**ALTUI : Evolutions récentes et les machines à état**

Altui, alexis mermet, avril 2016

**AltUI** est une user interface, alternative à celle fournie par Micasaverde, le fabriquant de la box domotique VERA. ALTUI est très simple d’utilisation puisqu’il s’agit d’un plugin VERA classique s’installant via l’App Store de Micasaverde. ALTUI apporte son lot d’améliorations ainsi que de nouvelles fonctionnalités non présentes sur la box de base. Pour mémoire ALTUI tourne sur n’importe quel périphérique client (PC MAC Tablette Phone ) et s’adapte automatiquement au format, permet d’afficher presque tous les plugins disponibles sur VERA, et ce même en mode accès à distance par internet.

Cette application a déjà été présentée dans les précédents articles

1. Introduction
   * <http://www.domotique-info.fr/2015/04/altui-pour-vera-une-alternative-ui5-ou-ui7/>
2. Les Pages Custom :
   * <http://www.touteladomotique.com/index.php?option=com_content&view=article&id=1444:altui-user-interface-alternative-pour-vera-custom-pages&catid=5:domotique&Itemid=89>
3. L’Extensibilité / les API:
   * <http://www.touteladomotique.com/index.php?option=com_content&view=article&id=1445:altuin-user-interface-alternative-pour-vera-extensions&catid=5:domotique&Itemid=89>
4. Les évolutions depuis Avril 2015 :
   * <http://www.touteladomotique.com/index.php?option=com_content&view=article&id=1579>
5. L’objet de cet article, le cinquième de la série, est de montrer certaines nouveautés et un nouveau concept : **les Machines à état**, ou « workflow » en anglais. L’objet de cet article est donc de passer en revue les évolutions depuis le dernier article et fournit ensuite un tutoriel sur cette vraie nouveauté. Il est donc conseillé de relire les articles précédents si vous n’êtes pas familier avec ALTUI

# Le cout

Parlons tout d’abord du cout. ALTUI reste en open source pour les utilisations personnelles et le code source est disponible, mais la philosophie est passé sur un système de donation annuelle. Un message vous invite à faire une contribution via paypal pour une année d’utilisation. Une fois la contribution enregistrée, il n’y a plus de message de départ et le bouton paypal disparaitra.

Le « Look and Feel » de ALTUI est très configurable grâce à certaines API internes, plutôt réservées à l’utilisation commerciale comme par exemple un intégrateur de solution domotique qui voudraient l’utiliser pour leur clients. Si une société est intéressée par une utilisation dans un tel cadre commercial contacter moi et nous pourrons formaliser un accord commercial.

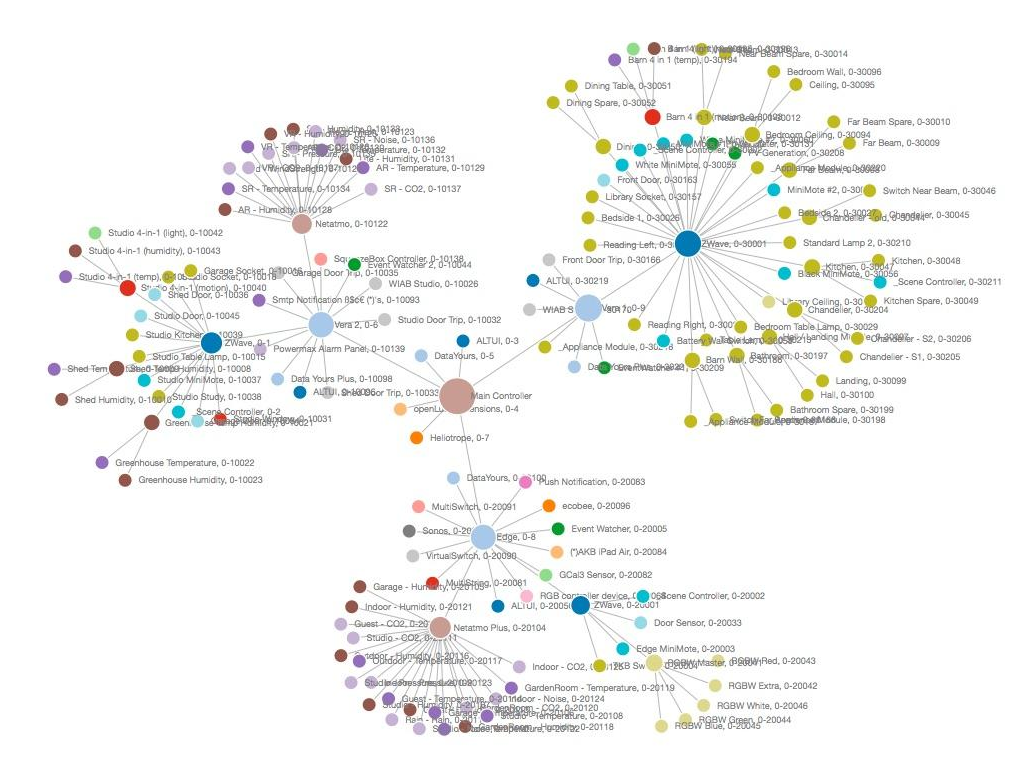
Dans un premier temps passons en revue les évolutions « naturelles » récentes  des fonctionnalités existantes.

# Evolution Globale

## OpenLuup

Un membre actif du forum MCV ( forum.micasaverde.com / akbooer ) a implémenté une version open source du moteur Luup de la VERA et permet ainsi de tourner sur une grande variété de système linux ( PI ou autres ) Luup ainsi que la plupart des plugins pour la VERA.

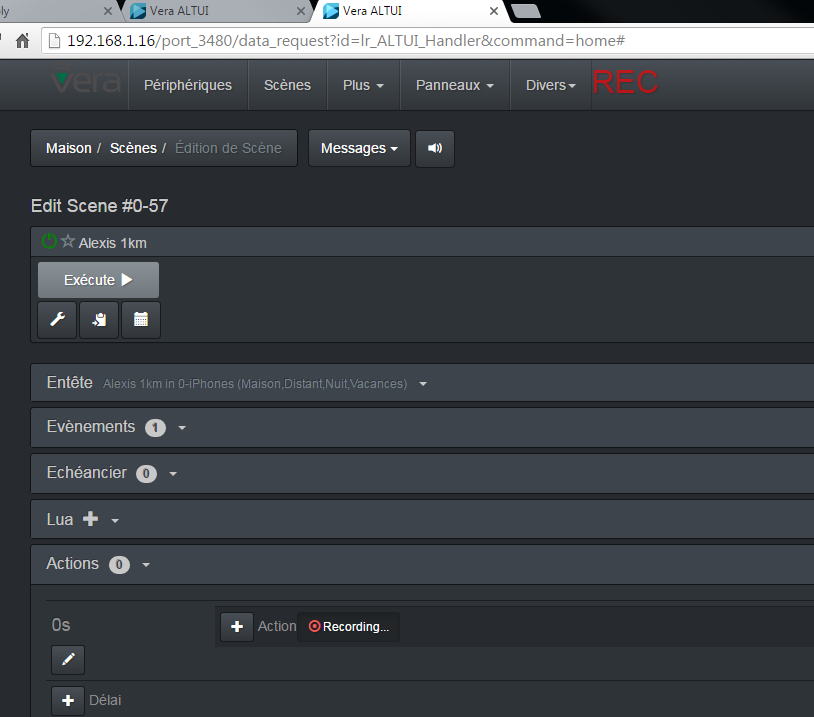
ALTUI est la seule UI officielle pour openLuup . la combinaison des 2 composants ( openLuup et ALTUI ) permet d’atteindre une échelle bien au-delà des standards actuel de la VERA et de beaucoup de « box » domotiques. Voici un exemple de diagramme réseau généré à partir de openLuup , 2 vera secondaires et ALTUI



## Recorder for Scene or Workflow editor

Lors de la création de scène ( ou de workflow, ce que l’on verra plus loin ) il est possible de choisir n’importe quel périphérique et des actions associées. Il y a une nouvelle possibilité d’enregistrer une séquence d’action en mode ‘camcorder’ plutôt que de choisir à la main des menus les actions sur les devises.

