

H-Sphere Customization Guide

Welcome to the H-Sphere Customization guide. It explains how to customize H-Sphere interface and add custom languages thereto.

- [Advanced Interface Customization](#)
- [Template Customization Rules](#)
- [Customizing System E-Mails](#) (Added: 18 Feb 2002)
- [Customizing Your Control Panel Menu](#) (Revised: 20 Mar 2003)
- [Providing Multilingual Support](#) (Revised: 14 Jun 2002)
- [Updating Translation of H-Sphere Interface](#)
- [Customizing User Signup Forms](#)
- [Compiling Templates With the Client-Side Form Validation](#) (Added: 7 April 2003)

Advanced Interface Customization

Related Docs: • [Template Customization Rules](#) • [XML Customization Rules](#) • [Customizing Control Panel Menu](#) • [Customizing System E-Mails](#) • [Providing Multilingual Support](#) • [Compiling Templates With Client-Side Form Validation](#)

H-Sphere has been created to provide required services with the minimum required adjustments after the installation . The Look and Feel feature of the admin panel allows you to customize control panel colors, images, and texts. See more in [Configuring H-Sphere Interface](#).

However, administrators can have designers and programmers perform more changes than it is allowed by the Control Panel, which would include:

- [Customizing templates](#) and
- [Editing interface texts](#)

You should remember that advanced customization may produce unpredictable results after updating H-Sphere, since updates involve changes in the template structure and page generating procedure.

Throughout this document we will often relate to the `shiva/` directory, which is located in `/hsphere/local/home/cpanel/`. Hereafter, we will refer to it as `shiva`.

Customizing Templates

Templates are HTML documents that contain instructions for including dynamically-generated data. In other words, context data is put into the templates to create relevant H-Sphere pages. Templates are modified to change the web page layout, e-mail notifications, and context help texts.

The default templates are located in the `shiva/shiva-templates/` directory in the 'cpanel' user's home directory.

E-mail notifications issued by H-Sphere are also generated from templates and must be changed according to the instructions laid out in this section. E-mail templates are located in the `shiva/shiva-templates/common/mail` directory. The purpose of each e-mail template is described [here](#).

Online help files are special templates, each with a topic header and a body. They can be modified according to the procedure described in this section. Online help files are located next to templates in the `shiva/shiva-templates/common/online_help` directory.

WARNING: DO NOT MAKE ANY CHANGES TO THE DEFAULT TEMPLATES, because

- 1) You may need them to restore the original setup;
- 2) You will lose all your changes with the next upgrade.

Important: Before you do any customization, log in as cpanel superuser under root:

```
# su - cpanel
```

To implement customization correctly, all template files and directories should have `cpanel:cpanel` ownership, and the `make` directive which is performed to rebuild templates should be run **ONLY** under the `cpanel` user.

Instead, do the following:

Step 1. In `shiva/`, create directory `custom/templates/`.

Step 2. Copy the templates you would like to substitute into `shiva/custom/templates/` preserving their file paths relative to this directory. E.g. If you are going to substitute the `shiva/shiva-templates/not_recomended/to_change/FILE`, copy it to

shiva/custom/templates/*not_recomended/to_change/FILE*. In case of mail templates, copy shiva/shiva-templates/common/mail/*mail_template* to shiva/custom/templates/common/mail/*mail_template*.

The original configuration can be restored without server restart by simply deleting your custom files from the shiva/custom/templates directory.

Important: Don't copy **all** the directory content! Your custom templates will override the default templates and you won't see the new features and bugfixes that come with new versions.

Step 3. Make sure the file is owned by cpanel : cpanel.

Step 4. Modify the templates you have copied to the shiva/custom/templates/ directory. Please carefully follow the [Template Customization Rules](#) and [XML Customization Rules](#).

Step 5. Now that you have had the templates edited, get them used instead of the defaults. In the file shiva/psoft_config/hsphere.properties find the *USER_TEMPLATE_PATH* line. Here, enter the full name of the directory with your custom templates, e.g. /hsphere/local/home/shiva/custom/templates/. **Important: The directory name must end with a slash!** Don't do anything if the directory name is already there.

Step 6. Log in as root and restart H-Sphere. You don't need to restart H-Sphere if you did nothing on step 4.

Editing Interface Texts
(version 2.07 and higher)

This sections explains how to override original menu items and labels for the admin and user control panels with custom texts. E-mail notifications and context help texts are changed according to the procedure described in the [Customizing Templates](#) paragraph.

H-Sphere original menu items and labels are stored in the shiva/psoft/hsphere/lang/ directory in the following files:

- hsphere_lang.properties – template & tooltip texts;
- menu.properties – the navigation menu texts;
- messages.properties – H-Sphere system messages.

WARNING: DO NOT MAKE ANY CHANGES TO THESE FILES!

To alter original menu items and labels, log in as the cpanel user and do the following:

Step 1: Create the directory `shiva/custom/bundles/`. If this directory already exists, skip this step.

Step 2: In the listed files, find the string you want to modify. Next, in the newly created directory, create a new empty file with exactly the same name as the one with the original string. For example, you are going to change the label *Shell Access* to *SSH Access*. It is stored in the `shiva/psoft/hsphere/lang/hsphere_lang.properties` file. Create the file `shiva/custom/bundles/hsphere_lang.properties`.

Step 3: Copy the line with the identifier and the value you want to change into the new file and change its value the way you want. You should use two single quotes (apostrophes) instead of one in labels containing curly brackets, such as `{0}`. For example:

```
search.view_invoice = View Client's Invoice
```

but

```
billing.del_no = No, I don't want to delete {0}
```

IMPORTANT: Don't copy the texts you are not changing!

