



Advanced customization of the captive portal

Version 5.1



Table of Contents

Advanced customization of the captive portal	1
1 Introduction	3
2 Portal structure	4
3 Exporting the source code	5
3.1 Source code organization	5
3.1.1 Source code listing.....	6
3.1.2 Contents of the "portal" folder.....	6
3.1.3 Contents of the "resources" folder.....	7
3.1.4 Contents of the "body" folder	7
3.1.5 Contents of the "reserved_block" folder	8
3.1.6 Contents of the "subscription" folder	8
4 Source code modification	9
4.1 Adding a text block	9
4.1.1 HTML mode	9
4.1.1.1 Creation of the HTML file	9
4.1.1.2 Positioning the text block using CSS	10
4.1.2 HTML + XML mode	11
4.1.2.1 Creation of the HTML file	11
4.1.2.2 Positioning the text block using CSS	11
4.1.2.3. 4.1.2.3. Managing languages using XML.....	12
4.2 Adding an image block.....	12
4.2.1 Creation of the HTML file	12
4.2.2 Positioning the image block using CSS	13
4.2.3 4.2.3. Copying the image	13
4.3 Modifying a translation.....	13
4.4 Modifying page rendering (CSS)	14
4.4.1 Superposition of different images (layer management)	14
4.4.2 Adding or changing a background image, or color.....	14
4.4.3 Adding differential borders.	15
4.5 Behavior modification (mail mode display in the home page).....	15
5 Importing the source code.....	17
6 Captive portal API.....	18
6.1 Main functions	18
6.2 Other functions	28

1 Introduction

This guide is intended for system and/or network administrators responsible for managing UCOPIA controllers.

This guide explains how to modify the source code of the captive portal visual model if you need advanced customization (e.g.: modification of behavior). This is useful for customization that cannot be made through the portal graphical editor.

We present examples for customizing the portal source code: export, import and modification. We also present the portal API to communicate with the UCOPIA controller.

2 Portal structure

The portal consists of many elements describing the page background, the main sections and also the reserved block. These components are organized as follows.

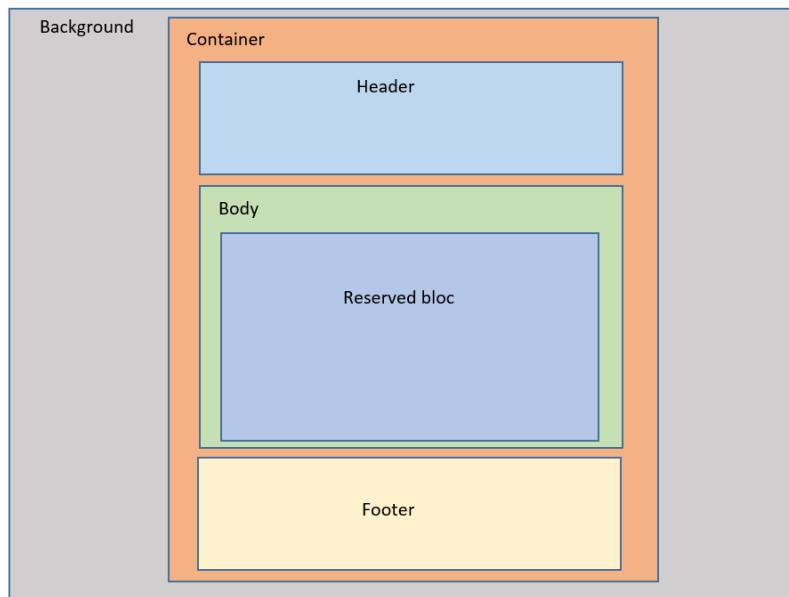


Figure 2.1. Captive portal structure

The sections are a main one named "container" and 3 subsections named "header", "body" and "footer".



Note

An additional “floating” section can appear in case of information portal.

3 Exporting the source code

The table displaying all available portal visual models (the "Configuration/Customization/Portals" menu in the administration tool) offers an icon for each visual model to export the corresponding source code.

Click on the export icon corresponding to the visual model to modify.

Model name		Zones	Configurations	Edit	Actions
Factory visual models					
	classic				
	neutral				
	welcome				
Visual models					
	default (*)	2	1		
	OpenX (*)	3	2		
	demo	0	0		

(*) This visual may not be compatible with all available features in the current controller version.

Figure 3.1. Exporting a visual model

The code of the visual model of the portal is presented as a compressed archive to be downloaded.

