

# Le Receiver

## Rôle

Le Receiver reçoit les données de checks passifs, et agit comme une commande tampon distribuée, qui sera lue par l'Arbiter. Il peut y avoir plusieurs Receiver pour du load-balancing et des rôles de spare en standby à chaud.

Il peut également utiliser des modules pour accepter des données provenant de différents protocoles.

- Module pour la collecte de données passives: module WS Arbiter

## Communications vers les autres démons

L'Arbiter récupère les données du Receiver chaque seconde.

## Résumé des connexions du Receiver

Source	Destination	Port	Protocole	Note
Arbiter	Receiver	7773	HTTP/HTTPS	

## Données

Le Receiver garde en mémoire tampon les données des commandes externes. Ces commandes externes ont des noms d'hôtes ou de checks.

## Gestion de l'inventaire par les modules de receiver

Le démon receiver peut recevoir un inventaire d'hôtes souhaité qui seront transmis à ses modules. Les paramètres pour activer ce mécanisme sont:

- elements\_sharding\_enabled
- elements\_sharding\_filter\_by\_template
- elements\_sharding\_add\_data\_of\_templates

Pour leur fonctionnement, voir la description des variables ci-dessous.

## Description des variables

Property	Default	Description
receiver_name	N/A	Cette variable est utilisée pour définir le nom raccourci du démon auquel les données sont associées.
address	N/A	Cette directive est utilisée pour définir l'adresse permettant à l'Arbiter de joindre ce Receiver. Par défaut "localhost", changez-le par un nom DNS ou une adresse IP.
port	7773	Cette directive est utilisée pour définir le port TCP utilisé par ce démon.
use_ssl	0	Cette variable est utilisée pour définir si le Receiver doit être contacté en HTTPS (*1*) ou HTTP (*0*). La valeur par défaut est *0* (HTTP).
spare	0	Cette variable est utilisée pour définir si le Receiver peut être géré comme un spare (prendra uniquement la configuration si le maître échoue). La valeur par défaut est *0* (maître).
realm	N/A	Cette variable est utilisée pour définir le royaume où le Receiver doit être. Si aucun n'est sélectionné, celui par défaut lui sera assigné.
direct_routing	0	Si activé, il enverra directement les commandes aux Schedulers s'il connaît le nom de l'hôte dans la commande.
timeout	3	Cette variable est utilisée pour définir le temps en secondes avant que l'Arbiter ne considère ce démon comme à l'arrêt. Si ce démon est joignable en HTTPS (use_ssl à 1) avec une latence élevée, nous vous conseillons alors d'augmenter cette valeur de timeout (l'Arbiter aura besoin de plus d'allers/retours pour le contacter).
data_timeout	120	Cette variable est utilisée pour définir le temps en secondes avant de considérer un transfert de configuration ou de données comme échoué.
max_check_att_empts	3	Si le ping permettant de détecter la disponibilité réseau du nœud est en échec N fois ou plus, alors le nœud est considéré comme mort. (par défaut, 3 tentatives)
check_interval	60	Intervalle de Ping toutes les N secondes.
modules	N/A	Cette variable est utilisée pour définir les modules chargés par le Receiver.

satellitemap	N/A	<p>Cette variable est utilisée dans le cas de royaume situé derrière un réseau NATé. Elle est de la forme d'une liste séparée par des "," de valeur <b>nom-démon=address:port</b></p> <p>Les démons ainsi listés seront contactés avec le couple address:port du paramètre au lieu de leur adresse dans leur .cfg.*</p> <p>Ceci permet ainsi à des démons derrière un réseau NAT d'échanger sur leur adresse locale au lieu de devoir ressortir sur leur adresse publique.</p> <p><i>Exemple: daemon1=192.168.0.1:7768,daemon2=192.168.0.1:7771</i></p>
elements_sharding_enabled	0	Mettre à 1 pour activer l'envoi de l'inventaires des hôtes vers le Receiver
elements_sharding_filter_by_template		Mettre le nom d'un modèle d'hôte qui va filtrer les hôtes à envoyer au Receiver
elements_sharding_add_data_of_templates		<ul style="list-style-type: none"> <li>mettre le nom d'un ou plusieurs modèles d'hôtes/cluster où seront prises les DATA à exporter dans l'inventaire des hôtes</li> <li>Cela permet de limiter le volume de donnée qui ira sur le Receiver ( qui peut être conséquent )</li> <li><b>Remarque:</b> les données de checks sont systématiquement présentes pour les checks.</li> </ul>
enabled	N/A	Cette variable est utilisée pour définir si le Receiver est activé ou non.

## Exemple de définition

⚠ Il est conseillé d'éditer les fichiers .cfg avec l'encodage utf-8

```

=====
# RECEIVER
=====
# The receiver manages passive information. It's just a "buffer" which will
# load passive modules and be read by the arbiter to dispatch data.
=====

define receiver {
# Shinken Enterprise. Lines added by import core. Do not remove it, it's used by Shinken Enterprise to
update your objects if you re-import them.
    _SE_UUID          core-receiver-1565c87e5adelle59d26080027f08538
    _SE_UUID_HASH     b0030c9912c63862c3957736875cae84
# End of Shinken Enterprise part

    #===== Daemon name and address =====
    # Daemon name. Must be unique
    receiver_name     receiver-master

    # IP/fqdn of this daemon (note: you MUST change it by the real ip/fqdn of this server)
    address           localhost

    # Port (HTTP/HTTPS) exposed by this daemon
    port              7773

    # 0 = use HTTP, 1 = use HTTPS
    use_ssl           0

    #===== Realm and architecture settings =====
    # Realm to set this daemon into
    spare            0

    # Realm to set this daemon into
    realm            All

    # 1 = will directly send commands to the schedulers if it know about the hostname in the command
    direct_routing   0

    #===== Daemon connection timeout and down state limit =====

```

```
# timeout: how many seconds to consider a node don't answer
timeout 3

# data_timeout: how many second to consider a configuration transfert to be failed
# because the network bandwidth is too small.
data_timeout 120

# max_check_attempts: how many fail check to consider this daemon as DEAD
max_check_attempts 3

# Check this daemon every X seconds
check_interval 60

#=====  
# Available:  
# - ws-arbiter: webservice interface  
modules

#=====  
# Some receiver modules can register to some host/checks data  
# and so if configured for it, receiver will have all the host/checks  
# useful DATA for the templates  
# By default this is disabled  
#elements_sharding_enabled 0

# Will only send host (and theirs checks) of this template  
#elements_sharding_filter_by_template filter-template

# Will only take DATA declared local to theses templates  
#elements_sharding_add_data_of_templates template-with-data1,template-wit-data2

# Host are shards across receivers. Set a higher weight if this receiver  
# need to manage more hosts than the others  
#elements_sharding_weight 1

#=====  
# 1 = is enabled, 0 = is disabled  
enabled 1
}
```