Step 4: Make sure the file is owned by `cpanel:cpanel`.

Step 5: Now you need to declare the custom files. In the `hsphere.properties` file, uncomment the line that corresponds to the file you have created:

```
CUSTOM_TEMPLATE_BUNDLE = custom.bundles.hsphere_lang
CUSTOM_MENU_BUNDLE = custom.bundles.menu
CUSTOM_USER_BUNDLE = custom.bundles.messages
```

Step 6: Create two symlinks in your document root:

`hsphere_lang_en.properties` to `hsphere_lang.properties` and

`menu_en.properties` to `menu.properties`

Step 7: Log in as root and restart H-Sphere. You don't need to restart H-Sphere if you did nothing on step 4.

Related Docs: • [Template Customization Rules](#) • [XML Customization Rules](#) • [Customizing Control Panel Menu](#) • [Customizing System E-Mails](#) • [Providing Multilingual Support](#) • [Compiling Templates With Client-Side Form Validation](#)

Template Customization Rules

(version 2.1 and higher)

Related Docs: • [XML Customization Rules](#) • [Advanced Customization](#) •
[Customizing Control Panel Menu](#) • [Customizing System E-Mails](#) •
[Providing Multilingual Support](#)

This document describes the rules for template customization for H-Sphere 2.1 and higher. It covers the following issues:

- [Interface Controls](#)
- [Interface Colors](#)

Interface Controls

To generate HTML representation of H-Sphere control panel, we use Java and Java-based FreeMarker package. H-Sphere templates contain calls to FreeMarker functions, variables, and substitutions. Due to the added support of multiple designs in user's control panel, some of the functions needed to be changed. If you are using templates that have been developed for H-Sphere earlier than 2.1, you need to bring parameters in all FreeMarker function calls in line with the following list:

Inserting HTML images:

```
<function draw_image(image_id)>  
<function draw_image_width(image_id, image_width)>  
<function draw_image_alt(image_id, alt_msg)>  
<function draw_image_align(image_id, img_align)>  
<function draw_image_align_alt(image_id, img_align, alt_msg)>  
<function draw_spacer(s__width,s__height)>
```

Inserting ON/OFF buttons:

```
<function draw_state(state,toDisableURL,toEnableURL)>  
<function draw_state_on(toDisableURL)>  
<function draw_state_off(toEnableURL)>  
<function draw_off()>  
<function draw_on()>
```

```
<function draw_on_always(>
```

Inserting other control images:

```
<function draw_add(addURL, label)>  
<function draw_edit(editURL, label)>  
<function draw_delete(deleteURL, label)>  
<function draw_change(editURL, label)>  
<function draw_select(selectURL, label)>  
<function draw_fix(selectURL, label)>  
<function draw_setup(selectURL, label)>  
<function draw_select_signup(plan)>  
<function draw_select_adminsignup(plan)>  
<function draw_preview(previewURL, label)>  
<function draw_preview_large(previewURL, label)>  
<function draw_launch(launchURL, label)>  
<function draw_launch_large(launchURL, label)>  
<function draw_uninstall(launchURL, label)>  
<function draw_credit(jumpURL, label)>  
<function draw_enlarge_credit(jumpURL, label)>  
<function draw_debit(jumpURL, label)>  
<function draw_set_period_begin(jumpURL, label)>  
<function draw_login_account(loginURL, label)>  
<function draw_suspend_account(toSuspendURL, label)>  
<function draw_resume_account(toResumeURL, label)>  
<function draw_delete_account(toDeleteURL, label)>  
<function draw_mail_type(editURL, image_id, label)>  
<function draw_mailbox_type(editURL)>  
<function draw_mailforward_type(editURL)>  
<function draw_mailalias_type(editURL)>  
<function draw_maillist_type(editURL)>  
<function draw_submit(field_name, image_id)>  
<function draw_oscommerce(previewURL, label)>  
<function draw_oscommerce_admin(previewURL, label)>
```

Inserting text labels/messages:

```
<function draw_colored_label(llabel,lcolor)>  
<function draw_colored_label_bold(llabel,lcolor)>  
<function draw_label(label)>  
<function draw_label_bold(label)>  
<function draw_header(header)>  
<function draw_important_label(label)>
```

```
<function draw_important_header(header)>
```

Inserting basic link elements

(t–target, p–picture, a–alt):

```
<function draw_link(url,label)>
<function draw_tlink(url,target,label)>
<function draw_plink(url, image_id)>
<function draw_palink(url, image_id, alter)>
<function draw_ptlink(url, target, image_id)>
<function draw_ptalink(url, target, image_id, alter)>
<function draw_onclick_palink(img, onClick, alt)>
```

Drawing traffic and disk usage charts:

```
<function draw_load_diagram(d_value, d_limit, d_app_percent, d_width)>
<function draw_diagram_legend()>
```

Interface Colors

Designs in H–Sphere 2.1 have customizable color schemes. To display interface elements, we use colors, each having its own `id_handle`. For example, `bgcolor` is the id for the background of the HTML page other than the menu and the header. Its color can be set in the Look And Feel menu.

To get the RGB value of the color by `color_id`, the following syntax is used:

```
${design.color("color_id")}
```

Note: the `color_id` value must be replaced with an appropriate handle and must be enclosed in double quotation marks.