Dans ce mode, un signal « REC » clignote en haut a droite de l’écran et les actions sur les devices ou l’exécution des scènes n’est pas déclenchée en réalité mais enregistrée. Lors de l’arrêt de l’enregistrement, dans l’éditeur de scène ou de workflow, la séquence des actions enregistrées vient remplir d’un coup la liste des actions dans l’éditeur.

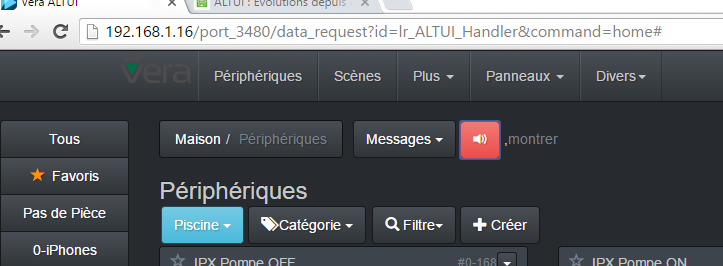


## Reconnaissance vocale

Sur Chrome (PC) il est possible de dicter certaines commandes vocales à ALTUI. Malheureusement ce support n’existe que sur Chrome PC, si votre environnement supporte la fonctionnalité, une icône haut-parleur apparait à côté du bouton « Messages ». il faudrait une intégration avec HomeKit sur IOS ou équivalent sur Android mais c’est peut-être pour le futur

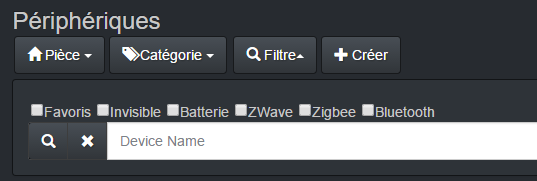
Il est possible de dicter une ou plusieurs commandes d’affilée, les commandes (configurables dans chaque langue) aujourd’hui possibles sont :

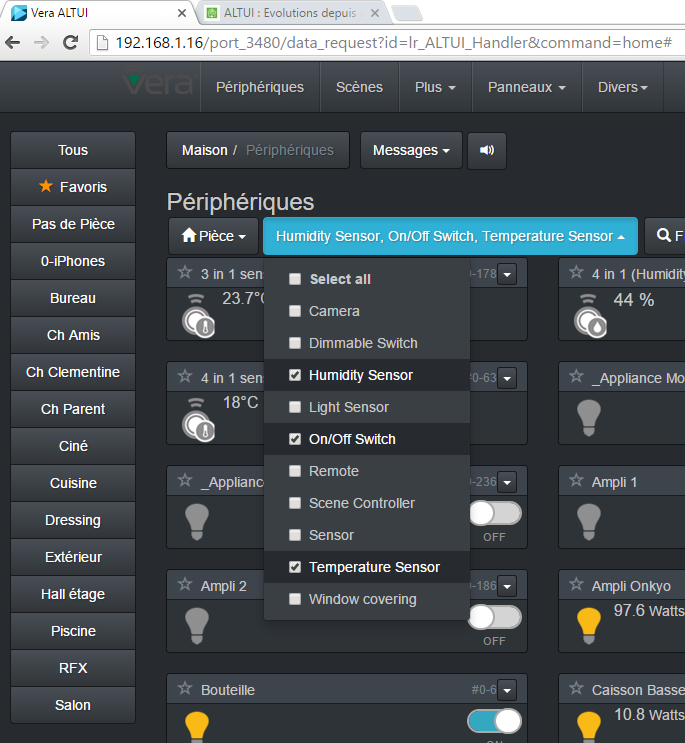
* allumer|allume|ouvrir|ouvre <nom de device>
* éteindre|éteins|fermer|ferme <nom de device>
* exécuter|exécute|lance|lancer <nom de scene>
* montre|montrer|ouvre|ouvrir <nom de piece>
* montre|montrer|ouvre|ouvrir <nom de page>



# Page Périphériques

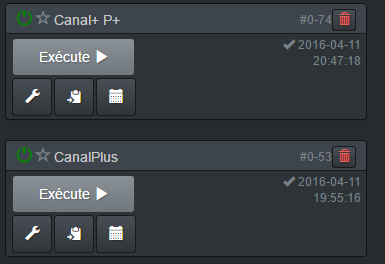
* Les filtres zWave, Zigbee, Bluetooth ont été rajoutes suite à l’introduction de la nouvelle vera qui supporte ces technologies



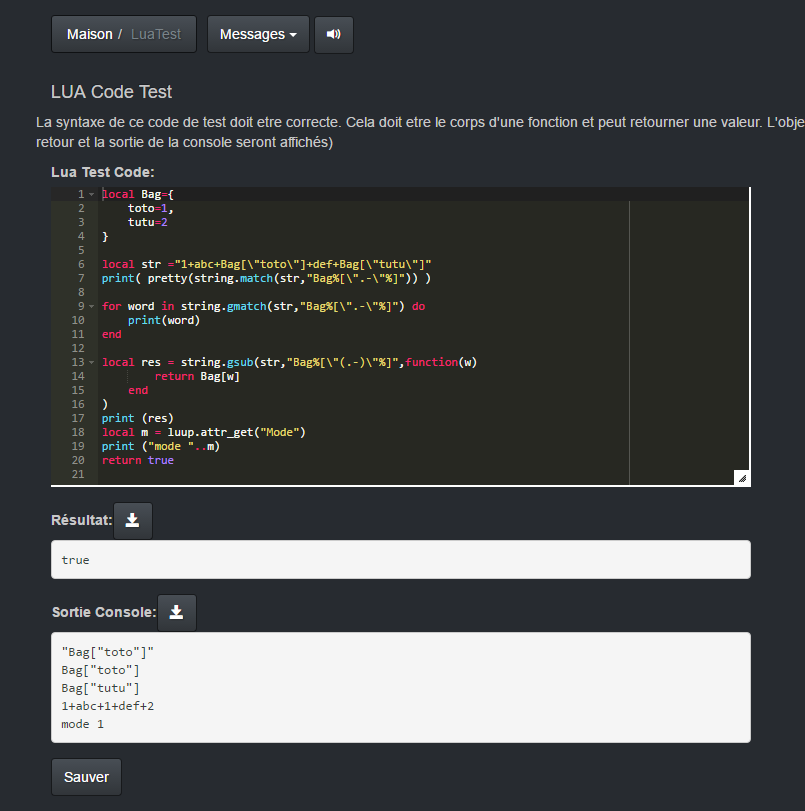
Les filtres de pièce ou de catégorie permettent la multi sélection :

# Page Scènes

On retrouve toujours l’éditeur de scène et le bouton historique pour voir les horaires des dernières exécutions de la scène. A cela s’ajoute maintenant, la capacité de copier une scène afin d’un créer un clone modifiable par la suite