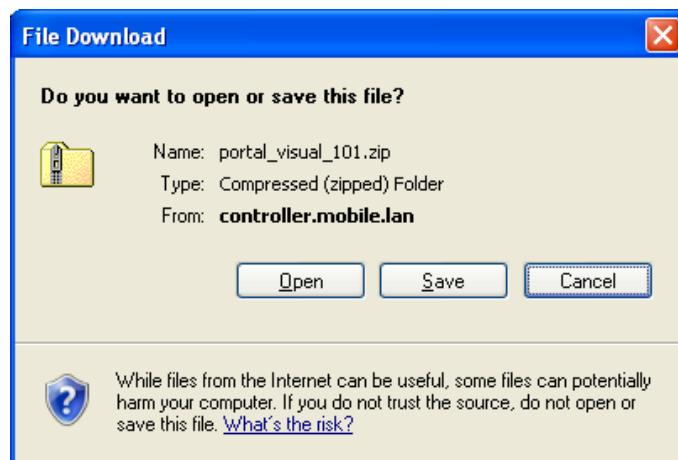


Figure 3.2. Downloading a visual model

The archive file is named "portal_visual_<number>.zip".

3.1 Source code organization

The source code of the portal is presented as an archive file (.zip).

The archive contains the following files:

- **visuel.zip**: an archive file

- **visuel.nfo**: internal information file, present only when the visual model is editable inside the editor offered by the portal.

The visuel.zip archive contains the following folders:

- **deleg**: resources for the delegation portal
- **portal**: resources for the captive portal, in PC format
- **portal_mobile**: resources for the captive portal, in Smart-phone format
- **portal_tablet**: resources for the captive portal, in tablet format
- **smartphone_apps**: resources for mobile applications customization (this directory is not always available)

Warning

In the current version it's not possible to modify the delegation portal code. Changes may be lost, or cause malfunction of the portal.

3.1.1 Source code listing

The set of file names present in the resource folders are prefixed by the name of their folder. For example, the names of the files inside the "portal" folder have the "portal" prefix.

There are three main file types:

- Files of type tmpl.html, which have only HTML code
- Files of type lang.xml, which have only the text translations in the different languages, stored in XML format
- Files of type .js, which have only JavaScript for dynamic rendering of the portal.

Names which are not suffixed (that is, names without extension), correspond to folders.

3.1.2 Contents of the "portal" folder

- **AUTOGEN_FULL_WEBPAGE**: contains the 3 files generated by the controller from data contained in the "resources" folder.

Warning

Do not modify or delete these files.

- **portal_index.html**: the HTML file containing the HTML code for the visual model.
- **portal.lang.xml**: the XML file containing all translations for the visual model.
- **portal.js**: the JS file containing all JavaScript code for the visual model.

- **portal.css**: the CSS file containing all styles for the visual model.
- **ressources**: this folder contains the set of files useful to elaborate the "portal_index.html", "portal.lang.xml" and "portal.js" files. It's inside this folder, and more precisely inside its sub-folders, where modifications and additions must be made.

3.1.3 Contents of the "resources" folder

- **container**: contains files describing the subsection of the page including the header, the body and the footer. Allows, for example, to present a portal always centered on screen.
- **header**: contains the header of the portal page. It's always placed at the top of the page.
- **body**: the visual model main section, containing the languages block, the reserved block, and also the set of added text and images blocks.
- **footer**: contains the footer of the portal page. The footer is always placed at the bottom of the page, unless the page is bigger than the screen displaying it.
- **custom**: contains the set of text and image blocks added with the portal editor. You should add any new text or image block inside this folder.
- **Widgets**: contains all widgets. Currently, only the information widget is available. This directory only appears in case of information portal.
- **_css**: contains the global CSS file of the portal. Could also contain your personal CSS file created if you've used the "Edit your custom CSS" in the portal editor. It may contain other CSS files, in this case you must be careful with inheritance between files. For Ucopia CSS file, global goes first then the custom CSS file.
- **_images**: contains the set of images present in the portal. You should add any new images inside this folder.
- **_javascript**: contains the main JavaScript files. It may contain other JavaScript files used by the portal. As a minimum, it contains the following files:
 - ▶ **jquery-1.5.2.min.js**: jQuery base module
 - ▶ **jquery.sprintf.js**: auxiliary jQuery module.
 - ▶ **portal_global.js**: contains the primitives for communications with the controller.
- **portal_index.lang.xml**: contains the translation of the title and generic errors.
- **portal_index tmpl.html**: the main HTML file, defining the auxiliary resource files used and the META tags (for smart-phone and tablet formats).

3.1.4 Contents of the "body" folder

- **background**: contains the background of the body element.
- **lang_block**: contains the languages flags bar. Dynamic display according to the configuration of the portal.
- **reserved_block**: the reserved block of the portal. This is the main element of the captive portal, managing all the processes for registration, authentication, payment, etc.

3.1.5 Contents of the "reserved_block" folder

Inside this folder, we find the set of files constituting the UCOPIA reserved block.

