# Meters To Home Automation (MetersToHA / Meters2HA / M2HA)

Ce script automatise le transfert de l’information de consommation d’eau et de gaz vers des systèmes domotiques tels que [Home Assistant](https://www.home-assistant.io/) et [Domoticz](https://domoticz.com/) et d’autres en MQTT.

C’est un fork de [veolia-idf](https://github.com/s0nik42/veolia-idf) déjà connu et accélérant le développement.

L’idée est que MetersToHA s’enrichit d’autre collecteurs et systèmes domotiques afin d’éviter la multiplication de modules à installer - à vous de jouer.

## Fonctionnalités :

* Récupération des valeurs de consommation au fil du temps;
* Gestion multi-contrat (veolia-idf)
* Vérification de l’intégrité de l’environnement (prérequis / configuration sur serveur domotique)
* Mode débogue graphique
* Possible intégration avec d’autre solution domotique (à vous de jouer)

## Table des Matières

* [Meters To Home Automation (MetersToHA / Meters2HA / M2HA)](#X57326979d50cc632bddf6abbd7a5d06323e06f1)
  + [Fonctionnalités :](#fonctionnalit%C3%A9s-)
  + [Table des Matières](#table-des-mati%C3%A8res)
  + [⚠️ Limitations](#warning-limitations)
  + [Informations générales](#informations-g%C3%A9n%C3%A9rales)
  + [Le fichier de configuration (config.json)](#le-fichier-de-configuration-configjson)
  + [Les paramètres du script](#les-param%C3%A8tres-du-script)
  + [Home Assistant](#home-assistant)
    - [Configuration](#configuration)
    - [AppDaemon](#appdaemon)
      * [Installation AppDaemon](#installation-appdaemon)
      * [Ajouter MetersToHA à l’AppDaemon avec HACS](#X7a50271d826192c15ded2244e1f30c88af30e91)
      * [Configuration MetersToHA sous AppDaemon](#configuration-meterstoha-sous-appdaemon)
      * [Débogue avec AppDaemon](#d%C3%A9bogue-avec-appdaemon)
      * [Lancer un appel à Veolia (AppDaemon)](#lancer-un-appel-%C3%A0-veolia-appdaemon)
      * [GRDF (AppDaemon)](#grdf-appdaemon)
      * [Automatisation Home Assistant (AppDaemon)](#automatisation-home-assistant-appdaemon)
    - [Ajout des informations au tableau “Énergie”](#Xc1ff5952d331a2ddb6ca47faf0c33ea05ed90e0)
  + [Domoticz](#domoticz)
    - [Création du “Virtual Sensor” sur Domoticz :](#X02840fc0699690ed7197fbd6a77c0285196872f)
    - [Spécificités de configuration](#sp%C3%A9cificit%C3%A9s-de-configuration)
  + [MQTT](#mqtt)
  + [Fournisseurs](#fournisseurs)
    - [Veolia](#veolia)
    - [GRDF](#grdf)
  + [“Serveurs”](#serveurs)
    - [Docker - “configuration système incluse”](#Xf38d47b5fc300c9da1b499fd8e41e5241b1bdfc)
    - [Installation “direct”](#installation-direct)
    - [Installation avec le sous-système Windows pour Linux (WSL)](#Xbfa259d36ed1a3f9ab173ca62d801711b93169a)
    - [Installation native sous Windows](#installation-native-sous-windows)
      * [Planification de tâche](#planification-de-t%C3%A2che)
    - [Installation de MetersToHA](#installation-de-meterstoha)
      * [Installation avec git](#installation-avec-git)
      * [Installation avec une archive](#installation-avec-une-archive)
    - [Automatisation :](#automatisation-)
  + [Dépannage](#d%C3%A9pannage)
    - [Première exécution :](#premi%C3%A8re-ex%C3%A9cution-)
    - [Paramétrer votre système pour le mode débogue (optionnel, mais recommandé)](#Xa574eb398e0c089aec0b026d5872ba3d5bd3e6e)
  + [Principe de fonctionnement](#principe-de-fonctionnement)
  + [Environnements testés:](#environnements-test%C3%A9s)
  + [Cas de figure:](#cas-de-figure)
  + [Remerciements/Contributeurs](#remerciementscontributeurs)

## ⚠️ Limitations

* GRDF n’est pas encore compatible avec Domoticz;
* GRDF utilise un captcha. Il peut être validé “tout seul” dans certains cas (si peu de requêtes par jour), ou nécessiter une résolution interactive, ou un service de résolution de captcha payant.
* Si vous migrez depuis [frtz13/homeassistant\_gazpar\_cl\_sensor](https://github.com/frtz13/homeassistant_gazpar_cl_sensor), vous devez désactiver ces automatisations et vos “sensors” de type template (dans configuration.yaml) - sinon les valeurs écrasent celles de MetersToHA.

## Informations générales

En résumé, il y a 4 étapes pour la mise en place:

* 1. Installation sur Linux ou Windows
* Distribution classique: Debian, Ubuntu, Alpine, etc. - physique ou Machine Virtuelle (VM);
* Docker - conteneurisation;
* Debian/Ubuntu avec le sous-système Windows pour Linux (WSL);
* Au sein/en parallèle de votre système domotique (AppDaemon pour Home Assistant par exemple).
* Utilisation sous Windows (hors Linux/WSL), avec navigation visible.

1. Configuration de MetersToHA - Fichier config.json avec les logins, etc.
2. Mise au point (essais, correction de configuration).
3. Automatisation (exécution régulière du script).

Ce document continue avec l’explication de la configuration: c’est commun à toutes les installations.  
Ensuite il explique comment l’utiliser avec le système domotique.  
Et au final il aborde les environnements d’exécution/installation.

Ce script fonctionne pour:

* [Veolia IDF - https://www.vedif.eau.veolia.fr](https://www.vedif.eau.veolia.fr/) valable pour Veolia en IDF.  
  Vous pouvez trouver votre portail en fonction de la ville en visitant [https://www.eau.veolia.fr](https://www.eau.veolia.fr/)> CONNECTEZ-VOUS.
* [GRDF - https://monespace.grdf.fr](https://monespace.grdf.fr/client/particulier/accueil).

## Le fichier de configuration (config.json)

Dans tous les cas il faut un fichier de configuration. Pour Home Assistant, le point de départ peut être config.json.exemple.home-assistant que vous renommez en config.json (ou autre).

Attention, AppDaemon efface le répertoire “MetersToHA” lors de la mise à jour - vous devez donc obligatoirement loger ce fichier ailleurs.

Exemple de configuration:

{  
 "veolia\_login": "MON\_LOGIN\_VEOLIA@mon.domaine",  
 "veolia\_password": "MONPASSEVEOLIA",  
 "veolia\_contract": "MONCONTRATVEOLIA",  
 "grdf\_login": "MON\_LOGIN\_GRDF@mon.domaine",  
 "grdf\_password": "XXXXXXXXXXXX",  
 "grdf\_pce": "21546000000000",  
 "ha\_server": "https://MONINSTANCEHA",  
 "ha\_token": "MONTOKEN.XXXXXXX.XXXXX-XXXXXXX",  
 "captchaai\_token": "XXXXXXXXXXX",  
 "type": "ha",  
 "timeout": "30"  
}

Les fournisseurs consultables dépendent des paramètres renseignés.  
C.a.d. qu’il convient de supprimer les clefs inutiles, remplacer “captchaai\_token” par “2captcha\_token” ou “capmonster\_token” en fonction de votre service, modifiez la valeur de “type” en fonction de votre plateforme.