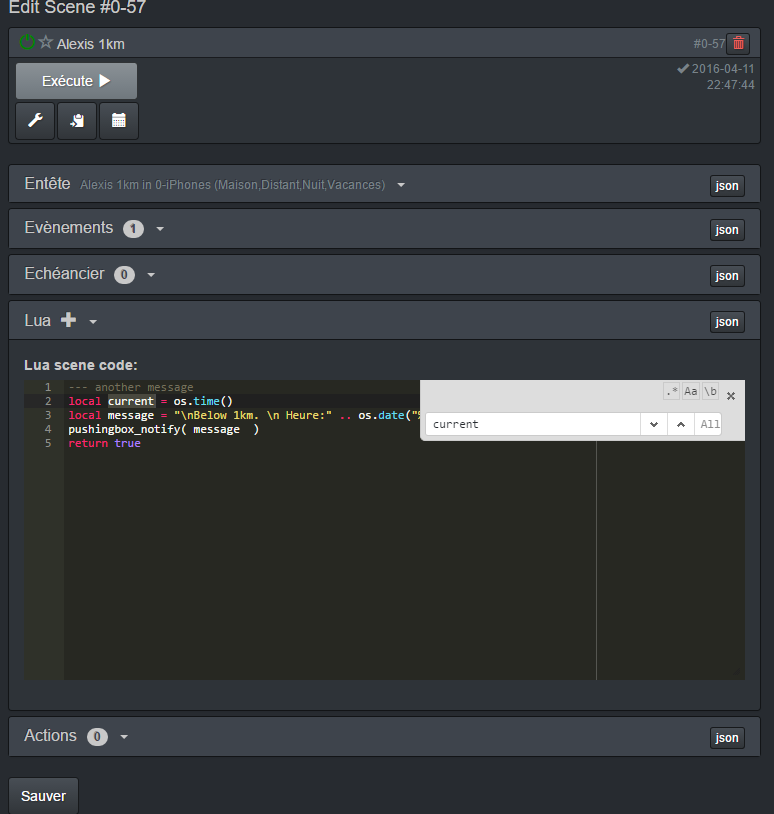


L’éditeur de code (lua, json, javascript ) devient intelligent et offre la coloration syntaxique, ainsi que le contrôle de la grammaire du langage utilisé aux divers endroits de l’application comme la fenêtre de test LUA qui permet de tester un code LUA.

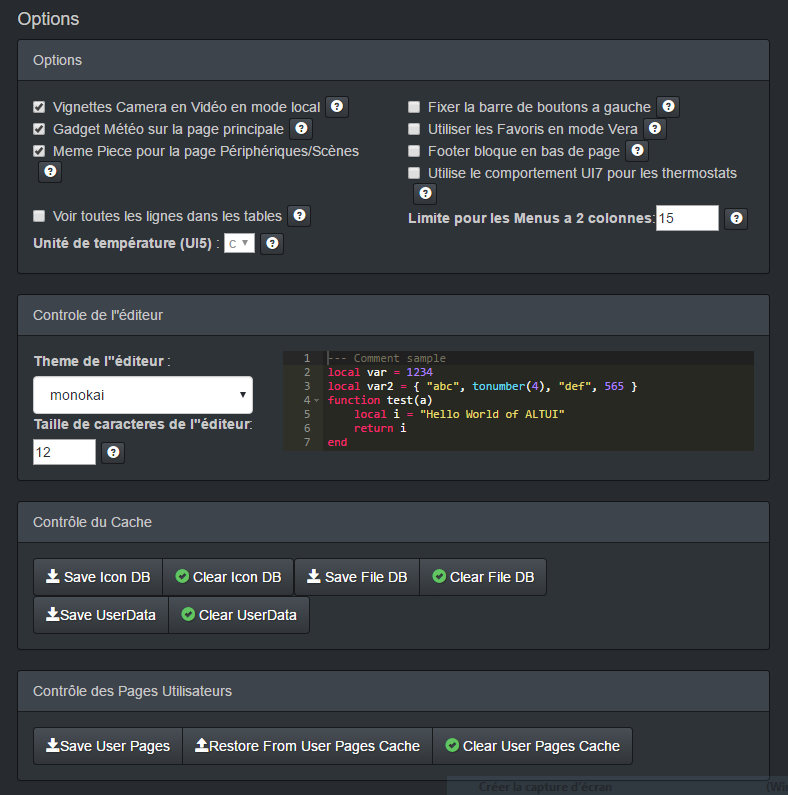


Les raccourcis claviers sont possible ( ctrl c , ctrl v , ctrl f pour ‘chercher’ ). La zone de l’éditeur est aussi redimensionnable afin de bénéficier d’un éditeur de grande taille si nécessaire

L’Editeur de Scène :

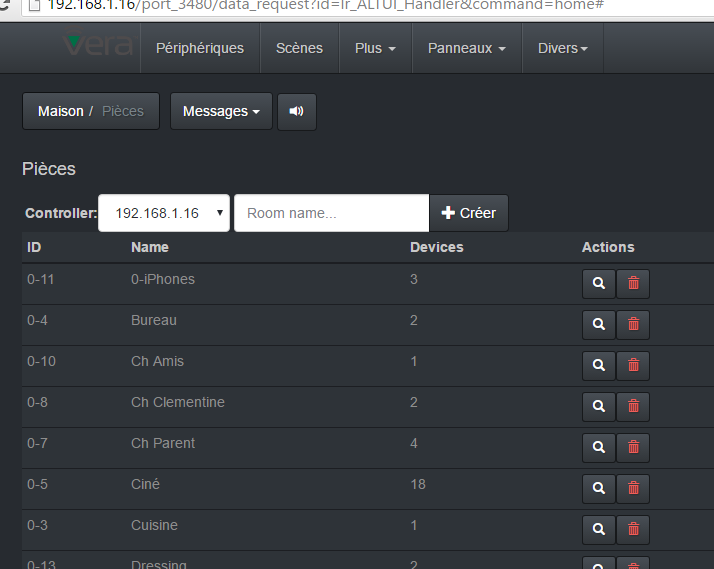


Il est bien sur possible de changer les thèmes de l’éditeur à partir du panneau des options



