HTTP Server

Contenu du connecteur

> Attention : Ce connecteur supporte seulement les authentifications BASIC ou NTLM v.1/2 (il n'est pas possible de faire des checks de pages web derrière des portails SSO).

Modèles

Le connecteur de supervision **HTTP Server** apporte un modèle d'hôte :

- **App-Protocol-HTTP-custom**

Le connecteur apporte les modèles de service suivants (classés selon le modèle d'hôte auquel ils sont rattachés) :

App-Protocol-HTTP-custom Non rattachés à un modèle d'hôte

Alias	Modèle de service	Description
HTTP-Response-Time	App-Protocol-HTTP-Response-Time-custom	Contrôle le temps de réponse d'une page web

> Les services listés ci-dessus sont créés automatiquement lorsque le modèle d'hôte **App-Protocol-HTTP-custom** est utilisé.

Métriques & statuts collectés

Voici le tableau des services pour ce connecteur, détaillant les métriques rattachées à chaque service.

Collection HTTP-Expected-Content HTTP-Json-Content HTTP-Response

Les métriques obtenues dépendent entièrement de la configuration de la collection. Vous pouvez vous reporter au tutoriel The Watch dédié pour ce mode.

Prérequis

La page ou application web interrogée doit être accessible via le protocole HTTP ou HTTPS depuis le collecteur. Il est possible d'utiliser un proxy lorsque cela est nécessaire.

Installer le connecteur de supervision

Pack

La procédure d'installation des connecteurs de supervision diffère légèrement suivant que votre licence est offline ou online.

1. Si la plateforme est configurée avec une licence *online*, l'installation d'un paquet n'est pas requise pour voir apparaître le connecteur dans le menu **Configuration > Connecteurs > Connecteurs de supervision**. Au contraire, si la plateforme utilise une licence *offline*, installez le paquet sur le **serveur central** via la commande correspondant au gestionnaire de paquets associé à sa distribution :

Alma / RHEL / Oracle Linux 8 Alma / RHEL / Oracle Linux 9 Debian 11 & 12 C

```
apt install centreon-pack-applications-protocol-http
```

2. Quel que soit le type de la licence (*online* ou *offline*), installez le connecteur **HTTP Server** depuis l'interface web et le menu **Configuration > Connecteurs > Connecteurs de supervision**.

Plugin

À partir de Centreon 22.04, il est possible de demander le déploiement automatique du plugin lors de l'utilisation d'un connecteur. Si cette fonctionnalité est activée, et que vous ne souhaitez pas découvrir des éléments pour la première fois, alors cette étape n'est pas requise.

> Plus d'informations dans la section Installer le plugin.

Utilisez les commandes ci-dessous en fonction du gestionnaire de paquets de votre système d'exploitation :

Alma / RHEL / Oracle Linux 8 Alma / RHEL / Oracle Linux 9 **Debian 11 & 12** C

```
apt install centreon-plugin-applications-protocol-http
```

Utiliser le connecteur de supervision

Utiliser un modèle d'hôte issu du connecteur

1. Ajoutez un hôte à Centreon depuis la page **Configuration > Hôtes**.
2. Complétez les champs **Nom**, **Alias** & **IP Address/DNS** correspondant à votre ressource.
3. Appliquez le modèle d'hôte **App-Protocol-HTTP-custom**. Une liste de macros apparaît. Les macros vous permettent de définir comment le connecteur se connectera à la ressource, ainsi que de personnaliser le comportement du connecteur.
4. Renseignez les macros désirées. Attention, certaines macros sont obligatoires.

Macro	Description	Valeur par défaut	Obligatoire
PROTOCOL	Specify https if needed	http	
PORT	Port used by web server	80	X
EXTRAOPTIONS	Any extra option you may want to add to every command (a --verbose flag for example). Toutes les options sont listées ici.		

5. Déployez la configuration. L'hôte apparaît dans la liste des hôtes supervisés, et dans la page **Statut des ressources**. La commande envoyée par le connecteur est indiquée dans le panneau de détails de l'hôte : celle-ci montre les valeurs des macros.

Utiliser un modèle de service issu du connecteur

1. Si vous avez utilisé un modèle d'hôte et coché la case **Créer aussi les services liés aux modèles**, les services associés au modèle ont été créés automatiquement, avec les modèles de services correspondants. Sinon, créez les services désirés manuellement et appliquez-leur un modèle de service.
2. Renseignez les macros désirées (par exemple, ajustez les seuils d'alerte). Les macros indiquées ci-dessous comme requises (**Obligatoire**) doivent être renseignées.