Explication des champs:

* veolia\_login, veolia\_password: veolia\_contract:  
  Les informations de login sur le site de Veolia IDF, et le numéro de votre contrat.  
  Seulement pour Veolia Ile-de-France.
* grdf\_login, grdf\_password, grdf\_pce: Les informations de login sur le site de GRDF. L’identification du Point de Comptage et Estimation (PCE) est optionnel (actuellement).  
  Seulement pour GRDF.
* ha\_server: le domaine + le port accessibles depuis là ou le script tourne.  
  Cela peut être <http://homeassistant.local:8123> dans le cas ou vous n’avez pas touché la configuration réseau.
* ha\_token: voir ci-dessous, permet d’accéder à Home Assistant depuis le script.  
  Seulement pour Home Assistant
* Paramètre pour résolution de captcha.  
  Seulement si vous souhaitez résoudre les captchas automatiquement (GRDF, sinon vous devez utiliser –debug qui nécessite un serveur X).  
  Un débat assez complet concernant ce type de service est dans [cet issue d’un autre projet](https://github.com/iv-org/invidious/issues/1256). Inutile d’ouvrir un issue de débat ici, sauf pour proposer une alternative s’appuyant (moins) sur l’humain.  
  Pour Veolia, vous n’avez pas besoin de ce type de service.  
  1,07 capthas/jour ont du être résolus en moyenne dans une configuration ou Meters2HA est exécuté jusqu’à deux fois par soir (au cas ou le premier appel n’a pas donné de résultat). Pour 27% des jours, aucune résolution n’était nécessaire, pour 55% une seule résolution, et pour 18%, 2 résolutions.  
  Deux services sont compatibles, ajouté le paramètre du service choisi, les estimations sont sur la base des tarifs et observations en aout 2023:
  + captchaai\_token:  
    1 Thread (fil de traitement) offert pour les utilisateurs de Meters2HA. Il faaut:
    1. Suivre ce [lien d’affiliation](https://captchaai.com/?from=151169) pour s’insrire.
    2. Demander de bénéficier de [l’offre](https://github.com/mdeweerd/MetersToHA/issues/8#issuecomment-1704032953) à travers un Ticket ([Home> Dashboard> Tickets](https://captchaai.com/tickets.php#?from=151169)> New Ticket):
  + I would like to benefit from the offer  
     " 1 Free Thread for MetersToHA Users".
    1. Ajouter la clef API à votre configuration (pas besoin d’attendre l’activation).
    2. Attendre l’activation de l’offre.
    3. Profitez de la résolution reCAPTCHA 100%, hCaptcha, Solve Media, and 27500+ captchas d’image.
    4. Tester et mettre en route MetersToHA.
  + 2captcha\_token: à obtenir sur [2captcha.com](https://2captcha.com/?from=16639177). Montant minimum 3€ pour environ 1000 captchas (paypal), soit 2.5 années. 2captcha indique que les captchas sont résolus par des personnes.
  + capmonster\_token: à obtenir sur [capmonster.cloud](https://capmonster.cloud/).  
    Montant minimum $6 pour environ 10000 captchas (paypal), soit 25 ans. Zennolabs indique que les captchas sont résolus automatiquement (par machine).
* type: “ha” pour Home Assistant, “domoticz” pour Domoticz, “url” pour écrire vers un fichier ou “POST”er vers une URL, “mqtt” pour MQTT.
* url: Si type est ‘url’, url de type “file://chemin/vers/fichier/local.extension” ou “https://domaine.url/pourPOST”
* timeout: Le délai en secondes que le script attend pour certaines étapes.
* mqtt\_server, mqtt\_port, mqtt\_login, mqtt\_password: Serveur, port et identifiant pour MQTT.

Ne mettez que les valeurs utiles. Si vous consultez seulement Veolia IDF avec Home Assistant, ceci suffit:

{  
 "veolia\_login": "MON\_LOGIN\_VEOLIA@mon.domaine",  
 "veolia\_password": "MONPASSEVEOLIA",  
 "veolia\_contract": "MONCONTRATVEOLIA",  
 "ha\_server": "https://MONINSTANCEHA",  
 "ha\_token": "MONTOKEN.XXXXXXX.XXXXX-XXXXXXX",  
 "type": "ha",  
 "timeout": "30"  
}

## Les paramètres du script

| option(s) | Description |
| --- | --- |
| -h, –help | Affiche l’aide |
| -r, –run | Exécute le script |
| –version | Affiche la version du programme |
| –version-check | Vérifie s’il y a une nouvelle version du script (inactif) |
| –veolia | Récupère les données de Veolia IDF |
| –grdf | Récupère les données auprès de GRDF |
| -d, –debug | Active l’interface graphique interactif (Serveur X nécessaire) |
| –screenshot | Prend une ou plusieurs captures d’écran du navigateur (pour débogue) |
| –local-config | Utilise un répertoire local pour la configuration navigateur |
| -l LOGS\_FOLDER, –logs-folder LOGS\_FOLDER | Dossier pour les fichiers des traces |
| -c CONFIG, –config CONFIG | Fichier de configuration |
| -k, –keep-output | Garde les fichiers récupérés |
| –insecure | Ignore les erreurs de certificat du système domotique (utile pour les certificats SSL auto-signés) |
| –server-type SERVER\_TYPE | Type de destination ‘url’, ‘ha’, ‘dom’, ‘mqtt’. Si ‘url’, le paramètre ‘–url’ est nécessaire |
| –url URL | Destination du fichier récupéré: Autre fichier (file://…) ou URL web pour une requête POST (http(s)://…) |
| –skip-download | Ne télécharge pas le fichier mais utilise le fichier déjà en place, utile pour utiliser un fichier téléchargé interactivement ou bien pour le débogue. |
| –chrome-version CHROME\_VERSION | La version (principale) de chrome lorsque undetected-chromedriver est installé. Un nombre tel que 109, 110, … . |

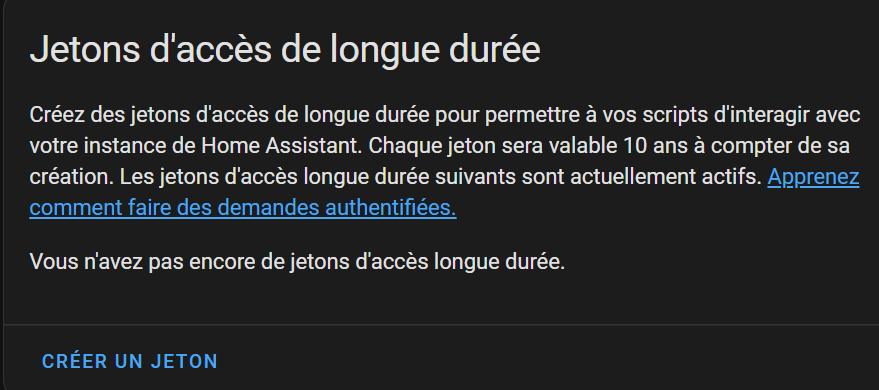
Lorsqu’une option est à la fois disponible dans le fichier de configuration que comme option en ligne de commande, la ligne de commande prend la priorité.  
Par exemple, l’option server-type est la même que type dans le fichier de configuration. Ce qui permet par exemple d’effectuer un transfert vers un web service en spécifiant --server-type url --url https://mon-service.web --skip-download.