Some colors are predefined in the 'functions' file:

```
<assign LIGHT_STRIP=design.color("table_light_strip")>
<assign DARK_STRIP=design.color("table_dark_strip")>
<assign HEADER_COLOR=design.color("header_color")>
<assign ERROR_COLOR = design.color("error_color")>
<assign BG_COLOR = design.color("bgcolor")>
```

Use them as follows:

`#{BG_COLOR}`

Related Docs: • [XML Customization Rules](#) • [Advanced Customization](#) • [Customizing Control Panel Menu](#) • [Customizing System E-Mails](#) • [Providing Multilingual Support](#)

Customizing System E-Mails

Related Docs: • [Advanced Interface Customization](#) • [Customizing Control Panel Menu](#) • [Providing Multilingual Support](#)

This document introduces you to system e-mail notifications sent to the customers and explains how to customize them.

System e-mail notifications are built up using system notification templates, text documents that contain instructions for including dynamically-generated data. In other words, context data is put into the templates to create complete e-mail messages. All e-mail templates are located in `shiva/shiva-templates/common/mail` of the *'cpanel'* user's home directory and can be customized according to your requirements.

System Notification Templates

Template (in <code>/shiva/shiva-templates/common/mail/</code>)	Sent
<code>/new_account.txt</code>	on account activation
<code>/new_account_moderated.txt</code>	on moderated check account registration

/new_account_moderated_cc.txt	on moderated credit card registration
/trial_account.txt	on account trial period expiration
/welcome.txt	optionally from mass mail
/login_psw.txt	optionally from mass mail
/invoice.txt	<ul style="list-style-type: none"> – on each paid operation – at the beginning of the next billing period – on switching to another billing period
/balance.txt	optionally from mass mail
/overlimit.txt	on reaching the set percentage of resource quota/limit usage
/suspended_account.txt	on account suspension
/resumed_account.txt	on account activation
/transfer_domain.txt	on successful domain transfer
/forgot_passwd.txt	on user's account password request from the login page

Note: To add paragraphs on support and checks info, go to *Look and Feel* → *Misc.Text* and fill in the *Customer Support Info* and *Checks Info* forms.

Customization of System Notification Templates

Important: Before you do any customization, log in as cpanel superuser under root:

```
# su - cpanel
```

To implement customization correctly, all template files and directories should have cpanel:cpanel ownership.

WARNING: DO NOT MAKE ANY CHANGES TO THE DEFAULT TEMPLATES, because

- 1) You may need them to restore the original setup;
- 2) You will lose all your changes with the next upgrade.

Instead, it is recommended to stick to the following steps:

Step 1. In your document root, create the directory `shiva/custom/templates/`. It may already be there.

Step 2. Copy the templates you would like to override (e.g. `shiva/shiva-templates/common/mail/mail_template.txt` into `shiva/custom/templates/common/mail/mail_template.txt`).

The original configuration can be restored without server restart, simply by deleting your custom e-mail templates from the `shiva/custom/templates` directory.

Important: Don't copy the whole directory! Your custom templates will override all default templates and you will not see the new features and bugfixes that come with new versions.

Step 3. Modify the templates that you have copied to the `shiva/custom/templates/` directory. It is important to keep FreeMarker expressions (such as `${...}`) intact while editing texts in the templates.

Step 4. Now that you have edited the templates, get them used instead of the defaults. In the file `shiva/psoft_config/hsphere.properties` find the `USER_TEMPLATE_PATH` line. Here, enter the full name of the directory with your custom templates, e.g. `/hsphere/local/home/shiva/custom/templates/`.

*You might have `/hsphere/local/home/cpanel/shiva/custom-templates/"` already written in `USER_TEMPLATE_PATH` line, in this case we advise you to change it to `"/hsphere/local/home/shiva/custom/templates/`

Important: The directory name must end with a slash!

Step 5. Log in as root and restart H-Sphere.

You don't need to restart H-Sphere if you did nothing on step 4.

Related Docs: • [Advanced Interface Customization](#) • [Customizing Control Panel Menu](#) • [Providing Multilingual Support](#)

Customizing Control Panel Menu

Related Docs: • [Advanced Interface Customization](#) • [Customizing System Emails](#) • [Providing Multilingual Support](#)

You can customize the Control Panel menu by:

- [Restructuring the menu](#);
- [Assigning external links to menu items](#);
- [Editing menu texts](#);
- [Customizing menu design](#).

Before you start customizing the CP menu, it is highly advisable that you become familiar with customizing H–Sphere templates (see the [Advanced Interface Customization](#) chapter of this manual).

Throughout this document we will refer to the `shiva/` directory, which is located in `/hsphere/local/home/cpanel/`. Hereafter, we will refer to it as `shiva`.

Restructuring the Menu

This section explains how to change the menu structure for existing plans or create new menus for new plans. You are expected to be familiar with XML technology.

The structure of the control panel navigation menu is defined in the file `shiva/psoft/hsphere/menu.xml`. It is a simple XML document consisting of 3 main parts:

1. DTD scheme description;
2. Definition of menu items and groups (tag `<menus>`);
3. Definition of menus for different hosting plans (tag `<interface>`).

Here is a fragment from the second part:

```

<menus>

<menu name="SiteStudio" label="sitestudio.label" defaultitem="SiteStudio-edit" tip="sitestudio.tip">
  <menuitem name="SiteStudio-edit" label="sitestudio.edit.label"
    URL="list.html" resource="" tip="sitestudio.edit.tip"/>
  <menuitem name="SiteStudio-add" label="sitestudio.add.label"
    URL="add.html" resource="" tip="sitestudio.add.tip"/>
</menu>
.
.
.
</menus>

```

As you can see, the <menus> part includes groups of menu items where each of group is defined by the tags <menu ...>...</menu> and comprises definitions of items in this group (the <menuitems .../> tags).

