**ALTUI : User Interface alternative pour VERA - Extensions**

# Introduction

**AltUI** est un plugin pour offrir une nouvelle user interface a la VERA (qu’elle soit sous UI5 ou UI7 ). ALTUI tourne sur n’importe quel périphérique client ( PC MAC Tablette Phone ) et s’adapte automatiquement au format. Elle est accessible en local mais aussi en mode remote à partir d’internet.

Cette application a déjà été présentée dans cet article ( <http://xxxxx> ) mais il est utile de revenir sur la partie extensibilité de son architecture.

L’un des challenges d’une user interface pour VERA est de faire face à la communauté des plugins développés par tous les auteurs de la communauté sans avoir à modifier son code pour chaque nouvelle réalisation. Tout n’est pas possible à 100% mais AltUI offre une compatibilité de base avec l’existant, mais offre aussi la possibilité d’étendre sa fonctionnalité par l’intermédiaire de plugin dynamiquement chargés.

# Compatibilité avec l’existant

Chaque auteur de plugin VERA sait qu’il doit définir un certain nombre de fichiers pour UI5 UI7 qui définissent son device , son type , les actions qu’il peut exécuter, ainsi qu’un fichier .JSON qui donne des informations a UI5 UI7 pour afficher son plugin ( icone, bouton, pages de configuration, etc. )

Plutôt que de redemander à chaque auteur de refaire ce travail pour ALTUI, ALTUI récupère les informations données par les auteurs des plugins dans les fichiers standard sur UI5/UI7 et s’en sert. Ainsi les pages de configuration des devices s’affichent dans ALTUI. Les pages de type « flash » dont la composition est définie dans les fichiers .JSON du plugin ou bien les pages de type « javascript » qui doivent utiliser un API définie par VERA et que ALTUI émule afin d’afficher la page proprement (dans 90% des cas, il reste des cas un peu compliques mais faite part de vos retours).

De la même façon, grâce à ces fichiers JSON, ALTUI récupère la définition des icônes ainsi que les règles d’affichage définies par l’auteur afin que l’icône change dynamiquement en fonction de l’état du plugin. De ce fait, les icones se comportent comme sur UI5 UI7 et même mieux en mode remote car elles sont affichées proprement dans ce cas, ce que UI5 U7 ne font pas ( ! )

Il en est de même pour les actions UPNP supportées par un plugin UI5 UI7 et leurs paramètres qui sont documentés dans les fichiers service S\_xx.xml, ALtUI s’en sert pour afficher une boite de dialogue permettant de lancer ces actions, voire de les inclure dans une scène via le scène éditeur.

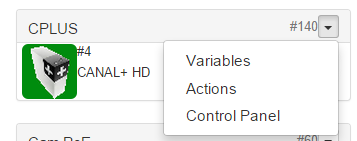
# Extensions

Mais si AltUI fourni des représentations par défaut il peut être utile d’aller plus loin, ALTUI permet de rajoute dynamiquement un module purement javascript afin de customiser l’affichage de 2 elements:

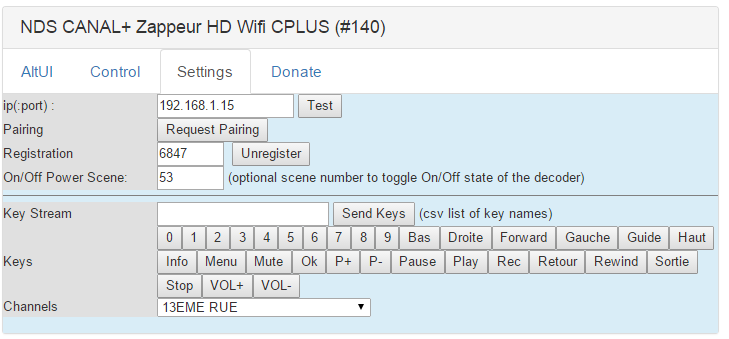
1. du dashboard pour le device pour un affichage minimaliste de l’essentiel, critére primordial afin de ne pas perdre d’espace et d’afficher beaucoup de device sur la page contrairement à UI7
2. Créer un control panel pour des fonctions interactives plus belles et avancées.