## Home Assistant

En sus des informations qui suivent, [Le forum HACF](https://forum.hacf.fr/t/veolia-idf-sedif-consommation-eau-potable-suivi-consommation/11492) peut également vous aider.

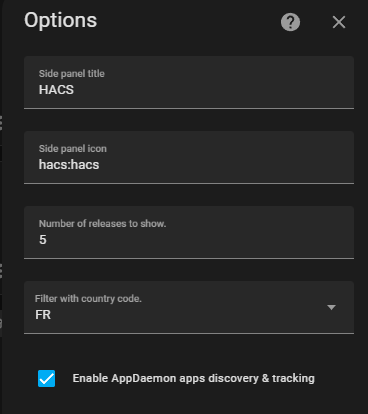
### Configuration

La valeur pour ha\_token est nécessaire et peut être obtenu dans [son profil Home Assistant](https://my.home-assistant.io/redirect/profile/). Cette fonctionnalité est disponible tout en bas de la page en question ou il faut cliquer “Créer un jeton”:



### AppDaemon

L’installation avec [AppDaemon](https://appdaemon.readthedocs.io/en/latest/INSTALL.html) semble le plus approprié si vous avez HomeAssistant OS (HAOS) puisque tout tournera sur la même machine (moyennant à peu près 500Mo pour installer AppDaemon et les paquets).  
Vous pouvez tout aussi bien utiliser les autres environnements.

Une fois AppDaemon installé, vous pourrez ajouter MetersToHA [HACS](https://hacs.xyz/) après [activation des applications AppDaemon dans HACS](https://hacs.xyz/docs/categories/appdaemon_apps/#enable-appdaemon-apps-in-hacs) comme ceci en suivant “Configuration” pour HACS dans “Paramètres>Appareils et services”: .

Les scripts sont placés dans le répertoire ../config/appdaemon/apps/meters-to-ha. Le fichier meters-to-ha-appdaemon.py assure l’intégration sous AppDaemon. meters\_to\_ha.py est le script indépendant et commun à toute installation.

#### Installation AppDaemon

1. [Ajouter le AddON/Module complémentaire « Home Assistant Community Add-on: AppDaemon »](https://appdaemon.readthedocs.io/en/latest/INSTALL.html) selon votre système, ou [suivez ces instructions pour HAOS](https://github.com/hassio-addons/addon-appdaemon/blob/main/appdaemon/DOCS.html#installation) ou aller directement vers le lien (en remplaçant ‘VOTREINSTANCE’): VOTREINSTANCE/hassio/addon/a0d7b954\_appdaemon/info , puis « Install »
2. Pour HAOS (et peut-être d’autres), configurer AppDaemon au moins avec ces paquets (configuration au format yaml pour HAOS):

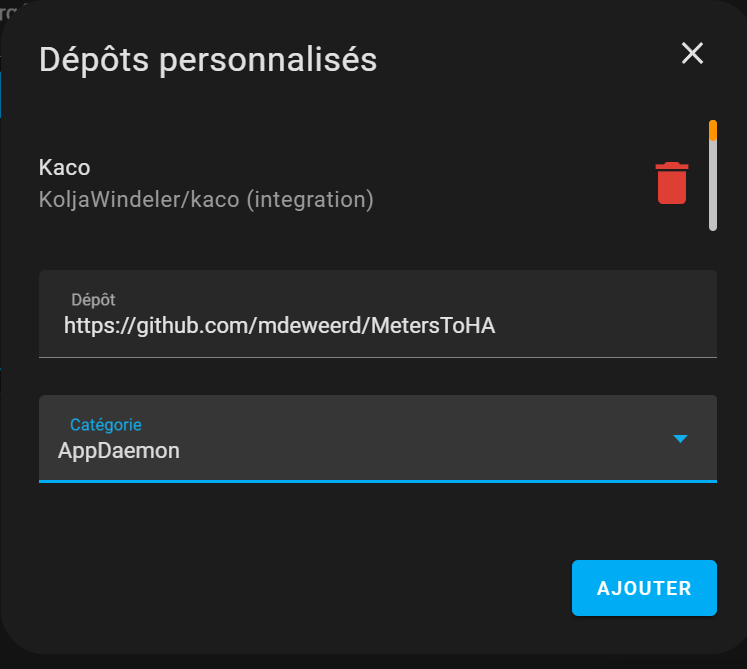
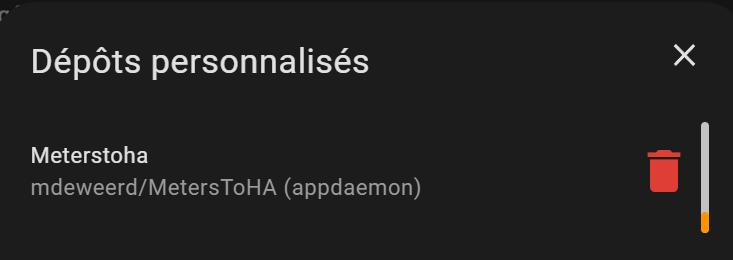
init\_commands: []  
python\_packages:  
 - selenium  
 - PyVirtualDisplay  
system\_packages:  
 - py-urllib3  
 - py3-colorama  
 - xvfb  
 - py3-pip  
 - xorg-server-xephyr  
 - chromium-chromedriver  
 - chromium  
 - py3-openssl  
 - py3-pysocks  
 - py3-wsproto  
 - py3-sniffio  
 - py3-async\_generator  
 - py3-sortedcontainers  
 - py3-attrs  
 - py3-outcome  
 - py3-trio