Attributes in the <menu> tag have the following meaning:

- name – the name of the group;
- label – the mnemonical identifier defined in the menu.properties file to show the item text;
- defaultitem – the name of the default active item in the group;
- tip – the identifier to define the alternative text.

Attributes in the <menuitem > tag have the following meaning:

- name – the name of the item
- label – the mnemonical identifier (see above) ;
- URL – the template file the item refers to;
- resource – the resource name that must exist in the account for this item to be shown in the menu;
- tip – the definition of the alternative text.

The third part includes <menudef ...></menudef> structures for all hosting plans in your system:

```

<interface>
<menudef id="SiteStudio">
  <initmenu name="acct-pref"/>
  <initmenu name="billing"/>
  <initmenu name="SiteStudio"/>
  <menuitem name="logout" label="logout.label"
    URL="design/logout.html?action=logout" resource="" tip="logout.tip"/>
</menudef>
.

```

```
.  
.   
</interface>
```

Every such structure lists the menu groups to show in this specific plan:
<initmenu /> tag defines which item group will be added to the menu.
<menuitem .../> adds menu items that don't belong to any group.

To restructure the menu, do the following:

- I. Regroup items in the <menus> part of the menu.xml file.
- II. Select groups of menu items for each hosting plan in the <interface> part of the menu.xml file.
- III. Edit menu labels as described below in the [Editing Menu Texts](#) chapter.
- IV. Login as root and restart H-Sphere.

Assigning External Links to Menu Items

H-Sphere is designed in such a way that it is not possible to put a direct URL to the external page in the menu XML description. The path in the URL attribute of the menuitem element is relative to the shiva-templates directory and would be searched by H-Sphere in one of the design subdirectories (common, replacements, and the like).

A solution may be in creating in one of the design directories a specially-formatted HTML document which would serve as a redirect to the external URL. So, the URL attribute should be set to this newly created document and thus any clicking on the menu item would redirect a user to the external link.

Let us consider the example of the menu customization where the admin user on logout would go directly to a certain page, let say, to his/her corporate site:

- 1) In ~cpanel/shiva/psoft/hsphere/menu.xml, find the element corresponding to the admin plan:

```
<menundef id="admin">
```

This would mean we are going to change CP menu only for the admin user and the change would not affect resellers and other plans.

- 2) Within this menundef element, find the line:

```
<menuitem name="logout" label="logout.label" URL="design/logout.html?action=logout" resource="" tip="logout.tip"/>
```

3) Change the URL attribute to the HTML file which would redirect to the URL you want. It is preferable to place this file to the ~cpanel/shiva/shiva-templates/common/misc directory.se. In this case, the URL attribute should be set as:

```
URL="misc/logout_redirect.html"
```

Note: Before making any change to the menu.xml file, either backup it to, for example, menu.xml.old, and/or copy and comment the line you would like to change, like this:

```
<!-- comment the old line: -->
<!-- <menuitem name="logout" label="logout.label" URL="design/logout.html&action=logout" resource="" tip="logout.tip"/> -->
<!-- the changed line: -->
<menuitem name="logout" label="logout.label" URL="misc/logout_redirect.html" resource="" tip="logout.tip"/>
```

4) In the specified directory, create the redirecting HTML file. In the example above, it would be ~cpanel/shiva/shiva-templates/common/misc/logout_redirect.html. The HTML redirecting document should be of the following format:

```
<HTML>
<HEAD>
<meta HTTP-EQUIV="refresh" CONTENT="0; URL=http://external_link">
</HEAD>
<BODY>
</BODY>
</HTML>
```

5) Restart H-Sphere.

6) Refresh browser to see the changes. It would be better to log on again.

Editing Menu Texts

The default menu texts are stored in the shiva/psoft/hsphere/lang/ directory in the menu.properties file. (H-Sphere may have several files similar to menu.properties, each of them comprising menu labels for different languages – menu_de.properties for German, menu_ru.properties for Russian,etc.).

To replace them with those of your own, do the following:

Step 1: Create the directory `shiva/custom/bundles/`. If this directory already exists, skip this step.

Step 2: In this directory, create an empty file named `menu.properties`. If you wish to customize any of the language properties files, you should also create the corresponding empty `menu_<lang>.properties` files. Even if you don't intend to change the English version, you should anyway create the empty `menu.properties` file.

Step 3. Open the file with default labels and copy the lines you want to modify into the new file making necessary changes to their content.

Important! Don't copy those default definitions you don't want to change, because your custom definitions would override any H-Sphere changes that come with the consequent updates!

Step 4: Open the `~cpanel/shiva/psoft_config/hsphere.properties` file and uncomment the line corresponding to the file you have created:

```
CUSTOM_MENU_BUNDLE = custom.bundles.menu
```

Step 5: Due to the peculiar mechanism of language bundle processing, it is essential that you create the following symlink into your custom bundle location (`shiva/custom/bundles/`): `menu_en.properties` to `menu.properties`.

Customizing menu design

This section explains how to change the appearance of the navigation menu in the H-Sphere control panel. You are expected to be familiar with the [FreeMarker](#) technology.

I. First of all, you need to go to `shiva` and create `custom/` directory. You may already have such directory.

II. Inside `custom/` directory, create the following directories:

- `templates/`
- `images/`
- `bundles/`

You may already have them.

