Le ws-arbiter (futur receiver-module-webservice)

Le ws-arbiter est le module qui permet de recevoir des résultats passifs pour des hôtes ou checks via HTTP(s).

Configuration du module Webservice

Ce module met à disposition une écoute HTTP qui écoute les requêtes HTTP POST et effectue des actions sur hôtes, clusters et checks.

- Le Web Service écoute sur port TCP 7760 (*par défaut*).
- Il supporte les accès authentifiés ou anonymes.

La configuration se défini dans le fichier de configuration du module (*présent par défaut dans* ***/etc/shinken/modules/ws-arbiter.cfg***).

```
#===============================================================================
# ws-arbiter   (webservice)
#===============================================================================
# Daemons that can load this module:
# - receiver
# - arbiter
# This module is a webservice that can be used to send checks to Shinken Enterprise
# as POST HTTP(s)
#===============================================================================


define module {

    #======== Module identity =========
    # Module name. Must be unique
    module_name               ws-arbiter
    # Module type (to load module code). Do not edit.
    module_type               ws_arbiter


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

    # port to listen
    port                      7760

    # HTTPs part, enable if you want to set the listening for HTTPS instead of default HTTP.
    # disabled by default. Set your own certificates.
    use_ssl                   0
    ssl_cert                  /etc/shinken/certs/server.cert
    ssl_key                   /etc/shinken/certs/server.key


    #======== HTTP authentication =========
    # You can use HTTP basic authentication method for this module.
    # If username is set to anonymous and password is commented, then
    # no authentication will be required.
    username                 anonymous
    #password                secret


}
```

Les valeurs peuvent être :

- **module_name**: définit un nom unique pour le module
- **module_type**: doit être ws_arbiter
- **host**: adresse de l'interface réseau sur laquelle effectuer l'écoute. 0.0.0.0 signifie "toutes les interfaces"
- **port**: port TCP à écouter
- **use_ssl** et **ssl_cert** / **ssl_key** : permet que le module écoute sur le port en SSL (*protocole HTTPS*) et d'utiliser des certificats
- **username** et **password**: si mis à "anonymous" aucune accréditation nécessaire. Si vous mettez un nom utilisateur/mot de passe, une authentification est nécessaire

Activation du module

Pour activer le module, ajouter simplement à votre Receiver le module **ws-arbiter** à la liste des modules dans le fichier de configuration du receiver.cfg

Sur une architecture distribuée importante, si vous avez plusieurs Receivers, vous pourrez activer un module **ws-arbiter** par Receiver.

Utilisation du module Webservice

Le webservice du Receiver écoute des requêtes HTTP pour ensuite effectuer des actions sur les hôtes/clusters ou checks concernés.

Un simple curl ou appel HTTP dans votre programme suffit pour envoyer des actions à Shinken.

/push_check_result: soumettre le résultat de check passif sur un hôte ou un check

Paramètres de l'appel

Nom	Description
`Méthode HTTP`	`POST`
`time_stamp`	(*optionnel*) date pour que Shinken puisse déterminer a quel moment a eu lieu la mesure. Par défaut, ce sera la date d'appel de l'API
`host_name`	Nom de l'hôte cible
`service_description`	Nom du check cible (*si résultat d'un check*)
`return_code`	Check: 0 => OK, 1 => WARNING, 2 = CRITICAL, 3 => UNKNOWN Hôte: 0 => OK, 1,2,3 => CRITIQUE
`output`	Résultat du check

Exemple:

```
curl -u user:password -X POST -d "time_stamp=$(date +%s)&host_name=host-checked&service_description=service-checked&return_code=0&output=Everything OK" http://shinken-srv:7760/push_check_result
```

Comme lors de l'écriture d'un check, le résultat du check peut être séparé du résultat long par un retour à la ligne.

Lors de l'envoi du résultat d'un check passif d'un hôte ou check, un retour à la ligne peut également être utilisé pour séparer résultat et résultat long.

L'exemple suivant permet d'envoyer un résultat et résultat long:

```
curl -u user:password -X POST -d $'host_name=host-checked&service_description=service-checked&return_code=0&output=short_update\nlong_output' http://shinken-srv:7760/push_check_result
```

On note dans l'exemple précédent que la chaine de caractères passées dans le curl pour les données POST est passée avec:

```
$'chaine'
```

et non

```
"chaine"
```

pour ordonner au check d'interpréter les caractères d'échappement ANSI et passer un véritable '\n' au lieu de '\' suivi de 'n'.

Cette astuce fonctionne si le shell utilisé est bash et peut ne pas être utilisable dans d'autres shells, ou si la requête est envoyée via un autre outil ou via un script dans un autre langage.

/acknowledge: Mettre un Prise en compte sur un hôte, un cluster ou un check

Paramètres de l'appel

Nom	Description
`Méthode HTTP`	`POST`
`action`	Options disponibles: add, delete. Ajoute un supprime la prise en compte. Par défaut: add
`time_stamp`	(*optionnel*) date pour que Shinken puisse déterminer a quel moment a eu lieu la mesure. Par défaut, ce sera la date d'appel de l'API
`host_name`	Nom de l'hôte cible
`service_description`	Nom du check cible (*si résultat d'un check*)
`comment`	le commentaire de la prise en compte (*optionnel*)
`notify`	1 ou 0 pour notifier les contacts qui sont signalé sur l'équipement comme recevant les notifications de changement sur cette élément.
`author`	Par défaut "anonyme". Il peut correspondre un utilisateur lambda non déclaré dans Shinken Enterprise.
`sticky`	Option présente dans Shinken Framework, dépréciée dans Shinken Entreprise. Si sticky=1, la prise en compte est automatiquement enlevée lorsque l'état est OK. Si sticky=2, la prise en compte est automatiquement enlevée lorsque l'état change, peu importe l'état (*Warning Critique par exemple*). Par défaut, sticky=1.
persistent	Option présente dans Shinken Framework, dépréciée dans Shinken Entreprise. Si persistent=1, la prise en compte sera toujours présente après un redémarrage des services Shinken. Dans Shinken Entreprise, ce comportement est celui par défaut et ne peut pas être changé.

Exemple:

```
curl -u user:password -X POST -d "time_stamp=$(date +%s)&host_name=host-checked&service_description=service-checked&comment=Nous sommes entrain de corriger le problème&notify=1" http://shinken-srv:7760/acknowledge
```

/downtime: Mettre une période de maintenance sur un hôte, un cluster ou un check

Paramètres de l'appel

Nom	Description
`Méthode HTTP`	`POST`
`time_stamp`	(*optionnel*) date pour que Shinken puisse déterminer a quel moment a eu lieu la mesure. Par défaut, ce sera la date d'appel de l'API
`host_name`	Nom de l'hôte cible
`service_description`	Nom du check cible (*si résultat d'un check*)
`comment`	Le commentaire de la période de maintenance (*optionnel*)
`start_time`	Sous forme de timestamp: correspond à l'heure du début de la période de maintenance
`end_time`	Sous forme de timestamp: correspond à l'heure de fin de la période de maintenance
`notify`	1 ou 0 pour notifier les contacts qui sont signalé sur l'équipement comme recevant les notifications de changement sur cette élément.
`author`	Par défaut "anonyme". Il peut correspondre un utilisateur lambda non déclaré dans Shinken Enterprise.

Exemple:

```
curl -u user:password -X POST -d "time_stamp=$(date +%s)&host_name=host-checked&service_description=service-checked&comment=Maintenance en cours&author=shinken_admin" http://shinken-srv:7760/downtime
```