* Activer le Watchdog du AddOn, Démarrer le AddOn

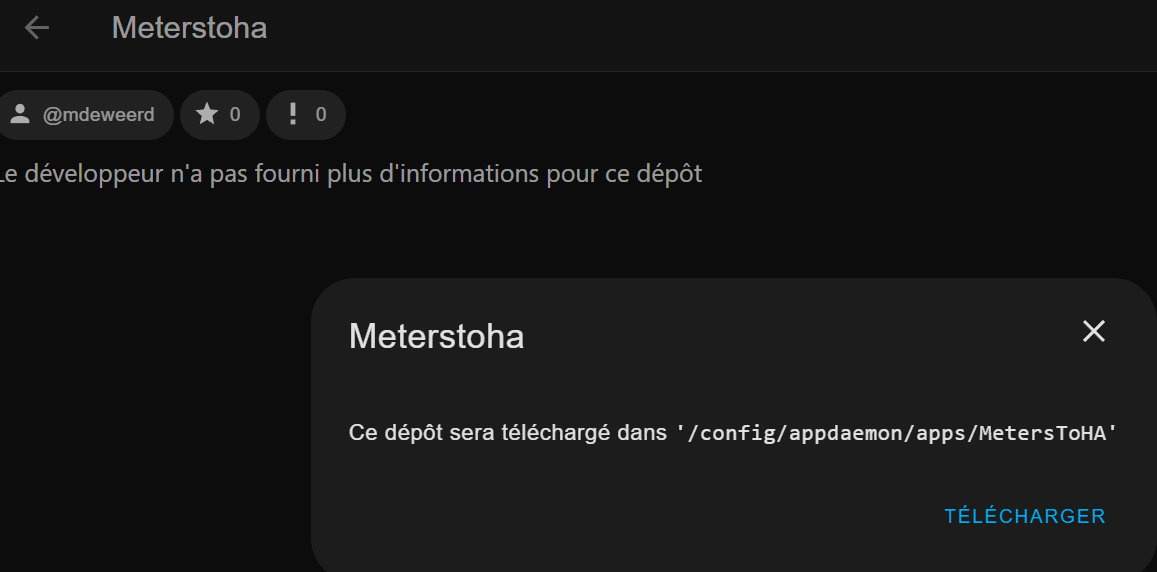
#### Ajouter MetersToHA à l’AppDaemon avec HACS

Cette procédure suppose que HACS est déjà actif et configuré pour AppDaemon. Ainsi que AppDaemon soit activé.

1. Ajouter [GitHub - mdeweerd/MetersToHA](https://github.com/mdeweerd/MetersToHA) comme source de type AppDaemon:

* 
* Note : après l’ajout, le popup reste affiché. Le nouveau module est disponible à la fin de la liste:
* 

1. Ensuite « télécharger » ce dépôt avec HACS - chercher meterstoha parmi les « AppDaemons » et cliquez Télécharger ou Download:

* 

Les scripts sont ainsi disponibles pour AppDaemon.

#### Configuration MetersToHA sous AppDaemon

Reste encore la configuration de MetersToHA sous AppDaemon. Plus haut la création du fichier config.json a été expliquée. Vous devez le déposer sur votre instance Home Assistant, de préférence dans un sous-répertoire de .../config.

Dans l’exemple ci-dessous il est supposé que ce fichier config.json est disponible au chemin /config/config.json.

Cette configuration indique aussi que la trace veolia.log sera déposé sous /config. Le fichier /config/veolia.log pourra aider à identifier des causes de dysfonctionnement.

Voici un exemple d’une configuration minimale à ajouter à /config/appdaemon/apps/apps.yaml - l’événement déclencheur est par défaut call\_meters\_to\_ha:

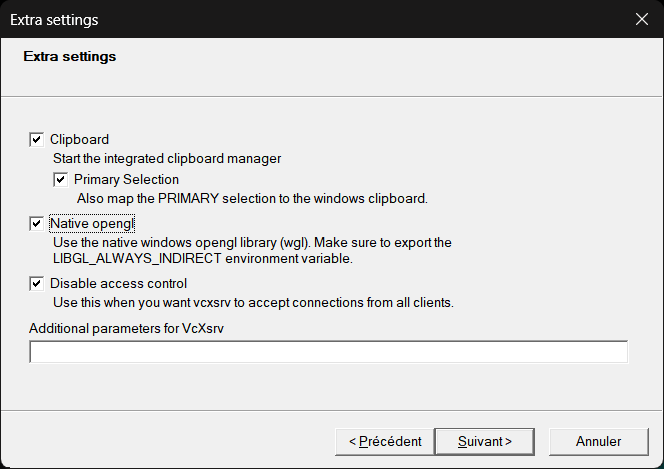
meters\_to\_ha:  
 module: meters\_to\_ha\_appdaemon  
 class: MetersToHA  
 config\_file: /config/config.json

L’exemple suivant montre l’ensemble des arguments disponibles, dont la précision du chemin vers le script meters\_to\_ha.py, tout en spécialisant pour un appel à Veolia seulement, et avec le débogue actif qui suppose un serveur X actif et disponible sur l’IP indiqué.

veolia\_idf:  
 module: meters\_to\_ha\_appdaemon  
 class: MetersToHA  
 # optionnel - Par défaut "call\_meters\_to\_ha".  
 # Permet de définir plusieurs lancements distincts, par exemple  
 # pour consulter Veolia à une certaine heure, et GRDF à une autre heure.  
 event\_name: call\_veolia  
 # optionnel - Par exemple --grdf pour ne faire que la requête auprès de GRDF  
 # --veolia pour ne faire la requête qu'auprès de Veolia  
 # --insecure pour accepter les certificats SSL non vérifiés  
 # (par exemple autosigné).  
 extra\_opts: [--veolia]  
 # optionnel  
 log\_folder: /config  
 # optionnel (Par défaut: "config.json" dans le répertoire de `meters\_to\_ha.py`)  
 config\_file: /config/meters\_to\_ha.json  
 # optionnel (Par défaut: "<REALMODULESCRIPTPATH>/meters\_to\_ha.py")  
 # script: /config/meters\_to\_ha/meters\_to\_ha.py  
 # optionnel (Par défaut: false) - add --keep-output option  
 keep\_output: true  
 # optionnel (Par défaut: false) - add --debug option - nécessite DISPLAY & serveur X!!  
 debug: true  
 # optionnel (Par défaut: None) - Set DISPLAY for GUI interface (when debug is true)  
 DISPLAY: 192.1.0.52:0  
 # optionnel (Par défaut: None) - Fichier pour la sortie STDOUT du script  
 outfile: /config/appdaemon/apps/meters\_to\_ha\_script.log  
 # optionnel (Par défaut: None) - Fichier pour la sortie STDERR du script  
 errfile: /config/appdaemon/apps/meters\_to\_ha\_err.log

L’option debug peut être intéressant lors de la mise en place en cas de diffucultés mais nécessite un serveur X, la bonne configuration de DISPLAY et l’autorisation d’accès depuis la machine.  
Par exemple avec

* [Mobaxterm Portable](https://mobaxterm.mobatek.net/download-home-edition.html).  
  Recommandé car:
  + “Sans installation”;
  + Lance un Serveur X automatiquement;
  + Un popop pour demander l’autorisation lorsque le process tente de se connecter;
  + Il suffit alors de définir DISPLAY à <IP\_OU\_NOM\_RESEAU\_PC>:0 après avoir lancé ce logiciel et accepté l’accès aux réseaux privés.
* [VcXsvr sous Windows](https://sourceforge.net/projects/vcxsrv/files/vcxsrv/) vous devez cocher la case Disable access control si l’exécution se fait depuis une autre machine (pas si c’est sur la même machine sous Docker).



