
#
# uri                                                mongodb://localhost/?w=1&fsync=false

#  Which database contains events data
#
#           Default : event_container
#
# database                                           event_container

#  username/password to authenticate to MongoDB.
#  Both parameters must be provided for authentication to function correctly.
#
# broker__module_event_manager_writer__database__username

#
# broker__module_event_manager_writer__database__password

#  SSH tunnel activation to secure your mongodb connection
#  That will allow all mongodb to be encrypted & authenticated with SSH
#
#           ...     : Enable  => 1 ( enable ssh tunnel )
#           Default : Disable => 0 ( disable ssh tunnel )
#
# use_ssh_tunnel                                     0

#  If the SSH connection goes wrong, then retry use_ssh_retry_failure time before_shinken_inactive
#
#           Default : 1 ( try )
#
# use_ssh_retry_failure                              1

#  SSH user to connect to the mongodb server.
#
#           Default : shinken
#
# ssh_user                                           shinken

#  SSH keyfile to connect to the mongodb server.
#
#           Default : ~shinken/.ssh/id_rsa
#
# ssh_keyfile                                        ~shinken/.ssh/id_rsa
```

```
    #  SSH Timeout used to test if the SSH tunnel is viable or not, in seconds.
    #
    #           Default : 10 ( seconds )
    #
    # ssh_tunnel_timeout                                10

    #   AutoReconnect Management   #

    #  When MongoDB require you to reconnect ( For example, It can occur when a new PRIMARY is elected
    #  in a MongoDB cluster ), it will raised the MongoDB AutoReconnect exception.

    #  How many try to reconnect before module go in error
    #
    #           Default : 4 ( try )
    #
    # auto_reconnect_max_try                            4

    #  Time between each try
    #
    #           Default : 3 ( seconds )
    #
    # auto_reconnect_sleep_between_try                  3

    #  NOTE: Change these values only if you have a MongoDB cluster and you change the
    #        heartbeatTimeoutSecs of your MongoDB replica set
    #        The value of auto_reconnect_max_try * auto_reconnect_sleep_between_try must be higher than
    #        heartbeatTimeoutSecs in the rs.conf(); of your MongoDB replica set.

    #  #
    #      WORKERS IN THE BROKER      #
    #  #

    #  This module will use workers in the broker, each worker will manage a shard of all hosts/checks.
    #  This parameter is used by the broker to set the number of workers.
    #  Each worker will use one CPU, which will balance the event processing load among CPUs.
    #
    #           Default : 1 ( worker )
    #
    # broker_module_nb_workers                          1

    #  #
    #      INTERNAL OPTIONS      #
    #  #

    #  INTERNAL : DO NOT EDIT FOLLOWING PARAMETER WITHOUT YOUR DEDICATED SUPPORT

    #  Broker idle time before considering that Shinken is inactive.
    #  Use this if you have Broker loop time that exceeds 30 seconds
    #
    #           Default : 30 ( seconds )
    #
    # time_before_shinken_inactive                      30

}
```

## Détails des sections composant le fichier de configuration

### Identification du module

Il est possible de définir plusieurs instances de module de type "event-manager-writer" dans l'architecture Shinken.

- Chaque instance devra avoir un nom unique.

| Nom | Type | Unité | Défaut | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| `module_name` | Texte | --- | **event-manager-writer** | **Il est conseillé de choisir un nom en fonction de l'utilisation du module pour que la configuration soit simple à maintenir.**<br><br>Doit être unique. |
| `module_type` | Texte | --- | **event_container** | Ne dois pas être modifié. |

### Taille du bac d'événement en nombre de jours

```
#  #
#      MODULE OPTIONS      #
#  #

#  Number of day the events are kept in database
#
#          Default : 30 ( days )
#
# day_keep_data                                    30
```

Le paramètre "`day_keep_data`" permet de choisir **le nombre de jours qu'un événement sera gardé dans la base**.

- Si la base MongoDB prend trop de place sur le disque, cela peut être monitorer avec le modèle d'hôte : Modèle shinken-broker-module-event-manager-writer ( *voir la page* Modèle shinken-broker-module-event-manager-writer ).
- Il est possible de diminuer le nombre de jours sauvegardés.

| Nom | Type | Unité | Défaut | Description |
|---|---|---|---|---|
| `day_keep_data` | Entier | jour | **30** | Durée en nombre de jours d'un événement dans le bac à événement. |

### Accès à la base MongoDB

Cette configuration s'effectue dans le fichier de configuration du module.

Pour se connecter à la base MongoDB utilisé pour le stockage des données, 2 méthodes sont disponibles :

- **Connexion directe :** Par défaut, mais non sécurisée.
- **Tunnel SSH :** Shinken se connecte à la base MongoDB au travers d'un module SSH pour plus de sécurité

#### Configuration de l'URI de connexion et de l'authentification par mot de passe