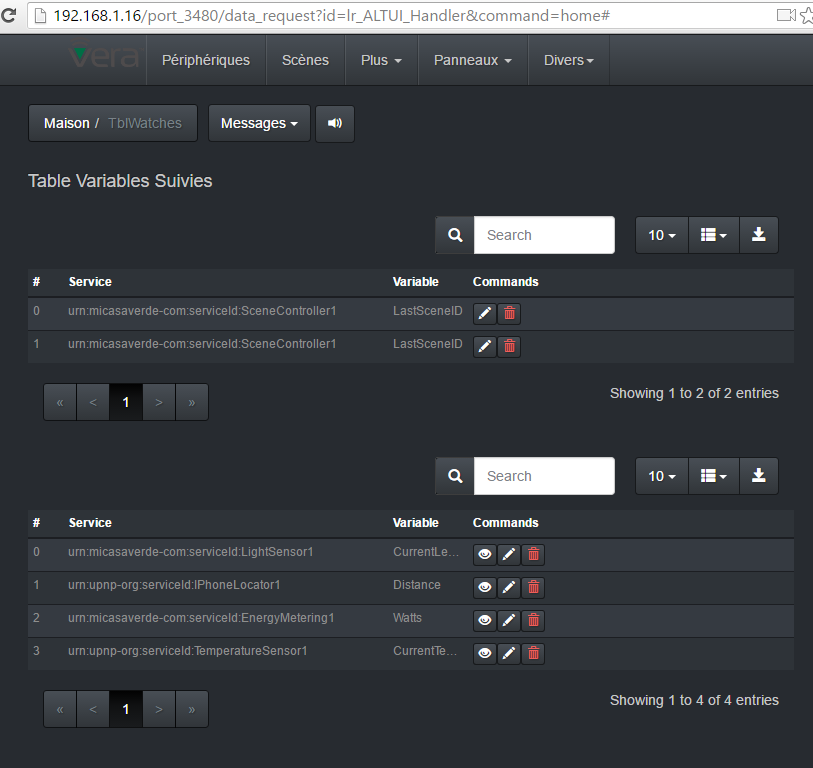
# Page Pièces

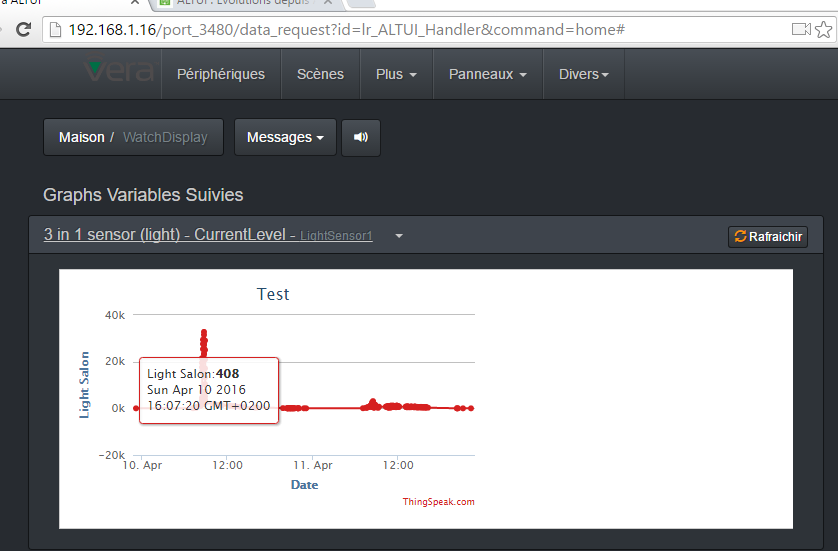
La page Pièce s’enrichie d’un bouton de recherche qui permet d’afficher directement les périphériques d’une pièce donnée.



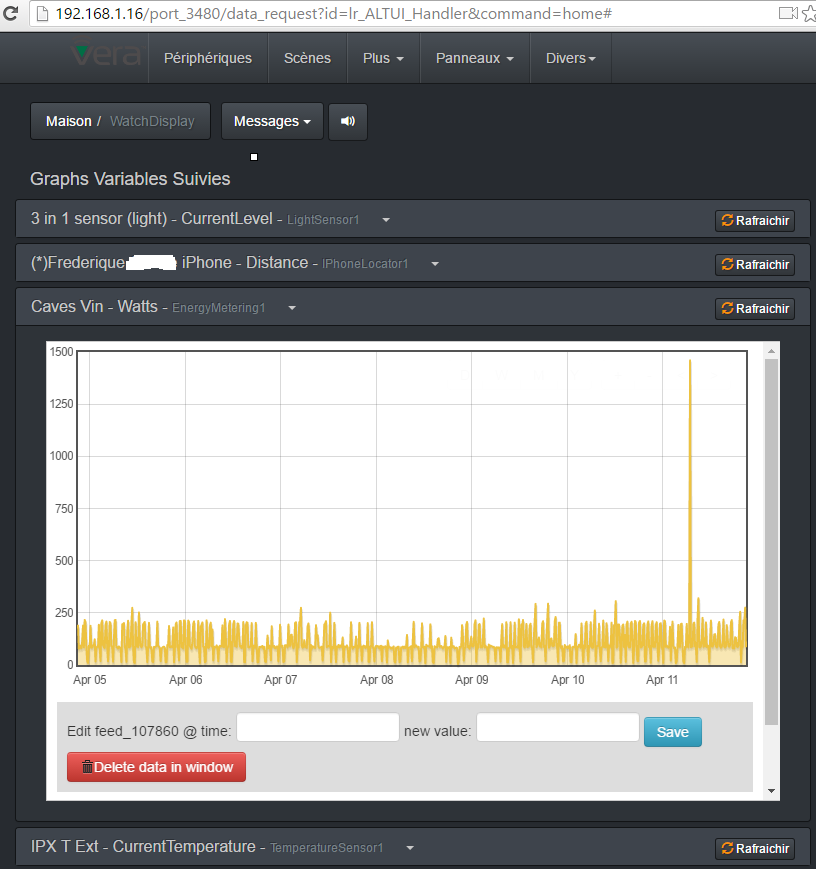
# Page Variable Suivies

Cette page permet de voir l’ensemble des variables suivies ( l’équivalent des Luup Watch en lua) programmées, soit pour l’exécution des scènes, soit pour les envoyer dans un service de stockage sur internet / « nuage » comme thingspeak ou emonCMS.

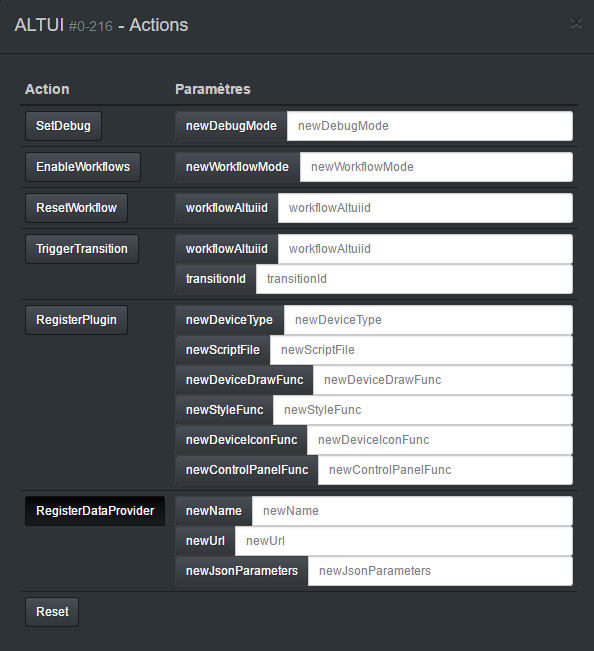




Tous les graphiques sont visualisables un par un ou bien sont regroupés sur une page qui montre tous les graphiques programmes dans un accordéon.



Il existe un moyen d’enregistrer dynamiquement dans ALTUI d’autres fournisseurs de stockage de données que les 2 fournis par défaut : thingspeak et emoncms. Par exemple le plugin Datamine2 de akbooer (sur le forum MCV) s’enregistre auprès de ALTUI et devient automatiquement disponible dans la user interface de ALTUI au moyen de l’action UPNP **RegisterDataProvider()** pour programmer le suivi de la variable et le graphique historique des valeurs. ALTUI construira une user interface dynamiquement, capturant les paramètres nécessaires pour votre data provider et s’occupera de remonter à ce service, les valeurs des variables suivies au fil du temps , ainsi que de récupérer les graphiques

Les actions UPNP de ALTUI :

# Pages Custom

Les pages custom gagnent aussi quelques améliorations et certain nouveaux widgets.

## Espacement

On trouve les nouveaux widget pour l’espacement automatique verticalement ou horizontalement des widgets :



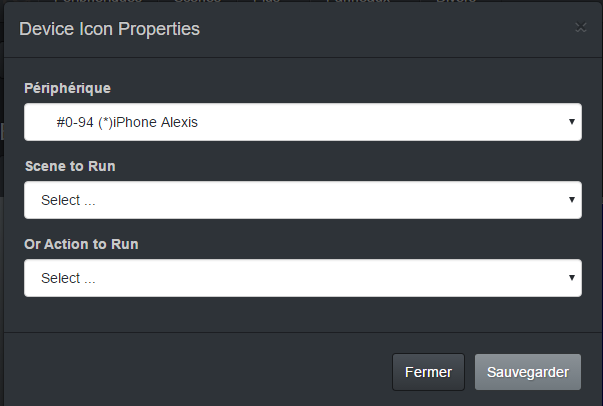
## Curseurs

On trouve aussi des nouveau widgets comme les curseurs verticaux ou horizontaux pour Controller les périphériques de type variateur ou volets.