Quelques exemples :

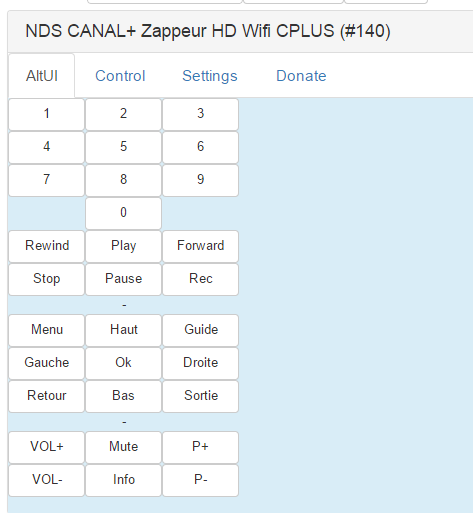
|  |  |
| --- | --- |
| Un dashboard par défaut | Un dashboard customisé par ALTUI |
|  |  |



De la même manière, une autre customisation est possible pour les control panels des devices. lorsque l’utilisateur demande à voir le control panel du périphérique, ALTUI va afficher sans que rien ne soit nécessaire les pages ‘flash’ et ‘json’ définies dans UI5 UI7 par exemple :



Mais si l’auteur du plugin veut créer un control panel supplémentaire et spécifique, il va pouvoir le faire, purement en javascript ce qui permet par exemple d’offrir un meilleur rendu, sans que ALTUI n’ait besoin d’être modifié, recompilé ou de republier une nouvelle version, et ci y compris sur les tablettes & téléphones contrairement à la quasi-totalité des applications mobiles pour VERA existantes.



# Extensions : Comment faire

Tout d’abord, le développeur doit créer un fichier javascript. Il est recommandé afin de facilite le debug d’avoir une toute première ligne comme ceci car le module sera dynamiquement chargé et risque de ne pas apparaitre autrement dans les debuggers javascript ( chrome, firefox.. )

|  |
| --- |
| //# sourceURL=J\_ALTUI\_iphone.js |

Ensuite le pattern des modules sous javascript sera utilisé afin d’éviter les conflits de nom, ainsi le fichier va contenir par exemple :

|  |
| --- |
| //# sourceURL=J\_ALTUI\_iphone.js  var ALTUI\_IPhoneLocator= ( function( window, undefined ) {  // return the html string inside the .panel-body of the .altui-device#id panel  function \_drawIPhone(devid, device) {  return “<p>Hello World!</p>”;  };    // explicitly return public methods when this object is instantiated  return {  //---------------------------------------------------------  // PUBLIC functions  //---------------------------------------------------------  drawIPhone : \_drawIPhone  };  })( window ); |

La fonction drawIPhone est celle qui sera appelée par ALTUI une fois qu’il aura chargé le module. Ce nom est bien sur totalement libre. Cette fonction doit simplement retourner la chaine HTML qui doit être affichée à l’intérieur du cadre du device. C’est-à-dire cette partie le reste étant toujours dessiné par AltUI.



ALTUI fourni des APIs afin de récupérer les informations/variables de devices , de lancer des actions UPNP etc qui sont disponibles , donc dans le cas présent cela donne :

|  |
| --- |
| function \_drawIPhone(devid, device) {  var dist = parseFloat(**VeraBox.getStatus**( devid, 'urn:upnp-org:serviceId:IPhoneLocator1', 'Distance' ));  var unit = **VeraBox.getStatus**( devid, 'urn:upnp-org:serviceId:IPhoneLocator1', 'Unit' );  var mute = **VeraBox.getStatus**( devid, 'urn:upnp-org:serviceId:IPhoneLocator1', 'Muted' );    var html ="";  html+=("<span class='altui-iphone' > "+dist+" </span>");  html+=("<small > "+unit+" </small>");  html += **ALTUI\_PluginDisplays.createOnOffButton**( mute,"altui-onoffbtn-"+devid, "Unmuted,Muted" , "pull-right");  html += "<script type='text/javascript'>";  html += " $('div#altui-onoffbtn-{0}').on('click touchend', function() { ALTUI\_IPhoneLocator.toggleMute({0},'div#altui-onoffbtn-{0}'); } );".format(devid);  html += "</script>";    return html;  }; |

Il se peut que des styles soit nécessaires, ALTUI offre aussi la possibilité de définir des styles additionnels, encore par l’intermédiaire d’une fonction :