- **error_info**: the banner displaying error or success messages for user actions.
- **waiting_icon**: a banner displaying a waiting icon, used mainly during redirections for payment in out-of band architecture.
- **feedback**: the post-authentication page, containing various information once the user has been connected (authorized services, connection time slots, etc.)
- **force_modify_pwd**: the page forcing a user to change his password (before the actual connection)
- **logon**: the main authentication page. The user fills here his identifier and password, and accepts the charter.
- **modify_pwd**: the page for password modification, once the user is authenticated.
- **secure_pwd**: the page asking for the portal securing password.
- **subscription**: contains the set of pages managing the subscription modes (SMS, email, ...)
- **portal_reserved_block.lang.xml**: generic translations for the reserved block: months, days, time credit...
- **portal_reserved_block.tpl.html**: the base file for the reserved block.
- **get_purchase_summary** : used by the module in charge of creating the purchase receipt.
- **manage_account** : used by the module in charge of devices and personal information management.
- **pwd_recovery_questions** : used by the module in charge of secret questions and password recovery.
- **upgrade_account** : used by the module in charge of purchasing additional connections.

3.1.6 Contents of the "subscription" folder

Inside this folder, we find the folders defining the different portal operation modes.

- **auto**: automatic connection mode.
- **direct**: free subscription mode.
- **mail**: email subscription mode.
- **One**: « One Click Button » subscription mode.
- **sms**: SMS subscription mode.
- **paypal**: mode with on-line PayPal payment.
- **ogone**: mode with on-line Ogone payment.
- **pms**: mode with PMS invoicing software
- **pps**: mode with PPS prepaid cards.
- **print**: mode with ticket printing.

4 Source code modification

4.1 Adding a text block

You can add a customized text block for example to manage translations according to the different languages available.

There are two ways of adding text blocks: via HTML for simple modifications; or via HTML and XML, which is better suited for advanced modifications (for example: manage languages with Russian or Arabic alphabets).

4.1.1 HTML mode

The following steps must be performed:

1. create an HTML file containing the text block
2. position the text block inside the page using CSS

4.1.1.1 Creation of the HTML file

You must first create a file named "text_block tmpl.html" inside the "custom" folder. The extension "tmpl.html" must be respected.

For the text block to be automatically translated (by clicking on the flags) it must follow these rules:

Each language must be encapsulated inside a <div> having the "id" and "lang" attributes set. If you don't want any language to be visible by default, you must also hide this <div> (using the "style" attribute).

Corresponding source code :

```

<div id="text_block">
  <div id="text_block_fr" lang="fr">
    Place your text in French here
  </div>
  <div id="text_block_de" lang="de">
    Place your text in German here
  </div>
  <div id="text_block_en" lang="en">
    Place your text in English here
  </div>
  <div id="text_block_es" lang="es">
    Place your text in Spanish here
  </div>
  <div id="text_block_it" lang="it">
    Place here your text in Italian here
  </div>
  <div id="text_block_nl" lang="nl">
    Place your text in Dutch here
  </div>
  <div id="text_block_pt" lang="pt">
    Place your text in Portuguese here
  </div>
  <div id="text_block_pt" lang="pt">
    Place your text in Polish here
  </div>
</div>

```

```

<div      id="text_block_pt"      lang="zh_CN"      style="display:none">
  Place      your      text      in      Simplified      Chinese      here
</div>

<div      id="text_block_pt"      lang="ar"      style="display:none">
  Place      your      text      in      Arabic      here
</div>

<div      id="text_block_pt"      lang="jp"      style="display:none">
  Place      your      text      in      Japanese      here
</div>

<div      id="text_block_pt"      lang="jp"      style="display:none">
  Place      your      text      in      Japanese      here
</div>
<div      id="text_block_pt"      lang="th"      style="display:none">
  Place      your      text      in      Thai      here
</div>

<div      id="text_block_pt"      lang="ru"      style="display:none">
  Place      your      text      in      Russian      here
</div>

<div      id="text_block_pt"      lang="id"      style="display:none">
  Place      your      text      in      Indonesian      here
</div>

<div      id="text_block_pt"      lang="ko"      style="display:none">
  Place      your      text      in      Korean      here
</div>

</div>
  
```



Warning

In HTML the values of the "id" attribute must be unique.

4.1.1.2 Positioning the text block using CSS

Once the block is created, you must correctly position it inside the page.



Note

It's also possible to place the text inside one of the existing blocks (for example, the header or the footer).

To position the text block inside the page you must add code in the CSS style sheet.

The style sheet is present in the "_css" folder, and is named "portal_global.css".

We are going to add, as a minimum, the following values in that file:

```
#text_block
  position: absolute; // ignores normal page flow for positioning the element
  top: 10px;          // the top left corner of this block will be 10 pixels below the top
                      // of the top left corner of its parent.
  left: 20px;         // the top left corner of this block will be 20 pixels to the right
                      // of the top left corner of its parent
  z-index: 2;         // regulates on top of which layer the text block is positioned (see
                      // below)
}
```

Note

The "parent" for text and image blocks will be the element having the id "body" if the HTML files of the latter are correctly placed inside the "custom" folder. Otherwise, it will be determined on a case by case basis.