* **Note:** If `images` directory is created, there should be a symlink to this directory from the Apache document root (usually `~cpanel/shiva/shiva-templates`); if not, users should place their images into the `IMAGES` directory.

III. Put your images into the `images/` directory and your lang/text bundles into the `bundles/` directory.

IV. In the `templates/` directory, create the `common/` directory and copy the `menu.fn` file from the `shiva-templates/common/` directory into it.

The following Freemarker functions are used in `menu.fn` to draw the navigation menu:

```
- <function draw_menu(activeItem)>;  
- <function draw_sub(item, level)>;  
- <function draw_item(item, level)>;  
- <function draw_blank_menu()>.
```

These functions draw different elements of the control panel menu. If you change them please note that the HTML tags you add must be well ordered and valid. For example, you have to make sure that the number of columns in the menu table, which is set in `draw_menu`, is the same in all these functions.

(a) The `draw_menu` function calls the other mentioned functions to draw the menu and also defines the menu table as follows:

```
<TABLE WIDTH="100%" BORDER="0" CELLPADDING="0" CELLSPACING="0">.
```

(b) The `draw_sub` function draws the menu item which is the node for the submenu (group name):

The `menu_was_drawn` variable is used to figure out how to show the next item.

(c) The `draw_item` function draws the menu item which does not have a submenu. It may be a second-level item:

Or, it could be even a first-level item that does not fall into any menu group:

(d) The *draw_sub_items* function checks the type of the menu item and calls functions (b) or (c):

If you don't want to use a standard H-Sphere image, change the following calls:

```
<call draw_image("standard-image-mnemonic-id")>
```

with:

</TD>

where *path_to_your_image* could be set either as the /IMAGES URL relative to the Apache document root, or as any absolute URL to your image, like <http://www.yourdomain.com/replacing/image/dir/>.

VI. Open the `shiva/psoft_config/hsphere.properties` file, uncomment and correct (if necessary) the following lines:

```
USER_TEMPLATE_PATH = <full_path_to_shiva>/custom/templates/  
CUSTOM_MENU_BUNDLE = custom.bundles.menu
```

where `<full_path_to_shiva>` is the physical path to the `shiva` directory (e.g. `/hsphere/local/home/cpanel/shiva`).

Note: Don't initialize the variables you are not using!

VII. Login as root and restart H-Sphere.

Related Docs: • [Advanced Interface Customization](#) • [Customizing System Emails](#) • [Providing Multilingual Support](#)

Providing Multilingual Support

Related Docs: • [Advanced Interface Customization](#) • [Customizing Control Panel Menu](#) • [Customizing System Emails](#) • [Updating Translation of H-Sphere](#)

- [Adding a New Language to the H-Sphere Interface](#)
- [Changing the Language of the Context Help](#)
- [Resource Bundle Lookup Sequence](#)
- [Supporting Input Localization for PostgreSQL](#)

Throughout this document we will often relate to the `shiva/` directory, which is located in `/hsphere/local/home/cpanel/`. Hereafter, we will refer to it as `shiva/`.

Adding a New Language to the H–Sphere Interface

To add a new language to the H–Sphere interface, you have to create a set of custom resource bundles and configure the system to use them instead of the defaults. This is achieved by the following steps:

Step 1: In `shiva/`, create the directory `custom/`. If this directory already exists, skip this step.

Step 2: In the `shiva/custom/` directory, create the directory `bundles/`. This directory may already exist.

Step 3: Next, you need to declare the custom files. In the `hsphere.properties` file, uncomment the line that corresponds to the file you have created:

```
CUSTOM_TEMPLATE_BUNDLE = custom.bundles.hsphere_lang
CUSTOM_MENU_BUNDLE = custom.bundles.menu
CUSTOM_USER_BUNDLE = custom.bundles.messages
```

Step 4: If you don't intend to change the standard H–Sphere interface labels, go to the directory `shiva/custom/bundles/` and create empty custom files with the following names:

```
hsphere_lang.properties
menu.properties
messages.properties
```

Otherwise, do the following:

- 1) create these three custom files as described above;
- 2) in the standard `lang.properties` files of the `shiva/psoft/hsphere/lang/` directory, find the labels you would like to edit;
- 3) copy–paste these labels into the corresponding new custom files.
- 4) edit the labels in the custom files. They will replace those that come with H–Sphere.

Step 5: To ensure that the new language is available to choose from the interface, add a label for your language (ID and definition) into the `shiva/custom/bundles/hsphere_lang.properties` file, following the pattern:

```
misc.langs.<XYZ>lang = <LANGUAGE> (<ENCODING>)
```

Consider this example for the Russian language:

```
misc.langs.ruslang = Russian (Win-1251)
```

Memorize the ID (e.g. *misc.langs.ruslang*), as you will need it for further steps.

Step 6: Copy all English language files from *shiva/psoft/hsphere/lang/* into the *shiva/custom/bundles/* directory changing each filename according to the following pattern:

```
hsphere_lang.properties -> hsphere_lang_<language>_<COUNTRY>.properties  
menu.properties -> menu_<language>_<COUNTRY>.properties  
messages.properties -> messages_<language>_<COUNTRY>.properties
```

Consider the following examples for the Russian language:

```
hsphere_lang_ru_RU.properties  
menu_ru_RU.properties  
messages_ru_RU.properties
```

Sometimes, you might want to add *_<ENCODING>* to the end of the filename to support different encodings for your language, for instance: *hsphere_lang_ru_RU_CP1251.properties*

The country, language and encoding parameters must correspond to the ISO standards.