#### Débogue avec AppDaemon

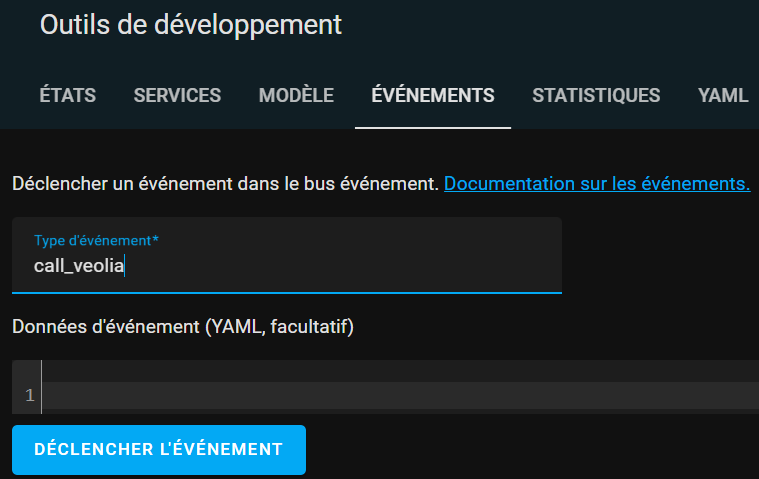
Pour info, il y a une interface web spécifique à AppDaemon (port 5050 par défaut) : [http://votreinstance:5050](http://votreinstance:5050/) qui donne entre outre accès à qqs traces et l’historique des appels de scripts.

Sur la page [http://votreinstance:5050/aui/index.html#/logs] on peut trouver par exemple des traces. Exemple avec une erreur:

2022-12-10 13:29:13.182428 ERROR veolia\_idf: Done MetersToHA  
2022-12-10 13:29:13.157362 ERROR veolia\_idf: NameError("name 'sys' is not defined")  
2022-12-10 13:29:13.140371 ERROR veolia\_idf: Start MetersToHA  
2022-12-10 13:29:09.467062 INFO AppDaemon: Initializing app veolia\_idf using class MetersToHA from module meters\_to\_ha

#### Lancer un appel à Veolia (AppDaemon)

L’appel est lancé en déclenchant l’événement call\_meters\_to\_ha (ou l’événement défini sous le paramètre event\_name). Cela peut être fait dans une automatisation (ce qui permet de le lancer selon un planning par exemple), ou de façon interactive dans les outils de développement. L’exemple est avec call\_veolia (2ième exemple de configuration plus haut):



Une trace est systématiquement créé comme service.log, soit à l’emplacement du script, soit dans le répertoire donné par log\_folder: . Cela peut déjà aider à identifier les causes, ou tout simplement suivre le bon déroulement du script.

Extrait de la fin d’une trace:

2022-06-01 18:31:55,541 : -- : Parsing csv file  
2022-06-01 18:31:55,813 : OK : update value for 2022-05-31  
2022-06-01 18:31:56,014 : OK : Close Browser  
2022-06-01 18:31:56,018 : OK : Close Display  
2022-06-01 18:31:56,019 : -- : Remove downloaded file historique\_jours\_litres.csv Finished on success

#### GRDF (AppDaemon)

La configuration c’est presque comme pour Veolia IDF. Comme la consommation GAZPAR est plutôt disponible en fin de journée, il est intéressant de consulter GRDF vers 21h par exemple.  
Je recommande donc de personnaliser l’event\_name.

Pour GRDF un captcha est présent sur la page et depuis Janvier 2023 les scripts “simples” ne suffisent plus.

La résolution du captcha se fait soit manuellement (avec débugue actif et configuration de DISPLAY), soit en s’appuyant sur [2captcha.com](https://2captcha.com/?from=16639177).

grdf:  
 module: meters\_to\_ha  
 class: MetersToHA  
 # optionnel - Par défault "call\_meters\_to\_ha".  
 # Permet de définir plusieurs lancements distincts, par exemple  
 # pour consulter Veolia à une certaine heure, et GRDF à une autre heure.  
 event\_name: call\_grdf  
 # extra\_opts - Paramètres complémentaires pour la ligne de commande (optionnel)  
 # --grdf: Consulter GRDF  
 # --veolia: Consulter Veolia IDF  
 # --screenshot: Prendre une capture d'écran avant connexion.  
 extra\_opts: [--grdf, --screenshot]  
 # optionnel - Emplacement des fichiers de trace, screenshot.  
 log\_folder: /config  
 # optionnel (Par défaut: "config.json" dans le répertoire de `meters\_to\_ha.py`)  
 config\_file: /config/meters\_to\_ha.json  
 # optionnel (Par défaut: "<REALMODULESCRIPTPATH>/meters\_to\_ha.py")  
 # script: /config/meters\_to\_ha/meters\_to\_ha.py  
 # optionnel (Par défaut: false) - add --keep-output option  
 keep\_output: true  
 # optionnel (Par défaut: false) - add --debug option - nécessite DISPLAY & serveur X!!  
 debug: true  
 # optionnel (Par défaut: None) - Set DISPLAY for GUI interface (when debug is true)  
 DISPLAY: 192.1.0.52:0  
 # optionnel (Par défaut: None) - Fichier pour la sortie STDOUT du script  
 outfile: /config/appdaemon/apps/meters\_to\_ha\_script.log  
 # optionnel (Par défaut: None) - Fichier pour la sortie STDERR du script  
 errfile: /config/appdaemon/apps/meters\_to\_ha\_err.log

Configuration typique:

grdf:  
 module: meters\_to\_ha  
 class: MetersToHA  
 event\_name: call\_grdf  
 extra\_opts: [--grdf, --screenshot]  
 log\_folder: /config  
 config\_file: /config/meters\_to\_ha.json  
 keep\_output: true

#### Automatisation Home Assistant (AppDaemon)

Pour réaliser la tache de récupération une fois par jour, vous pouvez ajouter un automatisme à votre configuration Home Assistant comme ceci:

alias: Veolia  
description: Déclencher l'événement qui démarre l'application MetersToHa sous AppDaemon  
trigger:  
 - platform: time\_pattern  
 hours: '1'  
 minutes: '7'  
 alias: Déclenchement à partir de l'heure choisie  
condition: []  
action:  
 - delay: '{{ range(0, 90\*60+1) | random }}'  
 alias: Avec un délai variable pour ne pas charger le serveur tous en même temps.  
 - event: call\_meters\_to\_ha  
 event\_data: {}  
 alias: Déclenche l'événement définit dans la configuration 'AppDaemon'  
mode: single

Cela récupère la consommation dans les 90 minutes suivant 1h07 en émettant l’événement call\_meters\_to\_ha ce qui déclenche le script sous AppDaemon. Il mettre en place une automatisation par fournisseur (avec événements différents) si vous souhaitez des horaires différents. Prenez en compte un délai de minimum 5 minutes entre les 2 événements (pour limiter les ressources utilisées sur votre système).

Exemple pour GRDF ou l’on tente de récupérer les données jusqu’à deux fois, avec une condition qui vérifie que la dernière mise à jour était il y a plus de 17h.

alias: Appel GRDF  
description: ''  
trigger:  
 - platform: time\_pattern  
 hours: '21'  
 minutes: '1'  
 seconds: '0'  
 - platform: time\_pattern  
 hours: '23'  
 minutes: '1'  
 seconds: '0'  
condition:  
 - condition: template  
 value\_template: >-  
 {{  
 (as\_timestamp(now())-as\_timestamp(states.sensor.gas\_consumption\_kwh.last\_updated|default('1970-01-01')))  
 > 17\*3600 }}  
action:  
 - delay: '{{ range(0, 55\*60+1) | random }}'  
 alias: Avec un délai variable pour ne pas charger le serveur tous en même temps.  
 - event: call\_grdf  
 event\_data: {}  
mode: single

### Ajout des informations au tableau “Énergie”

Quel que soit la méthode pour lancer le script, il convient de configurer votre tableau “Énergie” pour le suivre dans Home Assistant.