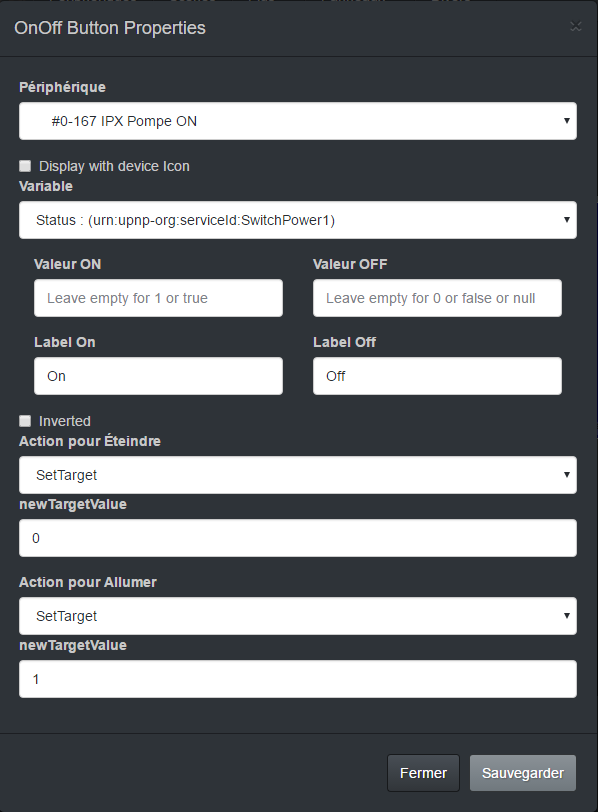
|  |  |
| --- | --- |
|  |  |

## Ameliorations de widget existants

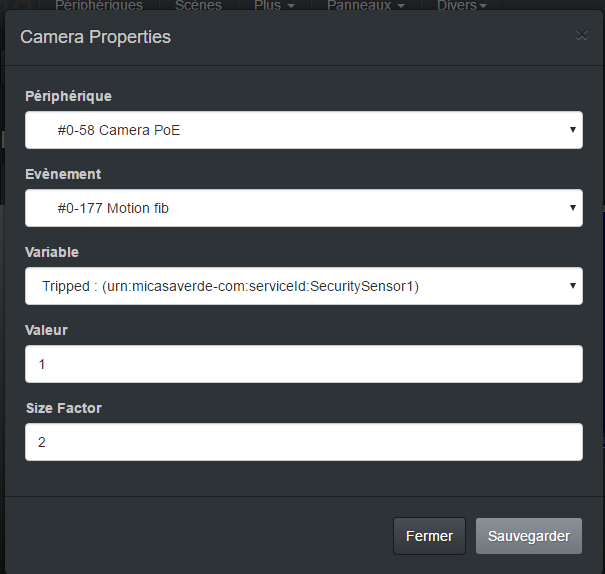
Les widgets Icones gagne la possibilite de lancer une scene ou une action d’un device



Les boutons 2States ON/OFF  gagnent quelques paramètres supplémentaires pour configurer les valeurs correspondantes aux états ON ou OFF des boutons car certains périphériques utilisent des valeurs différentes de 0 ou 1

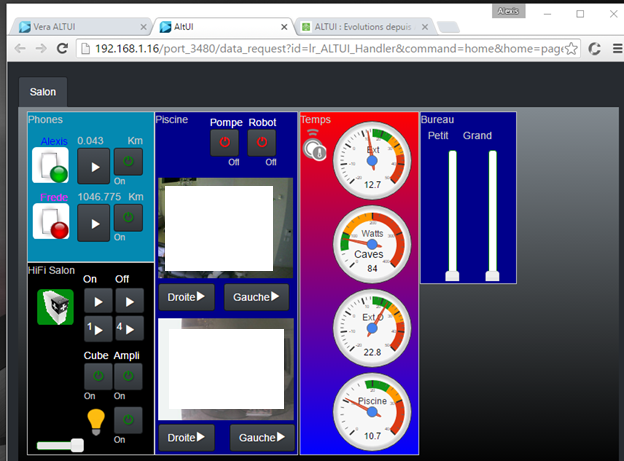


Les widgets Cameras possèdent un mode « Alerte » qui affiche la camera en zoom ( en plus gros ) si une variable (comme l’état d’un détecteur de mouvement) prend une certaine valeur



## Mode Panneau Simple

L’url d’ouverture de la page principale peut être customisée avec une option « lean » pour n’afficher que les pages utilisateurs sans aucunes fioriture/menus. Utile pour éviter que le petit dernier ne modifie les panneaux de contrôle.



# Les Machines à états

Mais le gros changement c’est le concept de machines à états que nous allons détailler maintenant.

La VERA est un système relativement ouvert, mais il faut bien reconnaitre qu’il est nécessaire de recourir à la programmation en LUA très souvent des que l’on veut commencer à faire des scenarios un peu compliques.

Pour comprendre, prenons quelques exemples :

* Allumer une lampe automatiquement le soir et lorsque qu’un mouvement est détecté par un détecteur, puis éteindre la lampe automatiquement. Par contre si la lampe était déjà allumée au moment du mouvement, on ne veut pas éteindre automatiquement cette lampe
* Détecter qu’une porte est ouverte, le soir, et envoyer un SMS/email de rappel toutes les 15 minutes tant qu’elle n’est pas fermée

En fait des que l’on veut prendre des décisions dépendantes de ce qu’il s’est passé auparavant, il est très compliqué de gérer cela comme il faut sans se lancer dans le code LUA et avec des plugins additionnels. De plus, la VERA relance son moteur Luup de manière imprévisible (pour faire une rotation des logs, ou récupérer de la mémoire) et les timers en attentes programmés par LUA sont perdus. Ce qui rend la tâche vraiment compliquée pour l’utilisateur

Alors bien sûr il reste l’option des plugins complexes comme PLEG, mais il n’y a pas vraiment d’user interface qui permette de facilement éditer ses conditions et de voir en temps réel ce qu’il se passe

Pour pouvoir garder une mémoire des états, beaucoup d’utilisateurs de vera utilisent des plugins comme le « variable container. Mais c’est assez lourd d’utilisation et nécessite un autre plugin et donc des ressources sur la VERA qui sont assez « précieuses »

Les machines à états de ALTUI apportent une réponse relativement simple, facile à comprendre, qui offre la capacité de gérer des mémoires/variables et possède une interface graphique pour éditer son workflow et voir, en voir , en temps réel, l’exécution. Les timers, les variables résistent aux reload/reboot donc le résultat est fiable

## La théorie

Un peu de théorie avant de passer aux photos.