Step 7: Translate interface texts stored in these files using the target encoding. If a label or message has a variable in curly brackets, e.g. {0}, single quotes and apostrophes (') must be replaced with two single quotes ("). For example, in English we write:

```
search.view_invoice = View Client's Invoice  
but  
billing.del_no = No, I don't want to delete {0}
```

Step 8: In the *hsphere.properties* file, find the following line:

```
LANG_LIST = en_US_ISO8859_1|ISO-8859-1:misc.langs.english ru_RU_CP1  
251|windows-1251:misc.langs.ruslang
```

This line contains definitions for the languages that come with H-Sphere, each including the following components:

```
<language>_<COUNTRY>_<ENCODING>|<html_encoding>;misc.langs.<XYZ>lang;
```

To the end of this line, add a definition for your custom language following the existing examples – using space as the delimiter.

Step 9: In the `hsphere.properties` file, set system locale and encoding to custom values. The locale should correspond to the Java ISO standards, the encoding must correspond to the browser standards. For example, settings for the Russian language will look as follows:

```
# Override system locale
LOCALE = ru_RU

# Encoding
ENCODING = Windows-1251
DB_ENCODING=windows-1251
```

The `LOCALE` value will affect not only the interface language, but also the currency, date, time, days of the week and other locale settings.

From the control panel navigation menu, users can choose the interface language that will override the language set by the reseller and the hosting provider. Users' language settings have limited life span that is determined by the age of cookies in users' browsers. After cookies expire, the user's CP language will switch back to that of the provider / reseller. You can set the age of these cookies in the `hsphere.properties` file:

```
#Language cookies maximum age (in seconds)
#Default value = 1 year
COOKIE_AGE = 31536000
```

Step 10: Log in as root and restart H-Sphere.

These steps must suffice to incorporate the possibility of changing the regional settings and the interface language of H-Sphere. To have your translation incorporated into the future versions of H-Sphere, send the translated files to support@psoft.net.

Changing the Language of Online Help

Online help texts take a somewhat different approach. Carefully follow all the procedure described in the [Customizing Templates](#) section – copying the online help files from the `shiva-templates/common/online_help/` folder to the `shiva/custom/templates/` directory. On step 3, translate the heading and the body of each help file in the corresponding encoding.

In addition to the steps described in the Customizing Templates section, change the name of each online help file in the custom-templates directory by adding the language, country and encoding to the name base, according to the pattern:

```
<base>.oh -> <base>_<language>_<COUNTRY>_<ENCODING>.oh
```

Consider the following example for the Russian language:

```
01balance.oh -> 01balance_ru_RU_Cp12.oh
```

Alternatively, you can omit encoding and country (or only encoding) in the file name:

```
<base>.oh -> <base>_<language>_<COUNTRY>.oh or  
<base>.oh -> <base>_<language>.oh
```

In case of the Russian language, the file name will look as follows:

```
01balance.oh -> 01balance_ru_RU.oh or  
01balance.oh -> 01balance_ru.oh
```

Mind, however, that omitting country and encoding in the file name may cause some texts to display incorrectly in some browsers.

Lookup Sequence

The resource bundle lookup searches for language and regional settings in the following order:

1. USER_TEMPLATE dir, file <_name>_<Lang>_<country>_<code_page>.<file_ext>;
2. DEFAULT_TEMPLATES, file <_name>_<Lang>_<country>_<code_page>.<file_ext>;
3. USER_TEMPLATE dir, file <file_name>_<Lang>_<country>.<file_ext>;
4. DEFAULT_TEMPLATES, file <file_name>_<Lang>_<country>.<file_ext>;
5. USER_TEMPLATE dir, file <file_name>_<Lang>.<file_ext>;
6. DEFAULT_TEMPLATES, file <file_name>_<Lang>.<file_ext>;
7. USER_TEMPLATE dir, file <file_name>.<file_ext>;
8. DEFAULT_TEMPLATES, file <file_name>.<file_ext>;
9. If no appropriate settings were found, the user will get a corresponding message and a possibility to send a trouble ticket.

Important:

Changing the language of the interface doesn't make the system handle non-English characters properly. Therefore, those who add new interface languages should ensure that the administrators and customers do not enter non-English symbols in the forms of the H-Sphere control panel.

Supporting Input Localization for PostgreSQL

(version 2.09 and higher)

To set up a custom language support (e.g. Russian) when entering data into PostgreSQL, go through the following steps:

1. Recompile PostgreSQL using the following keys:
 - enable-locale (enable locale support)
 - enable-recode (enable cyrillic recode support)
 - with-mb=WIN (enable multi-byte support, e.g. WIN)
2. Create H-Sphere database supporting the new encoding (e.g. WIN).

NOTE: if the browser encoding does not agree with the database encoding, it is impossible to guarantee a correct record in the database.

In the `~cpanel/shiva/psoft_config/hsphere.properties` configuration file, replace

```
DB_URL = jdbc:postgresql://127.0.0.1/hsphere
```

with

```
DB_URL = jdbc:postgresql://127.0.0.1/hsphere?charSet=<YOUR_LANGUAGE_ENCODING>
```

For instance, Russian language support takes the following line:

```
DB_URL = jdbc:postgresql://127.0.0.1/hsphere?charSet=WIN
```

Related Docs: • [Advanced Interface Customization](#) • [Customizing Control Panel Menu](#) • [Customizing System Emails](#) • [Updating Translation of H-Sphere](#)

Updating Translation of H-Sphere Interface

(version 2.08 and higher)

With every update of H-Sphere, your translation of the control panel interface becomes outdated, and you need to update your translated menu labels and messages. H-Sphere comes with a script that finds differences between two files – the new English file and its old translated version. This script, **diff_labels.pl**, is located in the directory **/hsphere/lang/**.