4.1.2 HTML + XML mode

The following steps must be performed:

1. create an HTML file containing the text block
2. position the text block inside the page using CSS
3. manage the translations of the different languages using XML

4.1.2.1 Creation of the HTML file

Create a file named "text_block tmpl.html" inside the "custom" folder.

```
<div
  <span id="text_block_text">Place your text here</span>
</div>
```

4.1.2.2 Positioning the text block using CSS

Once the block is created, you must correctly position it inside the page. To do so, the following code must be added to the CSS style sheet.

The style sheet is present in the "_css" folder, and is named "portal_global.css".

```
#text_block
  position: absolute; // ignores normal page flow for positioning the element
  top: 10px;          // the top left corner of this block will be 10 pixels below the top
                      // of the top left corner of its parent.
  left: 20px;         // the top left corner of this block will be 20 pixels to the right
                      // of the top left corner of its parent
  z-index: 2;         // regulates on top of which layer the text block is positioned (see
                      // below)
}
```

4.1.2.3. Managing languages using XML

The set of translations will have to be placed in the "portal_index.lang.xml" file.

Many tags can be used inside the XML files

- **title:** it must appear only once. Contains the translation of the page's title.
- **text:** the set of "classic" text fragments is grouped under this type (input field labels, notes, messages, etc).
- **button:** the set of text fragments for button labels.
- **generic:** the generic translations (dates, time credits, month of the year).
- **error:** errors sent by the controller are identified by their id.
- **info:** informative messages (for example, successful registration) sent by the controller are identified by their id.

Example:

```
<text>
    <msgid>text_block_text</msgid>
    <de>Place      your      text      in      German      here</de>
    <en>Place      your      text      in      English     here</en>
    <es>Place      your      text      in      Spanish     here</es>
    <fr>Place      your      text      in      French      here</fr>
    <it>Place      your      text      in      Italian     here</it>
    <nl>Place      your      text      in      Dutch       here</nl>
    here</nl>

    <pt>Place      your      text      in      Portuguese here</pt>

    <pl>Place      your      text      in      Polish      here</pt>
</text>
```

4.2 Adding an image block

Adding an image block is similar to the addition of a text block, but without language translations.

To do so you must:

1. create an HTML file containing the image block
2. position the image block inside the page using CSS
3. copy the image inside the "_images" folder

4.2.1 Creation of the HTML file

Create a file named "image_block.tpl.html" inside the "custom" folder. The extension "tmpl.html" here is important.

To add a simple image copy the following lines:

```
<div id="image_block"></div>
```

4.2.2 Positioning the image block using CSS

To position the image block inside the page you must add code in the CSS style sheet.

The style sheet is present in the "_css" folder, and is named "portal_global.css".

We are going to add, as a minimum, the following values in that file:

```
#image block {
  position: absolute; // ignores normal page flow for positioning the element
  top: 10px; // the top left corner of this block will be 10 pixels below the top
  of the top left corner of its parent
  left: 20px; // the top left corner of this block will be 20 pixels to the right
  of the top left corner of its parent
  z-index: 2; // it regulates on top of which layer the image block is positioned
  (see
  below)
}
```

4.2.3 Copying the image

For this step, you only need to copy the "my_image.png" file into the "_images" folder.

4.3 Modifying a translation

Here we describe how to modify the translation of a label or of an error message.

Let's take the example of the error message that can be produced when authentication fails.

"An error has occurred. Please try again, or contact the administrator". To modify this message, you must first identify the file containing it. In this case, the error message is found inside the "portal_logon.lang.xml" file, located inside the "logon" folder (resources/body/reserved_bloc/logon), with ID "error_logon_internal".

```
<error>
<msgid>error_logon_no-pms-package</msgid>
<msgid>error_logon_no-paypal-package</msgid>
<msgid>error_logon_cannot-connect-sql-database</msgid>
<msgid>error_logon_cannot-search-in-sql-database</msgid>
<msgid>error_logon_unknown-paypal-package</msgid>
<msgid>error_logon_unknown-paypal-group</msgid>
<msgid>error_logon_internal</msgid>
<msgid>error_disconnect_internal</msgid>
<!-- Configuration error / Generic error -->
<de>Es ist ein Fehler aufgetreten. Bitte versuchen Sie es erneut oder kontaktieren Sie
den Administrator</de>
<en>An error occurred. Please try again or contact the network administrator</en>
<es>Se ha producido un error. Vuelva a intentarlo o póngase en contacto con el
administrador</es>
<fr>Une erreur est survenue. Merci de réessayer ou de contacter l'administrateur</fr>
<it>Si è verificato un errore. Riprovare o contattare l'amministratore</it>
<nl>Er is een fout opgetreden. Probeer het opnieuw of neem contact op met de
beheerder</nl>
<pt>Ocorreu um erro. Por favor tente contactar de novo com o administrador</pt>
<pl>Wystąpił błąd. Proszę spróbować ponownie lub skontaktować się z
administratorem</pl>
</error>
```

You then need to modify messages in the different languages, to your liking.