### Les états

* Les états sont des états logique de votre installation, à vous de décider ce qu’il vous faut. Il est justement clé de bien identifier ces états. Ceci va permettre de programmer une réponse différenciée par rapport à une situation. Les états sont les « mémoire » des machines à états. En fonction d’où l’on est dans le workflow (c’est-à-dire de l’état actif), par où l’on est passé, on ne va pas déclencher les mêmes réactions. Par exemple une lampe allumée a la main, ou une lampe allumée par un détecteur de mouvement sont dans 2 états différents parce que l’on ne va pas effectuer les mêmes actions dans chacune de ces situations. Nous allons éteindre la lampe auto-allumée par le mouvement après 10 minutes, mais nous voulons la laisser allumée si elle avait été allumée manuellement. On pourrait aussi envisager que sur la pression d’une télécommande, la lampe auto allumée est alors considérée comme allumée manuellement et l’on ne l’éteindra pas. Une simple transition d’était sur la pression de la touche de la télécommande suffirait à implémenter cette logique

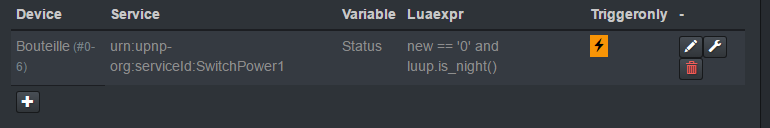
Les états permettent d’exécuter une série d’action en entrée (lorsqu’on arrive dans cet état) ou en sortie (lorsqu’on le quitte et avant d’entrer dans le nouvel état). Ces actions peuvent être

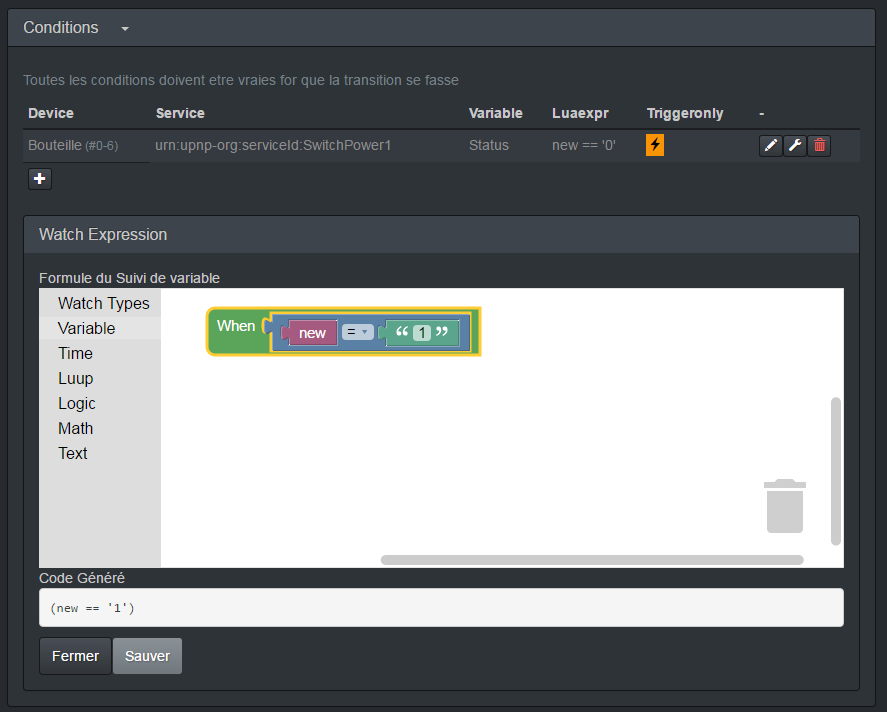
* des actions UPNP sur les périphériques,
* exécuter une scène,
* voire un petit bout de code Lua a votre convenance.

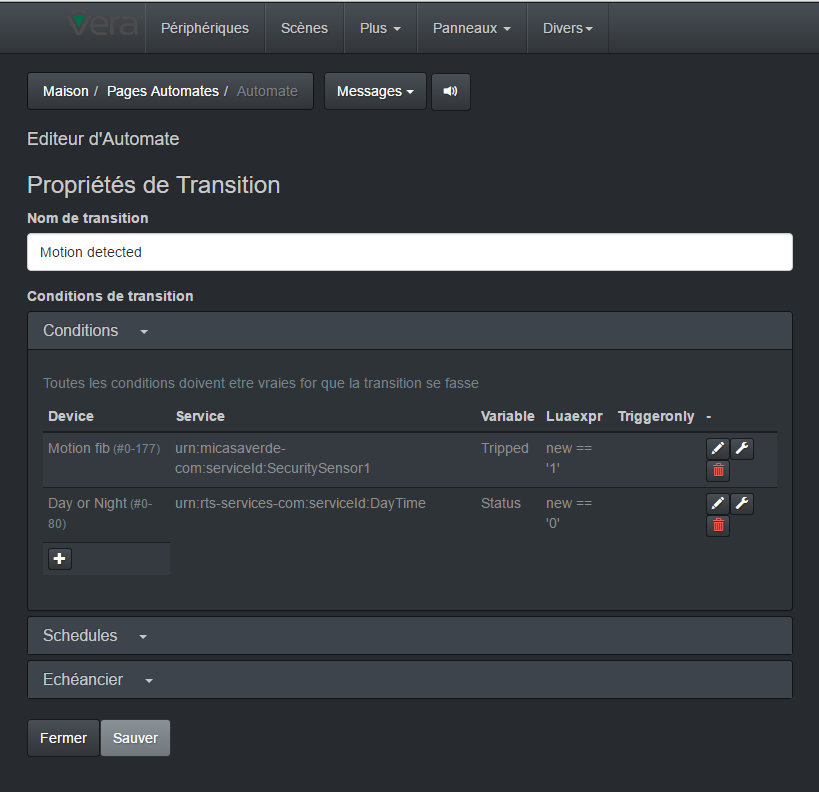
### Les transitions

* Les petites sœurs des états, ce sont les transitions qui relient les états entre eux. Lorsque le workflow est dans un état, les transitions qui partent de cet état (et seulement celles-ci) sont évaluées à chaque itération. Si l’une des transitions est « réputée » vraie, alors la transition est exécutée et l’état d’arrivée de la transition devient le nouvel état actif du workflow

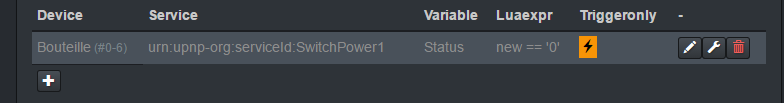
Les transitions peuvent être:

* un **Schedule programmé**, identique à celui des scènes de VERA. Toutes les mêmes options de programmation sont possibles. Au moment programmé, la transition est réputée vraie si l’état actif est bien l’état de départ de la transition
* un **timer** de x secondes. Apres avoir attendu x secondes sans que l’état actif ne change, la transition est réputée vraie
* ou bien un **ensemble de conditions**. Toutes les conditions doivent être VRAIES pour que la transition soit vraie. Par exemple ci-dessous, un mouvement doit être détecté ET de nuit. Les conditions sont des « watch » de variable de périphériques mais peuvent aussi contenir d’autres éléments, tant que le tout est une expression booléenne LUA valide. Il est possible d’utiliser l’editeur Blockly pour s’aider.  
  

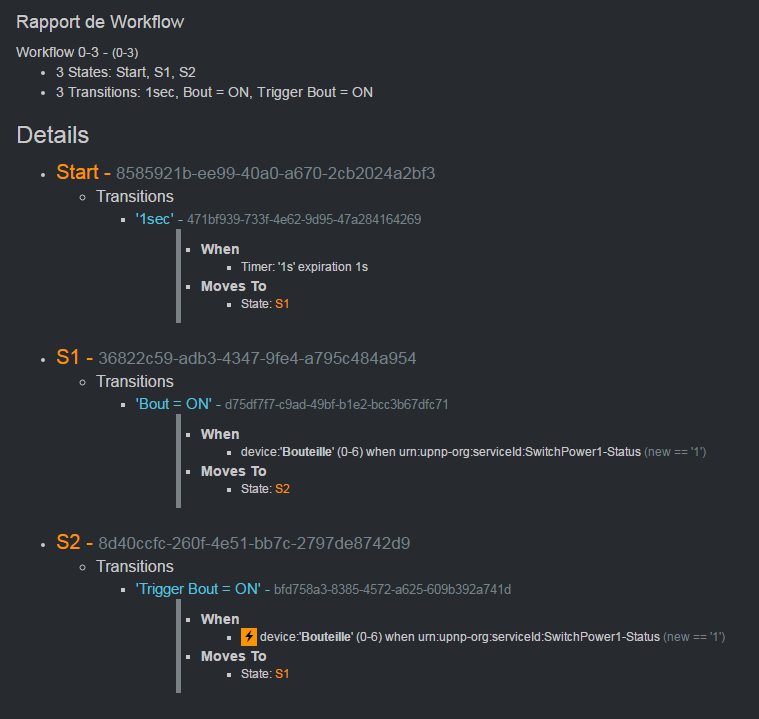




* Pour faire un « OU » entre plusieurs conditions possibles pour changer d’état, on aura recours à plusieurs transitions en parallèle
* Les conditions des transitions peuvent avoir une option « trigger-only » qui permet de ne considérer la condition comme VRAIE qu’au moment du changement de la variable et pas ensuite. (ainsi, presser un bouton de télécommande peut être considéré comme VRAI au moment de cette action et pas de manière continue pour les états suivants si aucun autre bouton n’est pressé entre temps). Cela permet donc de considérer soit la valeur (mode normal) , soit le changement de la valeur (mode Trigger-Only) comme évènement déclenchant de la transition