To compare two files, do the following:

Step 1. Login to your CP server under "cpanel:

```
su - cpanel
```

Step 2. Go to the 'langs' directory, where your translations are stored:

```
cd shiva/psoft/hsphere/lang/
```

Step 3. Execute the following command:

```
./diff_labels.pl hsphere_lang.properties hsphere_lang_x.properties > en-it.diff
```

* where `hsphere_lang_x.properties` is the file of your locale translation. It will write all the differences between these two files into the `en-it.diff` file. When you get this difference list, manually add the labels into `hsphere_lang_x.properties`.

Step 4. Add translations into your locale translation file.

Customizing User Signup Forms

This document explains how to replace user signup (order) forms with those of your own. Before you begin the signup customization, please note the following:

- The default signup forms contain validation scripts. It is recommended that your custom signup forms also provide a client side validation mechanism.
- When H-Sphere server-side validation rejects user data, the user is redirected to the error page generated based on the template `~cpanel/shiva/shiva-templates/common/signup/end.html` which has the look and feel of the standard H-Sphere interface and links to the STANDARD H-Sphere signup forms. Normally, you would want to customize this template to ensure that (i) it has the look and feel of your custom signup forms, and (ii) it gives a way to go back to the signup forms, then modify and re-submit the signup data. The template has been written in FreeMarker 1.5.1, and in order to make changes to its code, please become familiar with the FreeMarker technology version 1.5.1, the documentation available at <http://freemarker.org>. The way you customize the page will totally depend on how you organize your signup forms.
- Custom signup fields must match those in the default signup. If more fields are added in newer versions, you will need to update your custom forms.

Your signup script has to put the collected data into the html fields below and submit them to the following URL:

```
<form name="login" action="psoft.hsphere.CP" method="POST">
```

or

```
<form name="login" action="protocol://cp.domain.name:PORT/psoft/servlet/psoft.hsphere.CP" method="POST">. For example:
```

```
<form name="login" action="http://www.psoft.net:8080/psoft/servlet/psoft.hsphere.CP" method="POST">
```

Note: `psoft.hsphere.CP` is case sensitive!

Some signup texts can be customized through the control panel from *Look And Feel* -> *Signup Texts*. If you have customized texts through the control panel, they will override the texts in your custom signup forms, so you may need to remove them.

Basic (service fields signup depends on)

Field name	Possible Values	Explanation
<code>_eul_accept</code>	"1"	

		Accept terms of End User License Agreement
_mod	"" "signup" – transfer domain, "opensrs" OpenSRS registration, "nodomain" – signup without domain, "3ldomain" – third level domain, "service" – service domain	Signup modifier
action	"signup"	Service parameter
plan_id	numeric	Number of the plan for signup
signup	"yes"	Service parameter
login	alphanumeric	user login
password	alphanumeric	user password
password2	alphanumeric	Confirm user password
template_name	submit/signup/end.sbm (submit/signup/end_osrs.sbm for OpenSRS registration)	Processing template
admin_signup	"yes" – if we sign user up from the admin panel	Service parameter

Contact Info (User's contact information)

Field name	Possible Values	Explanation
_ci_first_name	alphanumeric	User's first name
_ci_last_name	alphanumeric	User's last name
_ci_address1	alphanumeric	User's address 1

_ci_address2	alphanumeric	User's address 2
_ci_city	alphanumeric	User's city of residence
_ci_company	alphanumeric	User's company name
_ci_country	alphanumeric	User's country code
_ci_email	alphanumeric	User's contact e-mail address
_ci_phone	numeric	User's phone number
_ci_postal_code	numeric	User's zip code
_ci_state	e.g.: "NY"; "NA" for non US or Canada residents.	User's state code. In the custom form for non US or Canada residents, you should add this field as hidden.
_ci_state2	alphanumeric	User's state or province for non US and Canada residents. Should be present in the custom form only in case <code>_ci_state= 'NA'</code> .

Billing Info (not used for Trial registration)

Field name	Possible Values	Explanation
_bi_first_name	alphanumeric	User's first name
_bi_last_name	alphanumeric	User's last name
_bi_address1	alphanumeric	User's address 1
_bi_address2	alphanumeric	User's address 2
_bi_city	alphanumeric	User's city of residence
_bi_company	alphanumeric	User's company name
_bi_country	alphanumeric	User's country code

_bi_email	alphanumeric	User's contact e-mail address
_bi_phone	numeric	User's phone number
_bi_postal_code	numeric	User's zip code
_bi_state	e.g.: "NY"; "NA" for non US or Canada residents.	User's state code. In the custom form for non US or Canada residents, you should add this field as hidden.
_bi_state2	alphanumeric	User's state or province for non US or Canada residents. Should be present in the custom form only in case <code>_bi_state='NA'</code> .
_bi_type	"CC" – credit card, "CHECK" – check, "TRIAL"	Payment type

Only Credit Card section

Field name	Possible Values	Explanation
_bi_cc_name	alphanumeric	Credit card name
_bi_cc_number	numeric	Credit Card number
_bi_cc_type	strings available in the Merchant Gateway Manager: "VISA", "MC", etc.	Credit Card type
_bi_cc_exp_month	two digits	the month of Credit Card expiry date
_bi_cc_exp_year	four digits	the year of Credit Card expiry date

Note: below are the fields for *Solo/Switch debit cards* used in some countries:

Field name	Possible Values	Explanation
-------------------	------------------------	--------------------

_bi_cc_issues_no	alphanumeric	Issue number
_bi_cc_start_month	two digits	Card Start Month
_bi_cc_start_year	four digits	Card Start Year

Billing period

Field name	Possible Values	Explanation
_bp	0 or a positive integer	Sequence number of the billing period in the list of billing periods for the selected plan. To see the list of the billing periods, go to your control panel, click the <i>Settings</i> link for this plan and scroll down to the <i>Billing configuration</i> section.