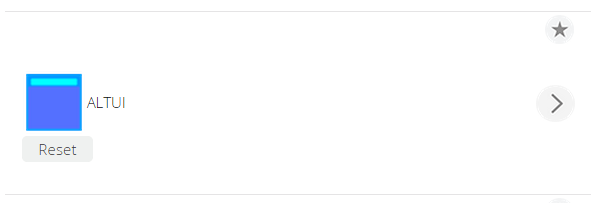
|  |
| --- |
| // return styles needed by this plugin module  function \_getStyle() {  var style="";  style += ".altui-iphone { font-size: 16px; }";  return style;  }; |

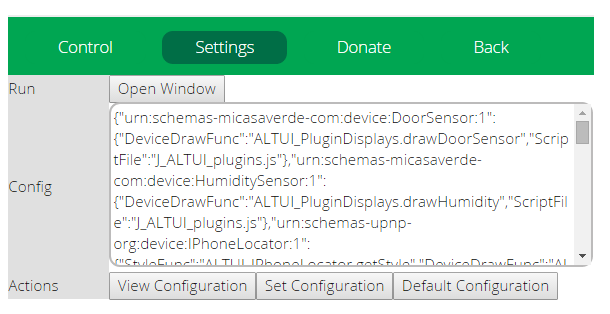
Tout ceci ensemble cela donne :

|  |
| --- |
| //# sourceURL=J\_ALTUI\_iphone.js  var ALTUI\_IPhoneLocator= ( function( window, undefined ) {    // return styles needed by this plugin module  function \_getStyle() {  var style="";  style += ".altui-iphone { font-size: 16px; }";  return style;  };    // return the html string inside the .panel-body of the .altui-device#id panel  function \_drawIPhone(devid, device) {  var dist = parseFloat(VeraBox.getStatus( devid, 'urn:upnp-org:serviceId:IPhoneLocator1', 'Distance' ));  var unit = VeraBox.getStatus( devid, 'urn:upnp-org:serviceId:IPhoneLocator1', 'Unit' );  var mute = VeraBox.getStatus( devid, 'urn:upnp-org:serviceId:IPhoneLocator1', 'Muted' );    var html ="";  html+=("<span class='altui-iphone' > "+dist+" </span>");  html+=("<small > "+unit+" </small>");  html += ALTUI\_PluginDisplays.createOnOffButton( mute,"altui-onoffbtn-"+devid, "Unmuted,Muted" , "pull-right");  html += "<script type='text/javascript'>";  html += " $('div#altui-onoffbtn-{0}').on('click touchend', function() { ALTUI\_IPhoneLocator.toggleMute({0},'div#altui-onoffbtn-{0}'); } );".format(devid);  html += "</script>";    return html;  };      // explicitly return public methods when this object is instantiated  return {  //---------------------------------------------------------  // PUBLIC functions  //---------------------------------------------------------  getStyle : \_getStyle,  drawIPhone : \_drawIPhone,  toggleMute : function (devid,htmlid) {  ALTUI\_PluginDisplays.toggleButton(  devid,  htmlid,  'urn:upnp-org:serviceId:IPhoneLocator1',  'Muted',  function(id,newval) {  UPnPHelper.UPnPAction(  devid,  'urn:upnp-org:serviceId:IPhoneLocator1',  'SetMute',  {newMuteStatus:newval} );  }  );  };  })( window ); |

Maintenant reste à voir comment configurer tout cela dans ALTUI sans qu’une nouvelle version soit nécessaire. 2 étapes sont nécessaires :

1. Charger sur la Vera le fichier script de la même manière que l’on charge un fichier de plugin ( par APPs/ My Apps / Files )
2. Déclarer ce fichier a ALTUI pour qu’il puisse s’en servir. Pour cette étape 2, tout se passe au niveau du plugin ALTUI dans la UI standard de VERA. Tout d’abord, trouver le device ALTUI



Ensuite aller dans les settings pour voir une variable « Config » du plugin. Il s’agit d’une variable qui doit contenir un object JSON qui va nous donner la configuration. 

On tape la nouvelle configuration et on clique **Set Configuration**, ensuite comme d’habitude avec VERA on reload LUUP et on force un refresh de son browser. Attention aux erreurs, cela ne pardonne pas, cela doit être du JSON 100% juste. Le bouton **Default Configuration** permet toujours de revenir à la configuration par défaut telle que fournie par moi, au cas où.

