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 recevra des 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.

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 ce que l'Arbiter joigne 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 si il connait 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 considére 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 timout (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_attempts	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.
enabled	N/A	Cette variable est utilisée pour définir si le Receiver est activé ou non.

Exemple de définition

```
#================================================================================
# 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 {

    #======== 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                   node1.mydomain

    # 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 brandwith 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



    #======== Modules to enable for this daemon =========
    # Available:
    # - ws-arbiter: webservice interface
    modules



    #======== Enable or not this daemon =========
    # 1 = is enabled, o = is disabled
    enabled                  1
}
```