Collection HTTP-Expected-Content HTTP-Json-Content HTTP-Response

Macro	Description	Valeur par défaut	Obligatoire
CONFIG	Configuration file or JSON code to use		X
EXTRAOPTIONS	Any extra option you may want to add to the command (a --verbose flag for example). Toutes les options sont listées ici.		

3. Déployez la configuration. Le service apparaît dans la liste des services supervisés, et dans la page **Statut des ressources**. La commande envoyée par le connecteur est indiquée dans le panneau de détails du service : celle-ci montre les valeurs des macros.

Comment puis-je tester le plugin et que signifient les options des commandes ?

Une fois le plugin installé, vous pouvez tester celui-ci directement en ligne de commande depuis votre collecteur Centreon en vous connectant avec l'utilisateur **centreon-engine** (`su - centreon-engine`). Vous pouvez tester que le connecteur arrive bien à superviser une ressource en utilisant une commande telle que celle-ci (remplacez les valeurs d'exemple par les vôtres) :

```
/usr/lib/centreon/plugins/centreon_protocol_http.pl \
        --plugin=apps::protocols::http::plugin \
        --mode=response \
        --hostname=10.0.0.1 \
        --proto='http' \
```

```
        --port='80' \
        --urlpath='/' \
        --warning=''  \
        --critical=''
```

La commande devrait retourner un message de sortie similaire à :

```
OK: response time 0.078s | 'http.response.time.seconds'=0.078s;;;0;
'http.response.size.count'=49602B;;;0;
'http.response.resolve.time.milliseconds'=4.176ms;;;0;
'http.response.connect.time.milliseconds'=4.176ms;;;0;
'http.response.processing.time.milliseconds'=44.163ms;;;0;
'http.response.transfer.time.milliseconds'=4.176ms;;;0;
```

Diagnostic des erreurs communes

Rendez-vous sur la documentation dédiée pour le diagnostic des erreurs communes des plugins Centreon.

Modes disponibles

Dans la plupart des cas, un mode correspond à un modèle de service. Le mode est renseigné dans la commande d'exécution du connecteur. Dans l'interface de Centreon, il n'est pas nécessaire de les spécifier explicitement, leur utilisation est implicite dès lors que vous utilisez un modèle de service. En revanche, vous devrez spécifier le mode correspondant à ce modèle si vous voulez tester la commande d'exécution du connecteur dans votre terminal.

Tous les modes disponibles peuvent être affichés en ajoutant le paramètre `--list-mode` à la commande :

```
/usr/lib/centreon/plugins/centreon_protocol_http.pl \
        --plugin=apps::protocols::http::plugin \
        --list-mode
```

Le plugin apporte les modes suivants :

Mode	Modèle de service associé
collection [code]	App-Protocol-HTTP-Collection-custom
expected-content [code]	App-Protocol-HTTP-Expected-Content-custom

Mode	Modèle de service associé
json-content [code]	App-Protocol-HTTP-Json-Content-custom
response [code]	App-Protocol-HTTP-Response-Time-custom
soap-content [code]	App-Protocol-HTTP-Soap-Content-custom

Options disponibles

Options génériques

Les options génériques sont listées ci-dessous :

Option	Description
--mode	Define the mode in which you want the plugin to be executed (see--list-mode).
--dyn-mode	Specify a mode with the module's path (advanced).
--list-mode	List all available modes.
--mode-version	Check minimal version of mode. If not, unknown error.
--version	Return the version of the plugin.
--pass-manager	Define the password manager you want to use. Supported managers are: environment, file, keepass, hashicorpvault and teampass.
--verbose	Display extended status information (long output).
--debug	Display debug messages.
--filter-perfdata	Filter perfdata that match the regexp. Example: adding --filter-perfdata='avg' will remove all metrics that do not contain 'avg' from performance data.