En Cliquant sur le boutton « **View Configuration** », on affiche une page qui montre l’objet JSON en plus clair. Cet objet JSON est au format suivant

{

« devicetype » : {

StyleFunc : « nom complet de la fonction des styles »

DeviceDrawFunc : « nom complet de la fonction de display du device dashboard »

DeviceIconFunc : « nom complet de la fonction de display de l’icon du device»

ControlPanelFunc : « nom complet de la fonction de display du control panel »

ScriptFile: « nom du script file a charger»

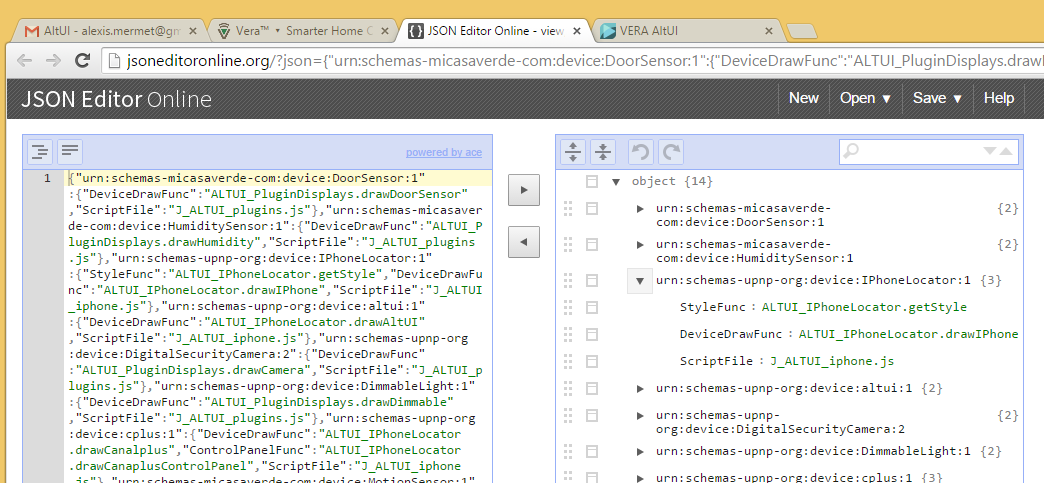
},

… autant que nécessaire, ne pas oublier la virgule entre chaque …

}

Dans tous les cas, si une fonction n’est pas fournie, AltUI fournit une implémentation par défaut.

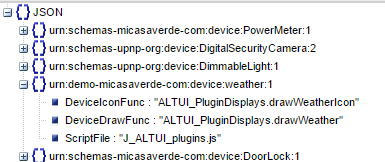
Ce qui donne :



Voilà cela suffit à ALTUI, lorsque qu’il rencontrera un périphérique de type « devicetype », il va charger le module javascript déclaré ici, puis charger les styles CSS définis dans la fonction style si elle existe, puis dessinera la cadre par default du device et appellera la fonction DeviceDrawFunc pour finir l’intérieur du cadre.

Un exemple de fonction **deviceIconFunc** avec le weatherUnderground plugin qui chargée de retourner le code HTML pour l’icône de device. Cela doit être un <IMG> avec les attributs soulignes en rouge ci dessous.

La configuration dans la variable config d’AltUI étant :



|  |
| --- |
| function \_drawWeatherIcon(devid, device) {  var html ="";  var conditionGroup = VeraBox.getStatus(  devid,  'urn:upnp-micasaverde-com:serviceId:Weather1',  'ConditionGroup');  var newsrc = "http://icons.wxug.com/i/c/i/"+conditionGroup+".gif";  return "<img class=**'altui-device-icon pull-left** img-rounded' src='"+newsrc+"' alt='"+conditionGroup+"' **onerror='UIManager.onDeviceIconError("+device.id+")'** ></img>";  }; |

Parlons à présent un peu de la fonction control Panel. Lorsque l’utilisateur demande l’affichage des control panel du device, ALTUI va regarder si un control panel custom a été défini et va l’afficher dans ce cas, sinon il va simplement afficher les pages définies dans UI5 UI7 par l’auteur du plugin vera.

Le control panel custom est lui aussi une fonction javascript mais ici l’objectif étant de donné plus de liberté au code pour l’affichage et l’interactivité avec l’utilisateur, on procède un peu différemment. Au lieu de retourner une simple chaine HTML, on passe à la fonction le nœud DOM parent ( un Object HTML qui est le containeur dans lequel la fonction est libre de faire ce qu’elle veut ).