Domains

Field name	Possible Values	Explanation
type_domain	"transfer_new_misc_domain" – transfer domain or register non-OpenSRS "without_domain" – register stopgap domain "3ldomain" – third level domain "service_domain" – service domain "parked_domain" – parked domain "new_opensrs_domain" – OpenSRS domain "domain_name" – domain name	Type of new domain

OpenSRS registration

Field name	Possible Values	Explanation
<code>period</code>	numeric	OpenSRS periods (years)
<code>_srs_owner_first_name</code>	alphanumeric	User's first name
<code>_srs_owner_last_name</code>	alphanumeric	User's last name
<code>_srs_owner_address1</code>	alphanumeric	User's address 1
<code>_srs_owner_address2</code>	alphanumeric	User's address 2
<code>_srs_owner_city</code>	alphanumeric	User's city of residence
<code>_srs_owner_org_name</code>	alphanumeric	User's company name
<code>_srs_owner_country</code>	alphanumeric	User's country code
<code>_srs_owner_email</code>	alphanumeric	User's contact e-mail address
<code>_srs_owner_phone</code>	numeric	User's phone number
<code>_srs_owner_postal_code</code>	numeric	User's zip code
<code>_srs_owner_state</code>	e.g.: "NY"; "NA" for non US or Canada residents.	User's state code. In the custom form for non US or Canada residents, you should add this field as hidden.
<code>_srs_owner_state2</code>	alphanumeric	User's state or province for non US or Canada residents. Should be present in the custom form only in case <code>_srs_owner_state='NA'</code> .

Contact Info

Field name	Possible Values	Explanation
<code>_srs_billing_first_name</code>	alphanumeric	User's first name

<code>_srs_billing_last_name</code>	alphanumeric	User's last name
<code>_srs_billing_address1</code>	alphanumeric	User's address 1
<code>_srs_billing_address2</code>	alphanumeric	User's address 2
<code>_srs_billing_city</code>	alphanumeric	User's city of residence
<code>_srs_billing_org_name</code>	alphanumeric	User's company name
<code>_srs_billing_country</code>	alphanumeric	User's country code
<code>_srs_billing_email</code>	alphanumeric	User's contact e-mail address
<code>_srs_billing_phone</code>	numeric	User's phone number
<code>_srs_billing_postal_code</code>	numeric	User's zip code
<code>_srs_billing_state</code>	e.g.: "NY"; "NA" for non US or Canada residents.	User's state code. In the custom form for non US or Canada residents, you should add this field as hidden.
<code>_srs_billing_state2</code>	alphanumeric	User's state or province for non US or Canada residents. Should be present in the custom form only in case <code>_srs_billing_state= 'NA'</code> .

Compiling Templates With Client-Side Form Validation

Related Docs: • [Advanced Interface Customization](#)

This document provides step-by-step instructions on how to create or modify control panel templates with the client-side validation of HTML form input fields. Such templates represented in the .html.in format require pre-compilation to generate HTML source, unlike .html templates that are used with server-side form validation and don't need to be pre-compiled.

A designer should have root access to the H-Sphere server and log in under the "cpanel" superuser:

```
# su - cpanel
```

To implement customization correctly, all template files and directories should have cpanel:cpanel ownership, and the make directive which is performed to rebuild templates should be run ONLY under the cpanel user.

1. Enter paths to JFlex.jar and java_cup.jar into CLASSPATH (bash_profile).
2. Enable write permission to H-Sphere's root directory.
3. Make sure that the make procedure in psoft/ has been done ('make shiva').
4. Check the following variables in psoft_config/hsphere.properties:

```
TEMPLATE_PATH = ... - a physical location of templates ; JS = ; IMAGES = .
```

By default JS(JavaScripts) and IMAGES are blank. It means that javascripts and images are placed inside each design directory. Yet, this can be changed if necessary. It is possible to create symlinks in the Document Root (TEMPLATE_PATH) directory as follows :

```
javascript -> /home/hsphere/javascript or images -> /home/hsphere/images.
```

Then it is necessary to add names of symlinks to the JS and IMAGES variables (JS=javascript or/and IMAGES=images).

5. Go to the templates directory (DocumentRoot). Check and edit variables in the *configure file:
 - (1) SHIVA_ROOT – physical location of the 'HSphere' root;
 - (2) HSPHERE_PROPERTIES – path to the hsphere.properties file;
 - (3) CFG_FILE – name of the config file for each design (design.cfg by default).
6. If necessary, you may configure design.cfg files in each design.
7. Run ./configure in the templates directory. This will create Makefiles for all of designs. There is a parameter '*clean*' which can help you clear all the created Makefiles.

You may need to install the GNU make utility and/or a java compiler for *configure to work correctly.
8. Run *make* (or *make all*). Html, js and java class files will be created. Aside from that, it is possible to run '*make all*' in each design directory.

The '*make clean*' command clears all of the created files in the nested directories.

Related Docs: • [Advanced Interface Customization](#)