Option	Description
--filter-perfdata-adv	Filter perfdata based on a "if" condition using the following variables: label, value, unit, warning, critical, min, max. Variables must be written either %{variable} or %(variable). Example: adding --filter-perfdata-adv='not (%(value) == 0 and %(max) eq "")' will remove all metrics whose value equals 0 and that don't have a maximum value.
--explode-perfdata-max	Create a new metric for each metric that comes with a maximum limit. The new metric will be named identically with a '_max' suffix). Example: it will split 'used_prct'=26.93%;0:80;0:90;0;100 into 'used_prct'=26.93%;0:80;0:90;0;100 'used_prct_max'=100%;;;;
--change-perfdata --extend-perfdata	Change or extend perfdata. Syntax: --extend-perfdata=searchlabel,newlabel,target[,[newuom],[min],[m ax]] Common examples: Convert storage free perfdata into used: --change-perfdata=free,used,invert() Convert storage free perfdata into used: --change-perfdata=used,free,invert() Scale traffic values automatically: --change-perfdata=traffic,,scale(auto) Scale traffic values in Mbps: --change-perfdata=traffic_in,,scale(Mbps),mbps Change traffic values in percent: --change-perfdata=traffic_in,,percent()
--extend-perfdata-group	Add new aggregated metrics (min, max, average or sum) for groups of metrics defined by a regex match on the metrics' names. Syntax: --extend-perfdata-group=regex,namesofnewmetrics,calculation[,[ne wuom],[min],[max]] regex: regular expression namesofnewmetrics: how the new metrics' names are composed (can use $1, $2... for groups defined by () in regex). calculation: how the values of the new metrics should be calculated newuom (optional): unit of measure for the new metrics min (optional): lowest value the metrics can reach max (optional): highest value the metrics can reach Common examples: Sum wrong packets from all interfaces (with interface need --units-errors=absolute): --extend-perfdata-group=',packets_wrong,sum(packets_(discard | error)_(in|out))' Sum traffic by interface: --extend-perfdata-group='traffic_in_(.*),traffic_$1,sum(traf fic_(in|out)_$1)'
--change-short-output	Modify the short/long output that is returned by the plugin. Syntax: --change-short-output=pattern~~replacement~~modifier Most commonly used

Option	Description
--change-long-output	modifiers are i (case insensitive) and g (replace all occurrences). Example: adding --change-short-output='OK~~Up~~gi' will replace all occurrences of 'OK', 'ok', 'Ok' or 'oK' with 'Up'
--change-exit	Replace an exit code with one of your choice. Example: adding --change-exit=unknown=critical will result in a CRITICAL state instead of an UNKNOWN state.
--range-perfdata	Rewrite the ranges displayed in the perfdata. Accepted values: 0: nothing is changed. 1: if the lower value of the range is equal to 0, it is removed. 2: remove the thresholds from the perfdata.
--filter-uom	Mask the units when they don't match the given regular expression.
--opt-exit	Replace the exit code in case of an execution error (i.e. wrong option provided, SSH connection refused, timeout, etc). Default: unknown.
--output-ignore-perfdata	Remove all the metrics from the service. The service will still have a status and an output.
--output-ignore-label	Remove the status label ("OK:", "WARNING:", "UNKNOWN:", CRITICAL:") from the beginning of the output. Example: 'OK: Ram Total:...' will become 'Ram Total:...'
--output-xml	Return the output in XML format (to send to an XML API).
--output-json	Return the output in JSON format (to send to a JSON API).
--output-openmetrics	Return the output in OpenMetrics format (to send to a tool expecting this format).
--output-file	Write output in file (can be combined with json, xml and openmetrics options). E.g.: --output-file=/tmp/output.txt will write the output in /tmp/output.txt.
--disco-	Applies only to modes beginning with 'list-'. Returns the list of available

Option	Description
format	macros to configure a service discovery rule (formatted in XML).
--disco-show	Applies only to modes beginning with 'list-'. Returns the list of discovered objects (formatted in XML) for service discovery.
--float-precision	Define the float precision for thresholds (default: 8).
--source-encoding	Define the character encoding of the response sent by the monitored resource Default: 'UTF-8'.

Options des modes

Les options disponibles pour chaque modèle de services sont listées ci-dessous :

Collection HTTP-Expected-Content HTTP-Json-Content HTTP-Response

Option	Description
--config	Configuration file or JSON code to use (required).
--filter-selection	Filter selections. Example: --filter-selection='name=test'
--constant	Add a constant. Example: --constant='warning=30' --constant='critical=45'

Pour un mode, la liste de toutes les options disponibles et leur signification peut être affichée en ajoutant le paramètre `--help` à la commande :

```
/usr/lib/centreon/plugins/centreon_protocol_http.pl \
        --plugin=apps::protocols::http::plugin \
        --mode=response \
        --help
```

Dernière mise à jour le **24 mars 2026**