```
#  #
#      DATABASE CONNECTION      #
#  #

#   MongoDB parameters   #

#  MongoDB uri definition . You can find the mongodb uri syntax at
#  https://docs.mongodb.com/manual/reference/connection-string/
#
#           Default : mongodb://localhost/?w=1&fsync=false
#
# uri                                                 mongodb://localhost/?w=1&fsync=false

#  Which database contains events data
#
#           Default : event_container
#
# database                                            event_container

#  username/password to authenticate to MongoDB.
#  Both parameters must be provided for authentication to function correctly.
#
# broker__module_event_manager_writer__database__username

#
# broker__module_event_manager_writer__database__password
```

| Nom | Type | Unité | Défaut | Description |
|---|---|---|---|---|
| `uri` | Texte | URL | **mongodb://localhost/?safe=true** | La description de la syntaxe de l'URI de MongoDB est disponible à l'adresse suivante https://docs.mongodb.com/manual/reference/connection-string/ |
| `database` | Texte | --- | **shinken** | Nom de la base de données où sont stockées les données événements. |
| `broker__module_event_manager_writer__database__username` | Texte | --- | | Utilisateur pour l'authentification avec mot de passe à la base MongoDB.<br><br>Utile uniquement si l'activation par mot de passe a été activé ( *voir la page MongoDB - activation de l'authentification par mot de passe* ) |
| `broker__module_event_manager_writer__database__password` | Texte | --- | | Mot de passe de l'utilisateur utilisé pour l'authentification avec mot de passe à la base MongoDB.<br><br>Utile uniquement si l'activation par mot de passe a été activé ( *voir la page MongoDB - activation de l'authentification par mot de passe* ) |

### Connexion directe au serveur MongoDB

Par défaut, le module se connecte de manière directe à la base MongoDB pour y lire et écrire les données.
Dans la configuration du module, ceci correspond au paramètre "`use_ssh_tunnel`" à 0.
C'est la méthode de connexion par défaut lorsque la base est sur la même machine que le démon ( *quand l'URL de la base est localhost* ).

Si la base est sur une autre machine, il faudra alors se connecter à la base via un tunnel SSH. Cela permet à la base distance de rester en écoute réseau sur l'interface réseau local, ce qui la sécurise des accès extérieurs ( *voir la page* Sécurisation des connexions aux bases MongoDB ).

### Connexion par SSH au serveur MongoDB

```
#  SSH tunnel activation to secure your mongodb connection
#  That will allow all mongodb to be encrypted & authenticated with SSH
#
#          ...     : Enable  => 1 ( enable ssh tunnel )
#          Default : Disable => 0 ( disable ssh tunnel )
#
# use_ssh_tunnel                                    0

#  If the SSH connection goes wrong, then retry use_ssh_retry_failure time before_shinken_inactive
#
#          Default : 1 ( try )
#
# use_ssh_retry_failure                             1

#  SSH user to connect to the mongodb server.
#
#          Default : shinken
#
# ssh_user                                          shinken

#  SSH keyfile to connect to the mongodb server.
#
#          Default : ~shinken/.ssh/id_rsa
#
# ssh_keyfile                                       ~shinken/.ssh/id_rsa

#  SSH Timeout used to test if the SSH tunnel is viable or not, in seconds.
#
#          Default : 10 ( seconds )
#
# ssh_tunnel_timeout                                10
```

| Nom | Type | Unité | Défaut | Description |
|---|---|---|---|---|
| `use_ssh_tunnel` | Booléen | --- | **0** | • 1 : Connexion par tunnel SSH.<br>• 0 : Connexion directe. |
| `use_ssh_retry_failure` | Entier | Nombre d'essais | **1** | Spécifie le nombre supplémentaire de tentatives lors de l'établissement du tunnel SSH si ce dernier n'arrive pas à être établi. |
| `ssh_user` | Texte | Utilisateur unix | **shinken** | L'utilisateur avec lequel le tunnel sera établi. |
| `ssh_keyfile` | Texte | Chemin de fichier | **~shinken/.ssh/id_rsa** | La clé SSH privée présente sur le serveur Shinken qui sera utilisé pour établir le tunnel. |
| `ssh_tunnel_timeout` | Entier | Seconde | **10** | Spécifie le timeout en secondes de la vérification du tunnel SSH avant que la connexion vers MongoDB soit effectuée. |

## Gestion de l'auto-reconnexion

```
#   AutoReconnect Management

#  When MongoDB require you to reconnect ( For example, It can occur when a new PRIMARY is elected
#  in a MongoDB cluster ), it will raised the MongoDB AutoReconnect exception.