Voici un exemple de workflow qui ferait un flip flop permanent entre 2 états, si l’une des deux transitions n’était pas en mode « trigger only » comme l’on peut voir dans le rapport de workflow , S1 passe a S2 dès que l’interrupteur est en position allumée, alors que S2 passe a S1 si et seulement si on vient d’allumer cet interrupteur, et il faudra l’éteindre et le re-allumer pour que cette transition puisse être validée une nouvelle fois



### « Start » : Le Zombie

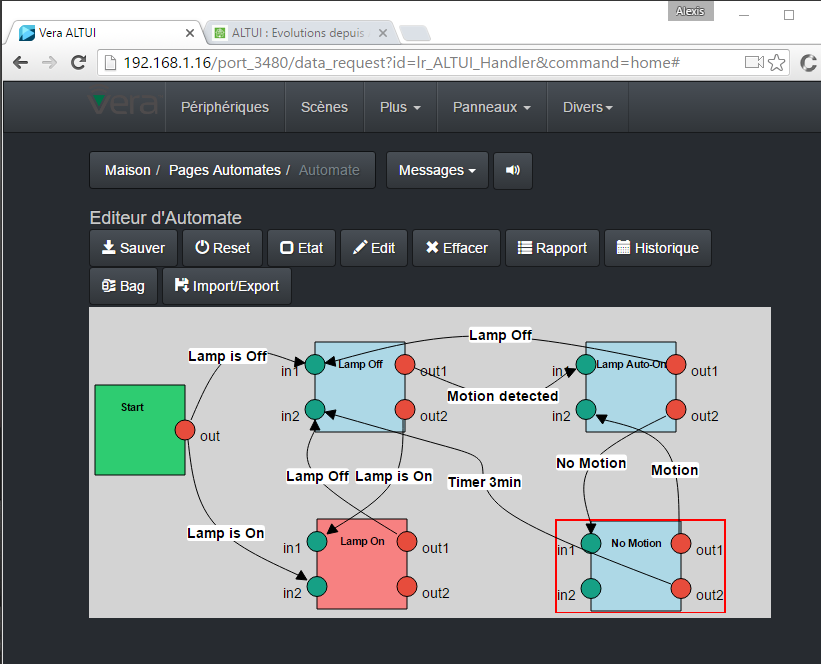
Apres un luup reload ou un reboot de la vera un workflow redémarre par default de l’état ou il était s’il existe et est valide. Il y a une action UPNP (et un bouton dans la user interface) pour forcer un « reset » du workflow.

Le zombie qui n’est ni un état ni une transition,  c’est l’état START. En fait lorsque qu’un workflow est remis à Zéro par une action « reset », tout redémarre de l’état START et c’est à vous de créer les premières transitions pour démarrer le workflow de la manière la plus logique possible.

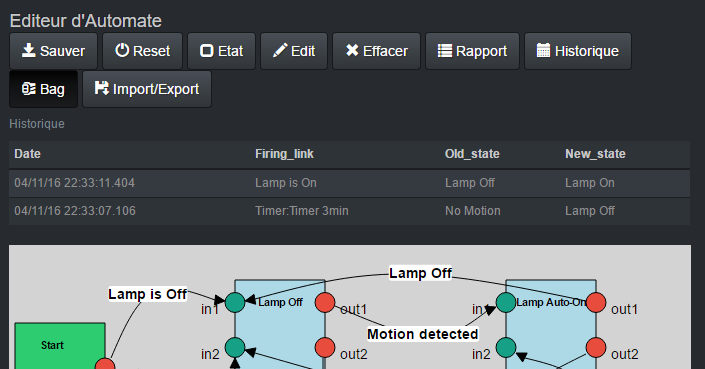
Mais l’état START possède aussi optionnellement des conditions, tout comme une transition. Ces conditions sont évaluées à **chaque changement** d’état dans le workflow et si elles sont vraies, le workflow est automatiquement redémarré à partir de l’état START, quel que soit l’état dans lequel il était. Cela peut être utile dans certains workflow spécifiques, sinon vous pouvez laisser ces conditions vides

### Tous ensemble

Voici par exemple le workflow de la lampe, on remarque les états et les transitions. Le workflow est wysiwyg et est éditable par drag and drop, les liens peuvent passer par des points fixes, ainsi on peut leur donner des formes et les labels des liens sont repositionnables.



L’état courant est rafraichi en quasi temps réel et est affiché en rouge. Il est possible de voir l’historique de l’exécution du workflow



Il existe aussi une page pour afficher une description complète d’un workflow. Ci-dessous l’exemple d’un workflow qui envoi des données sur thingspeak toutes les 15 minutes et qui compte le nombre d’envoi effectué depuis le dernier RESET du workflow.



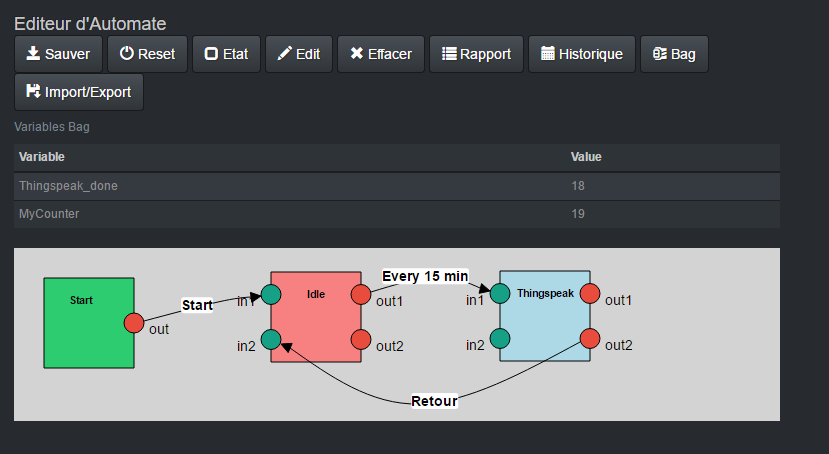
### Le Sac de variable ( Bag )

Il peut être utile pour un workflow d’avoir un jeu de variable persistantes (qui résistent au reboot/reload) sans avoir recours à un plugin externe.

Pour cela, il est possible d’utiliser la syntaxe **Bag[« nom de la variable »]** dans les zones de code Lua. La variable est créée automatiquement la première fois qu’elle est utilisée mais elle n’a pas de valeur, donc il est utile d’utiliser des expressions de type : Bag["MyCounter"] = (Bag["MyCounter"] or 0)+1 pour être sûr.