Pour cela, accédez à la [page de configuration du tableau “Énergie”](https://my.home-assistant.io/redirect/config_energy/).

Ajouter les nouveaux compteurs (eau, gaz) dans les bonnes classes. Il convient de choisir les totaux ici (pas les entités daily). Les compteurs permettent d’afficher la consommation journalière facilement sur d’autres pages que le Tableau Énergie.

La documentation officielle indique qu’il faut attendre deux heures pour voir apparaître la consommation sous le panneau Énergie. Mais ce sera plus car la première valeur sert de référence.

## Domoticz

Prérequis :

* “Virtual Sensor” sur Domoticz;
* Une installation de type “serveur” ou Docker.

### Création du “Virtual Sensor” sur Domoticz :

* Créer un Matériel de Type “Dummy”: Domoticz> Setup> Hardware> Dummy
* Créer un “Virtual Sensor” de type : “Managed Counter”
* Configurer le sensor: Domoticz> Utility> Bouton "edit" de votre sensor

| > | Sensor pour conso eau |
| --- | --- |
| **Type Counter** | water |
| **Counter Divider** | 1000 |
| **Meter Offset** | 0 |

### Spécificités de configuration

Outre que la configuration des informations fournisseur et éventuelle clef pour les captchas, vous devez définir les champs suivants:

| Clef JSON | Exemple | Description |
| --- | --- | --- |
| **“domoticz\_server”** | http://127.0.0.1:8080/ | Url du serveur Domoticz |
| **“domoticz\_idx”** | 123 | Le numero du “virtual sensor” Domoticz crée (se trouve dans : DomoticzDevices (Colonne Idx) |

## MQTT

Non testé.

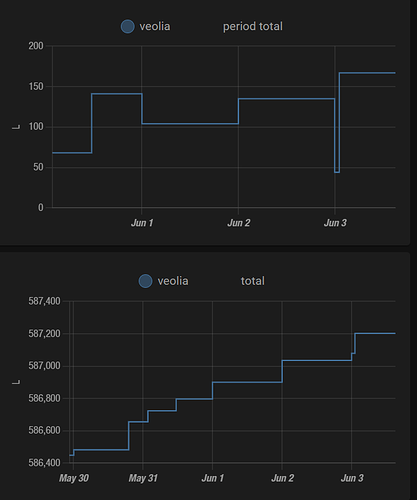
## Fournisseurs

### Veolia

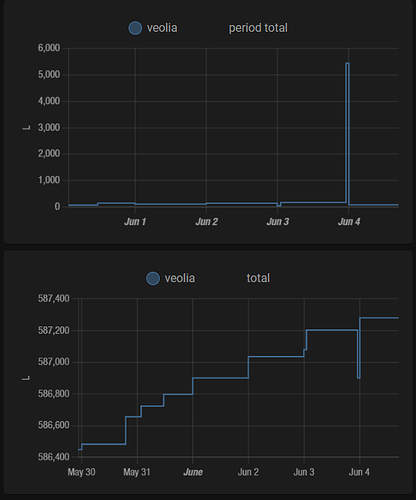
Il semblerait que les données restituées par Veolia sont des fois un peu “farfelus”. La meilleure méthode connue pour éviter cela est de contournement c’est de réaliser l’appel entre 1h du matin et minuit.

Le délai variable permet de repartir l’heure d’appel à Veolia entre les utilisateurs pour ne pas encombre le service. Vous pouvez aussi/en sus définir une heure différente de 1h07 dans votre configuration Vous pouvez sûrement accepter de récupérer l’information un peu plus tard que cela vu qu’elle est de tout façon déjà décalé de qqs jours.

Voici un exemple d’une récupération pour une journée partielle:



Et voici un exemple de données “farfelus” (les 5400L de conso journalière sont inexactes).



### GRDF

Pour le moment pas compatible avec Domoticz (le “connecteur” nécessite un développement).

Les données sont souvent à jour après 17h, mais régulièrement plus tard. Pour éviter des appels API inutiles (sans nouvelles données), il semble judicieux de les programmer à partir de 21h seulement.

## “Serveurs”

### Docker - “configuration système incluse”

La mise en place le plus rapide est à priori avec Docker. Cela peut vous aider à mettre au point votre fichier de configuration sans que cela soit un passage obligé.

Il vous faudra environ 500Mo en sus de l’installation de [Docker](https://docker.com).

Vous pouvez vous passez de Docker et économiser des ressources en vous appuyant sur un serveur Linux que vous utilisez par ailleurs, ou encore votre système Domotique (tel que AppDaemon avec Home Assistant).

En résumé, les fichier suivants donnent la configuration de Docker:

* docker-compose.yml : plusieurs configurations de conteneurs (environnements d’exécution) fonctionnelles, dont des configurations pour le débogue.
* Dockerfile\*: Fichiers définissant “l’installation” de conteneurs.

A cela vous devez “juste” ajouter votre fichier de configuration “config.json”.

Et puis vous exécutez l’une de ces commandes:

docker compose run --rm meters-to-ha-veolia  
docker compose run --rm meters-to-ha-grdf

Ou en mode débogue (nécessite un serveur X local):

docker compose run --rm meters-to-ha-debug-veolia  
docker compose run --rm meters-to-ha-debug-grdf

Docker a parfois [un bogue (?)](https://github.com/docker/for-win/issues/13398) concernant host.docker.internal utilisé comme adresse pour le hôte sur lequel tourne docker. Dans ce cas, vous pouvez fournir la configuration DISPLAY avec l’IP de votre PC directement avec -eDISPLAY=:

docker compose run -eDISPLAY=10.33.2.69:0.0 --rm meters-to-ha-debug-grdf

L’automatisation de l’exécution avec Docker dépendre de votre système - le conteneur Docker ne tourne pas en tache de fond - il n’est pas prévu pour automatiser la tache par lui-même.

Sous Windows vous pourrez utiliser l’outil “Planificateur de tâches”.  
Sous Linux, vous utiliserez cron (crontab).

Pour mettre à jour une image docker, il faut recourir à ‘build’. Exemple:

docker compose build meters-to-ha-veolia

### Installation “direct”

De façon générale, le “serveur” nécessite l’installation des logiciels et bibliothèques prérequis:

* Navigateur web + bibliothèque d’interface de contrôle:
  + firefox+geckodriver, ou,
  + chromium+chromium-driver
* xvfb : Framebuffer (virtuel)
* xephyr : Serveur X imbriqué (recommandé)
* python3 : Interpréteur de scripts “Python”
* Modules python3 (à installer) :
  + selenium
  + pyvirtualdisplay
  + colorama
  + urllib3
  + requests
  + paho-mqtt (en cas d’utilisation de MQTT)

Les fichiers Dockerfile (Ubuntu 22.04), DockerfileDebian (Debian bullseye), et DockerfileAlpine (Alpine 3.17) peuvent vous aider pour trouver les commandes d’installation.