#  How many try to reconnect before module go in error
#
#          Default : 4 ( try )
#
# auto_reconnect_max_try                          4

#  Time between each try
#
#          Default : 3 ( seconds )
#
# auto_reconnect_sleep_between_try                3

#  NOTE: Change these values only if you have a MongoDB cluster and you change the
#        heartbeatTimeoutSecs of your MongoDB replica set
#        The value of auto_reconnect_max_try * auto_reconnect_sleep_between_try must be higher than
#        heartbeatTimeoutSecs in the rs.conf(); of your MongoDB replica set.
```

**Définitions**

- **Primaire:** nom de MongoDB pour désigner un serveur maître, le serveur sur lequel il est possible de faire des requêtes d'écriture dans la base.
- **Election** : processus de MongoDB pour choisir un nouveau membre Primaire si le membre Primaire devient inaccessible

( *voir la page* Haute disponibilité de la base MongoDB (mise en place d'un cluster) )

Dans le cas de l'utilisation d'un cluster MongoDB, lorsque le membre Primaire devient inaccessible, une nouvelle élection est déclenchée, ce qui provoque une coupure temporaire de l'accès à la base.

Dans le but de ne pas interrompre le service, le module "event-manager-reader" va se reconnecter automatiquement au cluster MongoDB.
Pour ce faire, il va faire un nombre d'essais égaux au paramètre "`auto_reconnect_max_try` " avec une pause de X secondes entre chaque essai ( *correspondant au paramètre* `"auto_reconnect_sleep_between_try"` ).

Par défaut pour MongoDB le temps maximum avant qu'un membre Primaire soit considéré comme indisponible et qu'une nouvelle élection ait lieu est de 10 secondes.
Voir : "**heartbeatTimeoutSecs**" donné par la commande rs.conf(); dans un shell de MongoDB.

| Nom | Type | Unité | Défaut | Description |
|---|---|---|---|---|
| `auto_reconnect_max_try` | Entier | Nombre d'essais | **4** | Nombre d'essais de reconnexion à la base. |
| `auto_reconnect_sleep_between_try` | Entier | Seconde | **3** | Temps entre chaque essai. |

Les valeurs par défauts du fichier laisse 12 secondes, ce qui est amplement suffisant avec la configuration par défaut de MongoDB.

Il est conseillé de ne pas modifier ces valeurs.

## Utilisation des workers du module event-manager-writer

```
#  #
#      WORKERS IN THE BROKER      #
#  #

#  This module will use workers in the broker, each worker will manage a shard of all hosts/checks.
#  This parameter is used by the broker to set the number of workers.
#  Each worker will use one CPU, which will balance the event processing load among CPUs.
#
#          Default : 1 ( worker )
#
# broker_module_nb_workers                          1
```

Le paramètre "`broker_module_nb_workers`" va déterminer combien de fois le module va se cloner pour gérer le flux de donnée à enregistrer afin de repartir cette charge sur plusieurs CPU.

Il est possible de changer ce paramètre si l'utilisation CPU du processus : "*NOM DU BROKER* [ - Module: event-manager-writer ][ Worker: 0 ]" est trop élevé.

| Nom | Type | Unité | Défaut | Description |
|---|---|---|---|---|
| `broker_module_nb_workers` | Entier | worker | **1** | Nombre de workers qui traitent le flux de donnée pour sauvegarder les événements dans la base MongoDB ( *le traitement est réparti sur les workers* ). |

Ne pas dépasser le nombre de core cpu de la machine : cela serait contre-productif pour les performances.

## Options internes

```
#  #
#      INTERNAL OPTIONS      #
#  #

#  INTERNAL : DO NOT EDIT FOLLOWING PARAMETER WITHOUT YOUR DEDICATED SUPPORT

#  Broker idle time before considering that Shinken is inactive.
#  Use this if you have Broker loop time that exceeds 30 seconds
#
#          Default : 30 ( seconds )
#
# time_before_shinken_inactive                      30
```

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

| Nom | Type | Unité | Défaut | Description |
|---|---|---|---|---|
| `time_before_shinken_inactive` | Entier | Seconde | **30** | Temps d'inactivité du Broker avant de considérer que Shinken est inactif.<br><br>Utilisez cette option si le temps de boucle du Broker dépasse 30 secondes. |