**Note**

We note that there's several IDs for the same message. This is to group translations and avoid copying the same message text many times. Thus, you must pay attention if you want to modify the message to a unique error code. You must first dissociate the corresponding ID, and then create a new "<error> ... </error>" element.

4.4 Modifying page rendering (CSS)

Here we describe the way to modify the aspect of the portal page using CSS.

The style sheet is present in the "_css" folder, and is named "portal_global.css".

We are going to illustrate the process through the following examples:

- superposition of different images (layer management),
- adding or changing a background image, or color
- adding differentiated borders.

4.4.1 Superposition of different images (layer management)

To do so, the CSS "z-index" attribute is useful.

The image having the greatest "z-index" will be on top of the stack, and vice versa.

```
#image_block
  z-index: 10;           // This will place the block on top of everything
}
```

**Note**

Text blocks have a "z-index" of 1, image blocks have a "z-index" of 2, languages block a "z-index" of 3, and the reserved block a "z-index" of 4. A "z-index" of 10 will place the image on top of everything, even on top of the reserved block (in this example).

4.4.2 Adding or changing a background image, or color

For text blocks, the editor doesn't allow you to add a background image or color. We can do so, thanks to CSS.

For example, we are going to make the background color of the text block "text_block" red, we modify the CSS as follows.

```
#text_block
  position: absolute; // ignores normal page flow for positioning the element
  top: 10px;           // the top left corner of this block will be 10 pixels below the top
                      // of the top left corner of its parent
  left: 20px;          // the top left corner of this block will be 20 pixels to the right
                      // of the top left corner of its parent
  z-index: 2;          // regulates on top of which layer the text block is positioned (see
                      // below)
}
background-color: #FF0000;
}
```

To add a background image to the same block, you must proceed as follows:

```
#text_block
  position: absolute; // ignores normal page flow for positioning the element
  top: 10px;           // the top left corner of this block will be 10 pixels below the top
                      // of the top left corner of its parent
  left: 20px;          // the top left corner of this block will be 20 pixels to the right
                      // of the top left corner of its parent
  z-index: 2;          // regulates on top of which layer the text block is positioned (see
                      // below)
}
background-image: url('../_images/my_image.png');
}
```

Don't forget to copy the "my_image.png" image to the "_images" folder.

4.4.3 Adding differential borders.

You may wish to add a border to the header section for example, in order to create a separation.

You must then modify the CSS file as follows:

```
#header
  .
  .
  .
  border-bottom-style: solid;
  border-bottom-width: 1px;
  border-bottom-color: #FF0000;
  .
  .
  }
```

4.5 Behavior modification (mail mode display in the home page)

We are going to explain how to change the behavior using an example.

Let's take the case of a portal configured with auto registration via email. The first page presented to the user is the authentication page. This is comprised of a button (or link) taking the user to the registration page. We are going to invert the order of presentation of the pages, the registration page will appear first and then the authentication page.

The portal is based on system states, however we can modify certain parts of the behavior without altering the operation of the portal.

To change the behavior you must modify the "portal_global.js" file located inside the "_javascript" folder, and more specifically the "initPortal" function, as shown below.

```
// Analyze step
displayStep();

mailSubscriptionForm_display(true);
logonForm_display(false);

// Display reserved_block slowly
$('#reserved_block').fadeIn('slow');

}
```

The "mailSubscriptionForm_display" function manages display of the email registration form, so we want to force its display by specifying "true" as an argument.

The "logonForm_display" function manages display of the authentication form, so we want to avoid displaying it by specifying "false" as an argument.

This first modification does place the registration form on the front page, but does not behave correctly if the connected user refreshes the page. In fact the form will be present together with feedback.

We must then condition our display, according to the display states of the authentication block.

```
// Analyze step
displayStep();

if ($.settings.step == "LOGON") {
  mailSubscriptionForm_display(true);
  logonForm_display(false);
}
// Display reserved_block slowly
$('#reserved_block').fadeIn('slow');

}
```

The initial state is LOGON (to present the authentication form), so we want to know if that's the current state when loading the portal.

Now page reloading by a connected user works, but we still have to ensure compatibility with a configuration that consists of adding a password to secure the portal. In fact, after having filled the password to secure the portal, the user is redirected to the authentication form instead of the registration one.

You must then modify the code accordingly.

```
function securePwdFormAccessCallback() {
  displayStep();
  if ($.settings.step == "LOGON") {
    mailSubscriptionForm_display(true);
    logonForm_display(false);
  }
}
```



Warning

These modifications work exclusively for operation with a configuration in email mode.