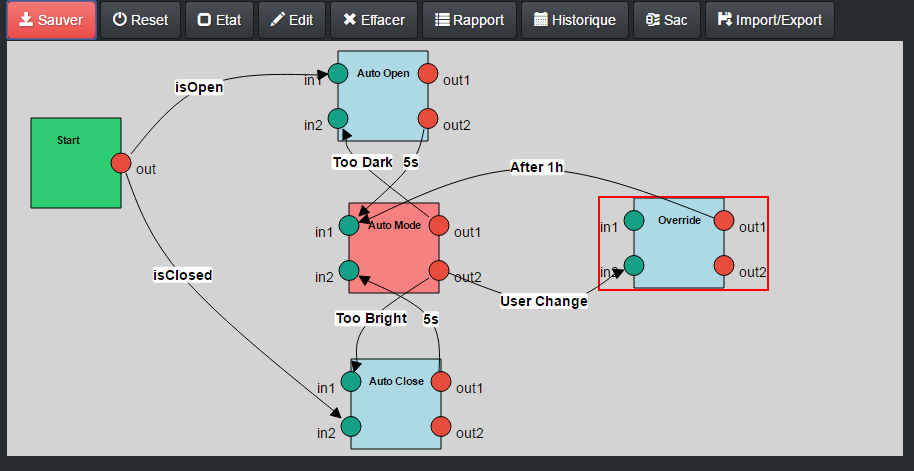
Chaque workflow possède son propre sac de variable, indépendant de celui des autres workflows. Pas de soucis de collision de nom.

Les « Bags » sont visibles et rafraichis sur l’écran du workflow



### Autre exemple : contrôler les stores en fonction de la lumière

Voici un workflow pour contrôler les stores en fonction d’un capteur de luminosité, mais avec une fonction « override » qui bloque le store pour 1 heure des que l’utilisateur impose une position



Voici le rapport de workflow détaillé

Details

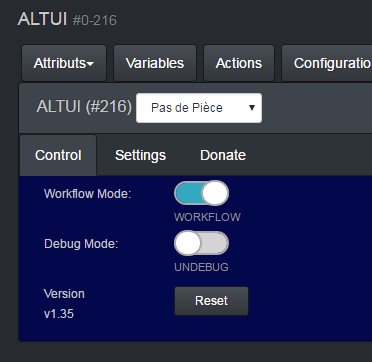
* Start - eaa39e8e-53c3-4f1c-a49f-19db760e6d2f
  + Global Conditions
    - Trigger, device:'**Day or Night**' (0-80) when urn:rts-services-com:serviceId:DayTime-Status (new == '0')
  + Transitions
    - 'isClosed' - d13c711d-2543-4cc8-b04b-275ffb3599cf
      * **When**
        + device:'**Store Cote**' (0-29) when urn:upnp-org:serviceId:Dimming1-LoadLevelStatus (new == '0')
      * **Moves To**
        + State: Auto Close
    - 'isOpen' - 8c76e108-64ed-46b4-b263-c8c2955a35c9
      * **When**
        + device:'**Store Cote**' (0-29) when urn:upnp-org:serviceId:Dimming1-LoadLevelStatus (tonumber(new) > 0)
      * **Moves To**
        + State: Auto Open
* Auto Open - 227e142b-8ca1-4e9b-9477-cd151632eeaa
  + OnEnter
    - on 0-30 urn:upnp-org:serviceId:Dimming1-SetLoadLevelTarget [{"name":"newLoadlevelTarget","value":100}]
    - on 0-29 urn:upnp-org:serviceId:Dimming1-SetLoadLevelTarget [{"name":"newLoadlevelTarget","value":100}]
  + Transitions
    - '5s' - ab04821e-fb18-4a7d-a0f5-eb8dd74d815b
      * **When**
        + Timer: '5s' expiration 5s
      * **Moves To**
        + State: Auto Mode
* Override - e97ea897-d50a-4284-9761-c9d6f9d627e0
  + Transitions
    - 'After 1h' - 3ea5657e-6918-409e-bdcf-cc056958f5d2
      * **When**
        + Timer: 'After 1h' expiration 3600s
      * **Moves To**
        + State: Auto Mode
* Auto Close - 6fa85087-dacf-4cc4-a386-fa89529f4012
  + OnEnter
    - on 0-29 urn:upnp-org:serviceId:Dimming1-SetLoadLevelTarget [{"name":"newLoadlevelTarget","value":0}]
    - on 0-30 urn:upnp-org:serviceId:Dimming1-SetLoadLevelTarget [{"name":"newLoadlevelTarget","value":0}]
  + Transitions
    - '5s' - 89ff8844-d246-42dd-b3a7-0f8986ce0f3c
      * **When**
        + Timer: '5s' expiration 5s
      * **Moves To**
        + State: Auto Mode
* Auto Mode - da88e8b7-7d01-45c6-b5b0-cf1f1aacb2e8
  + Transitions
    - 'Too Dark' - fed26280-c344-4d93-9913-3c3c6a5ae66c
      * **When**
        + Trigger, device:'**3 in 1 sensor (light)**' (0-179) when urn:micasaverde-com:serviceId:LightSensor1-CurrentLevel (tonumber(new) < 200)
      * **Moves To**
        + State: Auto Open
    - 'Too Bright' - 14719556-87e0-41a5-9544-c520e9f2b399
      * **When**
        + Trigger, device:'**3 in 1 sensor (light)**' (0-179) when urn:micasaverde-com:serviceId:LightSensor1-CurrentLevel (tonumber(new) > 800)
      * **Moves To**
        + State: Auto Close
    - 'User Change' - 3b91f92d-272f-472e-b8e2-76433d8a0f20
      * **When**
        + Trigger, device:'**Store Cote**' (0-29) when urn:upnp-org:serviceId:Dimming1-LoadLevelTarget (true)
      * **Moves To**
        + State: Override

### La sauvegarde

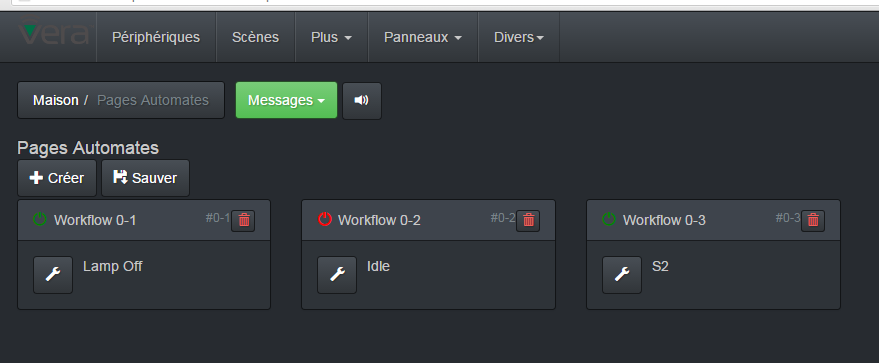
Tout comme les pages utilisateur ( customs ) , les workflow sont sauvegardés dans les variables du device ALTUI et donc ne nécessitent aucun fichiers additionnels, non gérés par VERA et profite donc du backup automatique et de la rotation des backups, au même titre que n’importe quel device/variable de la vera.

### Le contrôle

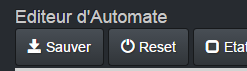
ALTUI offre un interrupteur qui contrôle l’exécution générale du moteur de workflow. En cas de problème il est possible de simplement passer cet interrupteur a OFF pour arrêter cette partie du logiciel



Individuellement, il est possible de mettre en mode « pause » n’importe quel workflow  comme présenté ci-dessous avec la petite icone rouge (comme les scènes)



Enfin, sur le workflow lui-même, il y a un bouton reset qui permet de faire un RAZ du workflow, et redémarrer dans l’état START :



Voilà j’espère avoir attisé votre intérêt pour les machines a états de ALTUI, à vos crayons et partagez vos réalisations avec les autres …