Par exemple, le control panel custom pour le plugin VERA CanalPlus se présente sous la forme suivante :

|  |
| --- |
| function \_drawCanaplusControlPanel(devid, device, domparent) {  var html="";  html +="<div class=''>";  html += "<table id='altui-cplus-keytbl'>";  html += "<tbody>";  html+="<tr><td><button class='altui-cplus-button btn btn-default' id='1'>1</button></td><td><button class='altui-cplus-button btn btn-default' id='2'>2</button></td><td><button class='altui-cplus-button btn btn-default' id='3'>3</button></td></tr>";  // CODE enlevé par soucis de simplification    html += "</tbody>";  html += "</table>";  html +="</div>";  $(domparent).append(html);  $(".altui-cplus-button").click( function() {  var id = $(this).prop('id');  UPnPHelper.UPnPAction( devid, 'urn:upnp-org:serviceId:cplus1', 'SendKey', {keyStream:id} );  });  }; |

On remarque que le code de cette fonction rajoute elle-même le contenu HTML qu’elle génère dans le nœud parent domparent et que n’importe quelle manipulation du DOM est possible ici.

|  |
| --- |
| $(domparent).append(html); |

On remarque aussi que la fonction définie l’interactivité et les fonctions de callback nécessaires. Ici elle enregistre un click Handler afin de réagir lorsque l’utilisateur presse sur un bouton pour appeler une fonction UPNP standard du périphérique en question. C’est du jQuery classique

|  |
| --- |
| $(".altui-cplus-button").click( function() {  var id = $(this).prop('id');  UPnPHelper.UPnPAction( devid, 'urn:upnp-org:serviceId:cplus1', 'SendKey', {keyStream:id} );  }); |

# Allez plus loin

Toutes les sources de ALTUI sont disponibles sur <http://code.mios.com/trac/mios_alternate_ui/browser>

J\_ALTUI\_plugins.js :

* Les exemples fournis par ALTUI sont dans le fichier J\_ALTUI\_plugins.js dont les développeurs pourront s’inspirer et voir les extensions réalisées pour les Door Lock, les Cameras, etc.

J\_ALTUI\_verabox.js :

* Les API pour appeler la functions UPNP de la VERA ou recuperer/changer les valeurs des variables d’un device VERA entre autre l’objet **UPnPHelper** et ses methodes, entre autre
  + UPnPSetAttr : \_UPnPSetAttr, // ( deviceID, attribute, value, cbfunc)
  + UPnPSet : \_UPnPSet, // ( deviceID, service, varName, varValue )
  + UPnPAction : \_UPnPAction, // ( deviceID, service, action, params, cbfunc )
* Ainsi que l’objet **VeraBox**  et ses methodes principales, entre autre
  + getIcon : \_getIcon, // workaround to get image from vera box
  + getWeatherSettings : \_getWeatherSettings,
  + getBoxInfo : \_getBoxInfo,
  + getLuaStartup : \_getLuaStartup,
  + getRooms : \_getRooms,
  + getDevices : \_getDevices,
  + getDeviceByID : \_getDeviceByID,
  + getDeviceBatteryLevel : \_deviceBatteryLevel,
  + getDeviceStaticUI : \_getDeviceStaticUI,
  + getDeviceActions : \_getDeviceActions,
  + getDeviceEvents : \_getDeviceEvents,
  + getScenes : \_getScenes,
  + getSceneByID : \_getSceneByID,
  + getPlugins : \_getPlugins,
  + getPluginByID : \_getPluginByID,
  + getHouseMode : \_getHouseMode,
  + setHouseMode : \_setHouseMode,
  + setStatus : \_setStatus,
  + getStatus : \_getStatus,
  + getStates : \_getStates,
  + runScene : \_runScene,
  + reloadEngine : \_reloadEngine,
  + isRemoteAccess

J\_ALTUI\_utils.js :

* des utilitaires importants comme le formatage de string, les escapes HTML, la fonction de Debugtrace avec l’objet **AltuiDebug**

Voilà qui devrait permettre aux développeurs HTML/javascript ou à des auteurs de plugin VERA de faire des extensions pour ALTUI, je peux les intégrer dans une release ALTUI si certains sont intéressés a publier leur réalisation.