We can further enhance the behavior further by changing the name of the "Return" button located now on the registration page. In fact it would be more appropriate to call it "Authenticate" now.

To rename the button, you must change its value in the XML files.

The name of the button is found inside the "portal_mail.lang.xml" file located inside the "mail" folder. Its id is "mailSubscriptionForm_back_button".

5 Importing the source code

Once the code has been modified and customized, it will be possible to import it back to the controller by using the modify visual model icon.

During the modification of the visual model you have to select "external" as source of the model and import the file describing the modified model by using the "**Browse...**" button.

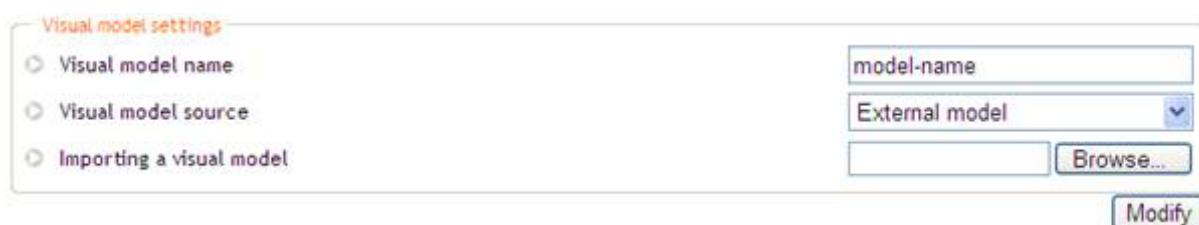


Figure 5.1. Importing an external visual model



Warning

To import back a visual model, the archive must contain the four folders defined above: deleg, portal, portal_mobile, portal_tablet.



Warning

Once the HTML code of the modified portal has been imported in the controller, it's not possible to edit it using the graphical editor.

6 Captive portal API

In the "portal_global.js" file are functions to communicate with the controller, these functions constitute the API of the captive portal and are written in JavaScript.



Warning

It's strongly advised that you avoid modifying functions sending requests to the controller, this may cause malfunction of the portal.

6.1 Main functions

■ Name: authenticate

Authenticates and connects the user.

► Arguments:

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - password: the password of the user [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration]
 - policyAccept: whether the charter has been accepted [optional, except if the option has been activated in the configuration]

■ Name: autoAuthenticate

Connection of a user for a portal configured in automatic mode.

► Arguments:

- callbacks: callback functions [required]
- params:
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration]
 - policyAccept: whether the charter has been accepted [optional, except if the option has been activated in the configuration]

■ Name: backAction

Go back one state in the state diagram.

► Arguments:

- callbacks: callback functions [required]

■ Name: checkUpgradeOrder

Allows to modify a user account after a successful purchase of additional connections.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: displayStep**

Displays the parts of the reserved block linked to the actual state of the transaction.

■ **Name: displayErrorInfo**

Displays translations of the information or error messages in the place dedicated to this effect.

► **Arguments:**

- displayLang: the desired display language [optional]

■ **Name: disconnect**

Disconnects the user.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - passwordDigest: the hash of the password of the user [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration].

■ **Name: forceModifyPwd**

Allows the user to modify its own password when changing the password is forced.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - password: the password of the user [required]
 - newPassword: the new password of the user [required]
 - newPasswordConfirm: confirmation of the new password of the user [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration].

■ **Name: getPwdRecoveryOptionsFromUser**

Gets available options for sending a new password to the user (by SMS, by email, ...).

► **Arguments:**

- callbacks: callback functions [required]
- params:

- login: the user identifier [required]
- emailAddress : the email address of the user [optional]
- prefix : the country dialing code of the user (33 for France, 49 for Germany...) [optional]
- phone : the phone number of the user [optional]
- securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration],

■ **Name: getPurchaseSummary**

Allows to generate a payment receipt when using Ogone.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: getCurrentPackage**

Gets information about the current package.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: getUpgradeOrder**

Upgrades the package in terms of concurrent connections.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - multidevice : the number of expected concurrent connections [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: InformativeWidgetAuthenticate**

Connects an user recognized by the information portal.

► **Arguments:**

- callbacks: callback functions [required],
- params (not used).

■ **Name: InitPortal**

Configures the portal according to configuration parameters (removing non-useful parts of the reserved block).