Les modules python3 sont disponibles pour la plupart comme paquet système, sinon vous pourrez aussi les installer avec pip (avec le requirements.txt fournit dans ce dépôt):

python3 -m pip3 install -r requirements.txt

### Installation avec le sous-système Windows pour Linux (WSL)

L’installation dans le sous-système Windows pour Linux (WSL) devrait être également possible. Et à priori on peut même y configurer des tâches avec cron. Toutefois, l’évolution d’une installation système WSL à une autre peut nécessiter de tout réinstaller - pensez à gardez une copie de votre configuration et un script d’installation des outils.

Toutefois cette méthode n’a pas été testée.

### Installation native sous Windows

Pour une solution n’utilisant aucune forme de Linux sous Windows (ni WSL, ni Docker, ni une VM), vous devez installer Python sous Windows et les modules nécessaires. La version testée est [3.10.9 obtenu depuis https://www.python.org/downloads/windows/](https://www.python.org/downloads/windows/).

Vous aurez également besoin de [ChromeDriver](https://chromedriver.chromium.org/downloads) pour Windows.

Vous pouvez extraire le fichier ChromeDriver.exe qui correspond à votre version Chrome directement dans le dépôt que vous avez récupéré.

Une autre méthode qui semble fonctionne c’est de laiser faire undetected-chromedriver à installer avec

pip install undetected-chromedriver

undetected-chromedriver est un module qui se charge de récuperer le binaire et de le patcher pour que la navigation soit moins détectable comme une navigation géré par un automatisme. Cela semble avoir une certaine efficacité pour la validation simple d’un captcha. Il faudra probablement ajouter l’option --chrome-version au lancement de meterstoha.py (voir <METERSTOHA.BAT> pour un exemple d’automatisation).

Ensuite vous devez [installer MetersToHA](#installation-de-meterstoha).

Pour le lancement sous Windows, vous devez renseigner les chemins de chrome et chromedriver (absolu ou relatifs depuis là ou vous lancez le script).  
Vous pouvez omettre chromedriver si vous avez installé le module undetected-chromedriver.

Un exemple d’un fichier de configuration est:

{  
 "veolia\_login": "monm\u00e9l@mon.domaine",  
 "veolia\_password": "mot de passe",  
 "veolia\_contract": "5453325",  
 "grdf\_login": "monm\u00e9l@mon.domaine",  
 "grdf\_password": "mot de passe",  
 "grdf\_pce": "21546000000000",  
 "ha\_server": "http://homeassistant.local:8123",  
 "ha\_token": "XXXXXXXXXXXXXXXXXXXXXX",  
 "chromium": "c:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe",  
 "chromedriver": ".\\chromedriver.exe",  
 "type": "ha",  
 "timeout": "30"  
}

Sans l’option --debug le déroulement sera visible. Dans ce cas s’il y a un captcha à résoudre, le login ne sera pas réussi. L’option --screenshot permet d’enregister le ontenu du navigateur (caché) dans screen\_before\_connection.png ou vous pouvez vérifier s’il y avait un captcha ou pas.

Sinon, pour la résolution du captcha sous Windows en mode interactif, l’option --debug est nécessaire. Lorsque le popup apparaiti, vous  
devez le captcha et le valider. Cliquez le bouton “Connexion” également. Si le captcha est validé automatiquement, le script avance automatiquement comme en mode caché et vous n’aurez pas à résoudre de captcha.

Exemple de lancement, avec un fichier de configuration appelée winconfig.json, avec une capture d’écran avant connexion.

python apps/meters\_to\_ha/meters\_to\_ha.py -c winconfig.json --grdf -r --debug --screenshot

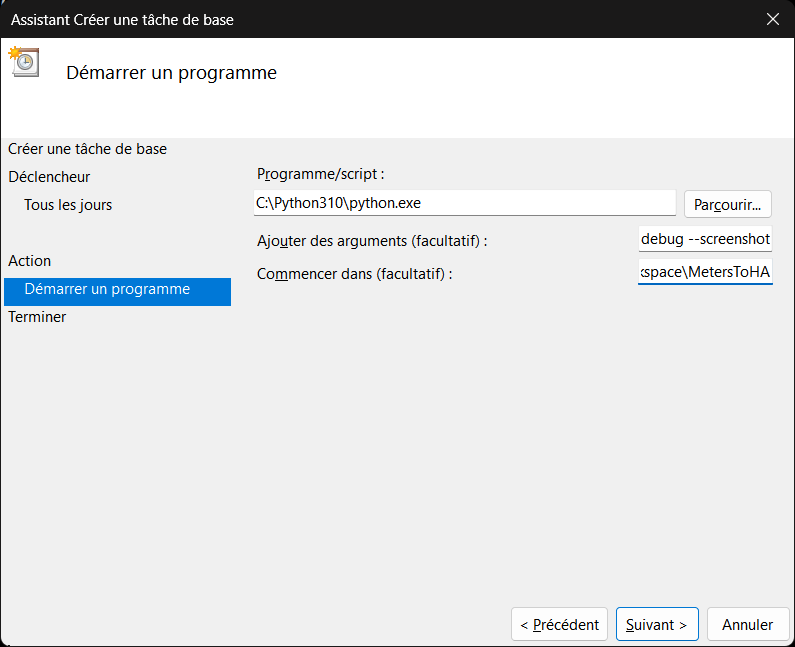
Le fichier <METERSTOHA.BAT> montre plus de variantes sur le lancement (interactif ou pas, avec trace ou pas).

Pour plus d’information concernant le contenu du fichier de configuration, vérifiez [le paragraphe correspondant](#le-fichier-de-configuration-configjson) dans le présent document.

#### Planification de tâche

Vous pouvez planifier l’exécution de la tâche sous Windows.

Pour y parvenir, cherchez Planificateur de tâches dans la zone de recherche windows.  
Choissez “Créer une tâche de base” et suivre le processus. Choississez “Tous les jours” et “Démarrez le” à une heure qui sera celle de tous les jours. Puis “Démarrer un programme”.  
Il est ensuite important de choisir votre pythonw.exe comme programme, les paramètres et “Commencez dans” qui doit correspondre au chemin de MetersToHA (contenant le sous-répertoire apps).



Avant de “Terminer”, choisissez d’ouvrir les propriétés de la tâche afin de pouvoir activer l’exécution même si l’utilisateur n’est pas connecté. Vous pourrez y ajuster également d’autres paramètres (voir les onglets).

Trouvez la tache dans “Bibliothèque du Planificateur de tâches” et faites “Exécuter” (dans le menu accessible par clique droit de la tâche) afin de vérifier que cela fonctionne (vérifiez les fichiers de sortie).  
Quand cela ne fonctionne pas, assurez-vous que tous les chemins sont corrects.

### Installation de MetersToHA

Vous pouvez extraire les fichiers de ce dépôt ou vous voulez.