► **Arguments:**

- data (not taken into account)

■ **Name: modifyPwd**

Allows the user to modify its own password.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - password: the password of the user [required]
 - newPassword: the new password of the user [required],
 - newPasswordConfirm: confirmation of the new password of the user [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: ogoneSendSms**

Sends by SMS the identifiers of the recently created customer account via Ogone registration.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - password : the password of the user [required]
 - prefix : the country dialing code of the user (33 for France, 49 for Germany...) [required]
 - phone : the phone number of the user [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: ogoneSubmitPayment**

Allows to validate the selected package from a portal using Ogone.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - orderid : purchase identifier [required]

■ **Name: paypalSendSms**

Sends by SMS the identifiers of the recently created customer account via PayPal registration.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]

- password: the password of the user [required]
- prefix: the country dialing code of the user (33 for France, 49 for Germany...) [requested]
- phone: the telephone number of the user [requested]
- securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: paypalSubmitPayment**

Creates a PayPal transaction according to the selected package.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - password: the password of the user [required]
 - package: the selected package [required]

■ **Name: regenPwdWithSmsResend**

Generate a new password and send it by SMS.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: regenPwdWithEmailResend**

Generate a new password and send it by email.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: regenPwdWithSecretQuestions**

Generates a new password if the answers to secret questions are correct.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - answer_[...]: answer associated to the selected question [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: refresh**

Keeps the connection open.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - passwordDigest: the hash of the password of the user [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration].

■ **Name: securePwd**

Grants access to the rest of the portal after having entered a correct password to secure the portal.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - securePwd: the password to secure the portal [required].

■ **Name : selectOgonePackage**

Allows to forward the selected package from a portal using Ogone.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - package : the package selected by the user [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration],
 - policyAccept : whether the charter has been accepted [optional, except if the option has been activated in the configuration]

■ **Name: socialNetworkAuthenticate**

Trigger the first step of the Social Network authentication. Check the following parameters.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]
 - policyAccept : whether the charter has been accepted [optional, except if the option has been activated in the configuration]

■ **Name: subscribe**

Creates a user account in the desired registration mode.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - type: the type of registration desired (SMS, mail, direct, one, print, PayPal, Ogone, PPS) [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration]

For "sms" registration type:

- prefix: the country dialing code of the user (33 for France, 49 for Germany...) [required]
- phone: the telephone number of the user [required]
- lastName: the family name of the user [required]
- firstName: the given name of the user [required]
- emailAddress: the email address of the user [optional, except if the option has been activated in the configuration]
- policyAccept: whether the charter has been accepted [optional, except if the option has been activated in the configuration]
- personalField_[1-3] : the 3 additional fields [optional, except if the option has been activated in the configuration]

For "mail" registration type:

- emailAddress: the email address of the user [required]
- lastName: the family name of the user [required]
- firstName: the given name of the user [required]
- prefix: the country dialing code of the user (33 for France, 49 for Germany...) [optional, except if the option has been activated in the configuration]
- phone: the telephone number of the user [optional, except if the option has been activated in the configuration]
- policyAccept: whether the charter has been accepted [optional, except if the option has been activated in the configuration]
- personalField_[1-3] : the 3 additional fields [optional, except if the option has been activated in the configuration]

For both “mail” and “SMS” registration types:

- organizationalUnitName : organization name [optional, except if the option has been activated in the configuration]
- postalAddress : postal address [optional, except if the option has been activated in the configuration]
- postalCode : ZIP code [optional, except if the option has been activated in the configuration]
- postalLocalityName : city name [optional, except if the option has been activated in the configuration]
- postalPostOfficeBox : postoffice box number [optional, except if the option has been activated in the configuration]
- postalStateOrProvinceName : state name [optional, except if the option has been activated in the configuration]

- postalCountryName : country name [optional, except if the option has been activated in the configuration]

For "direct", "one", "print", "Ogone" and "PayPal" registration types:

- lastName: the family name of the user [required]
- firstName: the given name of the user [required]
- login: the new identifier of the user [optional, except if the option has been activated in the configuration]
- password: the new password of the user [optional, except if the option has been activated in the configuration]
- passwordConfirm: confirmation of the new password of the user [optional, except if the option has been activated in the configuration]
- emailAddress: the email address of the user [optional, except if the option has been activated in the configuration]
- prefix: the country dialing code of the user (33 for France, 49 for Germany...) [optional, except if the option has been activated in the configuration]
- phone: the telephone number of the user [optional, except if the option has been activated in the configuration]
- policyAccept: whether the charter has been accepted [optional, except if the option has been activated in the configuration]
- personalField_[1-3] : the 3 additional fields [optional, except if the option has been activated in the configuration]
- organizationalUnitName : organization name [optional, except if the option has been activated in the configuration]
- postalAddress : postal address [optional, except if the option has been activated in the configuration]
- postalCode : ZIP code [optional, except if the option has been activated in the configuration]
- postalLocalityName : city name [optional, except if the option has been activated in the configuration]
- postalPostofficeBox : postoffice box number [optional, except if the option has been activated in the configuration]
- postalStateOrProvinceName : state name [optional, except if the option has been activated in the configuration]
- postalCountryName : country name [optional, except if the option has been activated in the configuration]

For "one" registration type:

- connectPolicyAccept : whether the charter has been accepted [optional, except if the option has been activated in the configuration]

For "PPS" registration type:

- scratchCode: the code revealed on the prepaid card [required]
- captchaCode: the code present on the portal to validate the connection [required]
- policyAccept: whether the charter has been accepted [optional, except if the option has been activated in the configuration]

■ **Name: selectPmsPackage**

Transmits the package chosen by the user for a portal operating in PMS mode.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - password: the password of the user [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration]
 - policyAccept : whether the charter has been accepted [optional, except if the option has been activated in the configuration]

■ **Name: selectPaypalPackage**

Transmits the package chosen by the user for a portal operating in PayPal mode.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - package: the package chosen by the user [required]
 - securePwd: the password to secure the portal [optional, except if the option has been activated in the configuration]
 - policyAccept : whether the charter has been accepted [optional, except if the option has been activated in the configuration]

■ **Name: switchLanguage**

Translates the whole portal to the chosen language.

► **Arguments:**

- callbacks: callback functions [required],
- params:
 - language: the language chosen for portal translation [requested]

■ **Name: shibAuthenticated**

Checks that the user is connected via Shibeoleth . If yes, user authentication is performed at UCOPIA controller level, the feedback is displayed as result.

■ **Name: updateUpgradeOrder**

Allows to confirm the purchase order before online payment.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: the user identifier [required]
 - orderid : purchase identifier [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration],

■ **Name: updatePersonalSettings**

Allows to update personal user information.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - lastName : the last name of the user [optional, except if the option has been activated in the configuration]
 - firstName : the first name of the user [optional, except if the option has been activated in the configuration]
 - emailAddress : the email address of the user [optional, except if the option has been activated in the configuration]
 - prefix : the country dialing code of the user (33 for France, 49 for Germany...) [optional, except if the option has been activated in the configuration]
 - phone : the phone number of the user [optional, except if the option has been activated in the configuration]
 - organizationalUnitName : organization name [optional, except if the option has been activated in the configuration]
 - postalAddress : postal address [optional, except if the option has been activated in the configuration]
 - postalCode : ZIP code [optional, except if the option has been activated in the configuration]
 - postalLocalityName : city name [optional, except if the option has been activated in the configuration]
 - postalPostOfficeBox : postoffice box number [optional, except if the option has been activated in the configuration]
 - postalStateOrProvinceName : state name [optional, except if the option has been activated in the configuration]
 - postalCountryName : country name [optional, except if the option has been activated in the configuration]
 - personalField_[1-3] : the 3 additional fields [optional, except if the option has been activated in the configuration]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: updateDevicesSettings**

Allows to update the list of user's devices.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - device_[...] : comment associated to the device [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: updatePwdRecoverySecretQuestions**

Allows to update the list of secret questions.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - question_[...] : the selected secret question [required]
 - answer_[...] : the answer associated to the selected question [required]
 - securePwd : the password to secure the portal [optional, except if the option has been activated in the configuration]

■ **Name: verifyPayment**

Verify the payment status. Called after coming back from Ogone/PayPal Web site.

► **Arguments:**

- callbacks: callback functions [required]
- params:
 - login: user identifier [required]
 - type: payment type [required]

6.2 Other functions

■ **Name: clearUserSettings**

Cleans user values stored in JavaScript variables.

■ **Name: extractPolicyText**

Returns the translation for the charter object in the chosen language.

► **Arguments:**

- obj: the charter in object form [required]
- lang: the language chosen for translation [optional]

■ **Name: explodeDate**

Extracts values from a JavaScript Date object, in an usable format.

► **Arguments:**

- date: the JavaScript Date object to parse [required]

■ **Name: getInputTranslation**

Returns a text translation in a given language.

► **Arguments:**

- id: identifier of the field to be translated [required]
- displayLang: translation language [optional]

■ **Name: getGenericTranslation**

Returns the translation of a generic label in the chosen language.

► **Arguments:**

- label: the label to translate [required]
- displayLang: the language chosen for the translation [optional]
- plural: return the plural translation [optional]

■ Name: getInputTranslation

Returns the translation of an entry in the chosen language, managing plurals.

► Arguments:

- id: the identifier of the field to translate [required]
- displayLang: the language chosen for the translation [optional]

■ Name: startRefresh, stopRefresh, doRefresh and refreshCallback

Manages refreshing the connection of the user.

■ Name: replaceAllFreeURLs

Search and replace all URLs tags (if needed) by free URLs.

**Note**

Actually only three callback functions can be used.

- beforeSend: before the action of data exchange with the controller
- success: once the data exchange has been successful
- complete: once the data exchange has finished

■ Name: translate

Translates known text according to data present in the XML file.

► Arguments:

- displayLang: the language chosen for text translation [optional]