Le script apps/meters\_to\_ha/meters\_to\_ha.py et son fichier de configuration config.json suffisent (en sus des prérequis).  
Le fichier config.json.exemple peut servir comme base pour réaliser votre fichier de configuration.

En utilisant git, vous facilitez la mise à jour, sinon téléchargez l’archive.

#### Installation avec git

Récupération initiale:

cd REPERTOIRE\_DE\_DESTIONATION  
git clone https://github.com/mdeweerd/MetersToHA  
cd MetersToHA  
  
pip3 install -r requirements.txt  
# ou  
python -m pip install -r requirements.txt

Mise à jour:

git pull

#### Installation avec une archive

Extraire l’archive, puis s’assurer que le script est exécutable (\*nix):

chmod ugo+x apps/meters\_to\_ha/meters\_to\_ha.py

### Automatisation :

Une automatisation permettra de lancer la récupération une fois par jour. Il est bien sûr préférable de d’abord valider le fonctionnement sans automatisation.

Vous pouvez faire cela avec cron et un de ses fichiers de configuration crontab. Pour cela, ajoutez la ligne suivante à votre planificateur de tâches :

./apps/meters\_to\_ha/meters\_to\_ha.py --run

Exemple ici avec crontab que l’on peut éditer avec :

crontab -e

Pour y ajouter le contenu qui suivent tout en:

* Modifiant les chemins selon votre installation;
* Garder que les lignes utiles (Veolia et/ou GRDF)

SHELL=/bin/bash  
M2HA\_PATH=/opt/MetersToHA/apps/meters\_to\_ha/  
M2HA\_SCRIPT=${M2HA\_PATH}apps/meters\_to\_ha/meters\_to\_ha.py  
M2HA\_CONFIG=${M2HA\_PATH}config.json  
M2HA\_LOG=${M2HA\_PATH}meters\_to\_ha.log  
# Veolia  
0 1 \* \* \* sleep ${RANDOM:0:2}m && ${M2HA\_SCRIPT} --veolia -c ${M2HA\_CONFIG} -log ${M2HA\_LOG}.veolia  
# GRDF  
0 20 \* \* \* sleep ${RANDOM:0:2}m && ${M2HA\_SCRIPT} --grdf -c ${M2HA\_CONFIG} -log ${M2HA\_LOG}.grdf

## Dépannage

### Première exécution :

Par default le script est muet (il n’affiche rien sur la console et ne lance pas la version graphique de Firefox). Il enregistre toutes les actions dans le fichier INSTALL\_DIR/veolia.log. Je vous recommande pour la première utilisation d’activer le mode débogue. Cela permet d’avoir une sortie visuelle de l’exécution du script sur la console et un suivi des actions dans Firefox.

Déroulement de l’exécution :

1. Chargement de tous les modules python –> si erreur installer les modules manquants (pip3 install …)
2. “Sanity check” de l’environnement :

* Version
* Prérequis logiciel externe –> si erreur installer le logiciel manquant
* Configuration Domoticz –> si erreur configurer correctement Domoticz

1. Connection au site Veolia et téléchargement de l’historique
2. Téléversement des données dans Domoticz

./apps/meters\_to\_ha/meters\_to\_ha.py --run --keep-output --debug

Afficher toutes les options disponibles :

./apps/meters\_to\_ha/meters\_to\_ha.py --help

Lorque la récupération des données a déjà fonctionné (avec --keep-output), vous pouvez pendant les essais réutiliser le fichier déjà téléchargé avec l’option --skip-download. Cela permettra la mise au point la configuration de l’envoi des données vers votre système domotique.

Exemple:

./apps/meters\_to\_ha/meters\_to\_ha.py --grdf -r --keep-output --skip-download

### Paramétrer votre système pour le mode débogue (optionnel, mais recommandé)

Si vous rencontrez des problèmes à l’exécution, regardez dans un premier temps le fichier “veolia.log”.

Si cela ne suffit pas, pour aller plus loin il sera utile d’utiliser le mode débogue (option --debug).

Dans ce dernier cas il y a 3 scenarios :

1. Le script est exécuté en locale par l’utilisateur avec lequel vous êtes logués ==> ca devrait fonctionner tout seul, mais vous devez utiliser une machine de type “Linux” avec interface graphique ;
2. Vous exécutez le script sur une machine distante Linux. Il convient alors de vérifier que la commande suivante fonctionne après être connecté sur la machine Linux distante (via ssh probablement) : xlogo;
3. Vous êtes sous Windows, vous pouvez par exemple utiliser la solution [Docker](https://www.docker.com/) un serveur X (p.e. [VcXsvr](https://sourceforge.net/projects/vcxsrv/)) et le lancer (!) avec l’option “Disable Access Control”, puis lancer l’un des scripts docker\\*Run.BAT après avoir ajouté l’option ‘–debug’ à la ligne de lancement du script.

Si vous voyez bien une fenêtre X s’afficher à l’écran c’est que l’environnement X11 est correctement configuré. Le mode débogue du script devrait fonctionner.

Si par contre rien ne s’affiche, il convient de chercher sur internet comment le faire fonctionner, il y a pleins de tutos pour cela. Ensuite vous pourrez utiliser le mode débogue.

## Principe de fonctionnement

L’outil simule la visite du site a grâce à l’outil selenium.  
Il procède alors aux étapes d’identification, parcourt les pages autant que nécessaire, et télécharge un fichier d’historique adéquat.  
Ce fichier est alors décortiqué pour en extraire les informations utiles.  
Ces données sont ensuite envoyés au système domotique choisi à travers son API.

Selenium exécute un navigateur Firefox ou Chromium en mode “Headless”.

Le mode Headless indique que le système n’a pas d’écran.

Le système Graphique (GUI) existe, mais l’affichage n’existe que dans une zone mémoire.

Il est néanmoins possible de voir le déroulement en temps réel avec l’option --debug. L’affichage n’est alors plus “Headless” et il vous faudra un serveur X attaché à un écran physique.

## Environnements testés:

Actuels (à travers Docker):

* Debian Bullseye - Chromium - (Docker 1.2GB)
* Alpine 3.17 - Chromium - (Docker 540MB)
* Ubuntu 22.04 - Firefox - (Docker 1.2GB)
* Home Assistant/AppDaemon - Alpine Linux v3.17 - Chromium

Anciens (plus testés avec les versions récentes):

* Debian Buster - Chromium
* Ubuntu 20.04 - Firefox
* Ubuntu 21.04 - Firefox

A noter qu’Ubuntu supporte probablement aussi la solution avec Chromium.

## Cas de figure:

* #9 NoSuchElementException pour find\_element(By.TAG\_NAME, "pre"). Le processus pense avoir attînt l’étape du résultat.
  + Vérifiez que la résulution du captcha se fait correctement. Cette erreur a été constaté lorsque cet autre message apparaissait: capmonster status 402{"errorId":1,"errorCode":"ERROR\_ZERO\_BALANCE","errorDescription":"Account has zero balance"}

## Remerciements/Contributeurs

* [s0nik42](https://github.com/s0nik42)
* [k20human](https://github.com/k20human)
* [guillaumezin](https://github.com/guillaumezin)
* [mdeweerd](https://github.com/mdeweerd)