H-Sphere Customization Guide

PDF document generated: 04 Jan 2008

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

Introduction To Customization And Packaging

Basic interface settings can be configured through the control panel. The **Look and Feel** menu allows to set *skins and colors*, *images and icons*, and *some interface texts*.

Advanced interface customization is what goes beyond the scope of the Control Panel settings. It is performed on the H-Sphere CP server by designers and programmers with administrative rights, in order to create or modify H-Sphere interface elements.

Packaging allows to create installable plugins, or packages. This is a way to share custom elements between H-Sphere installations. Third parties can use it to develop and distribute plugins that add new or extend/override standard H-Sphere functionalities. Documentation on building packages is introduced in *H-Sphere Developer Guide*. .

Customizable Elements	Description	Customization	Packaging
Templates	Design and control patterns for dynamic HTML generation. They are to be modified if you need to restructure the layout of certain pages of the Control Panel interface, or to change the look of the Control Panel header and footer.	• Template Customization	*Template Packages*
System E-Mail Notifications	A special type of templates used to generate standard email notifications sent by H-Sphere.	• System E-Mail Customization (Templates) • *Editing System E-Mails in CP*	N/A
Context Help	A special type of templates to generate context help for certain elements of the Control Panel interface.	• Context Help • Changing Context Help Language	*Template Packages*
Interface Texts	Standard messages and labels that appear on the interface pages are placed in the special configuration files and may be set for different languages.	• Interface Text Customization • Compiling Language Bundles	*Custom Packages*
H-Sphere Languages	Adding new languages to the interface and modifying language files with interface texts in different languages.	• Adding New Languages • Compiling Language	*Custom Packages*

		Bundles	
Menu	Generating and modifying control panel menus and submenus and adding external links to the menu.	• Menu Customization • XML Merge Customization	*XML Customization With Packages*
Skins And Icon Sets (Designs)	H-Sphere interface design has a broader meaning than just configuration of certain color schemes and the corresponding icon sets, what is called the skin. It also determines the set of skins available for this design, specifies the sets of icons in the Quick Access page and enables to override the standard settings with the custom ones.	• Skin And Icon Set Customization • XML Merge Customization	*XML Customization With Packages*
CP Crons	H-Sphere utilities regularily executed on the Control Panel server.	• *Custom CP Crons* • XML Merge Customization	*XML Customization With Packages*
Plan Wizards	Custom plan wizards defined and configured in XML documents.	• *Creating Custom XML Plan Wizards* • XML Merge Customization	*XML Customization With Packages*
Merchant Gateways	The media for making real-time payments with online credit card processing centers automatically from the CP.	• *Adding Custom Merchant Gateways*	*Custom Packages*
Web Payment Systems	The media for making payments manually from the web interface of the payment systems.	• *Adding Custom Web Payment Systems*	*Custom Packages*
Signup Forms	Generating custom signup forms to sign up users aside from the standard signup procedure provided in H-Sphere, as well as modifying the standard signup pages.	• User Signup Customization	*Custom Packages*

WARNING:

1. Advanced customization may produce unpredictable results after updating H-Sphere, since updates affect the template structure and the page generation.
2. Advanced customization performed by H-Sphere customers is done at their own risk and is not supported by the PSoft team.
3. You may order customization from the PSoft team. Such customization is supported hereafter. Please contact sales@psoft.net for pricing details.

Understanding H-Sphere Templates

Related Docs: · Template Customization

What Are Templates

H-Sphere templates is basically what lays behind the H-Sphere Control Panel Web interface. For the most part, templates are written using Freemarker Java processing language for dynamic content generation. Currently, version 1.7.5 of Freemarker is used.

Location of Templates

1) Template root directory

H-Sphere template root directory is by default `/hsphere/local/home/cpanel/shiva/shiva-templates`. It is set by the `TEMPLATE_PATH` parameter in the `~cpanel/shiva/psoft_config/hsphere.properties` file:

```
TEMPLATE_PATH = /hsphere/local/home/cpanel/shiva/shiva-templates/
```

2) Default (common) design directory

Design, or skin, is the Control Panel GUI representation. Each design is defined by its own sets of templates, images, CSS styles, JavaScripts, etc. The most important part of GUI, templates, are placed in separate subdirectories of the template root directory in accordance with a design they belong to. The special common subdirectory is used to store templates that are the same for different designs. Also, this directory contains templates for the Left Menu design which is the default H-Sphere design. The path to this directory, relative to `TEMPLATE_PATH`, is set in hsphere.properties:

```
DEFAULT_TEMPLATES = common/
```

Important: Common template directory must always exist!

3) Custom template directory

Custom template directory is usually `/hsphere/local/home/cpanel/shiva/custom/templates`. Its location is set in hsphere.properties:

```
USER_TEMPLATE_PATH=/hsphere/local/home/cpanel/shiva/custom/templates/
```

Custom templates directory structure should correspond with the default template root directory (`shiva-templates`) tree. However, you should be aware that templates in the custom template directory override the corresponding templates in the default template directory, thus all modifications and new features in existing default templates coming with new H-Sphere releases would be also overriden by custom templates. Therefore, only customized templates should be placed to your custom template directory.

System E-Mail Notification Templates

System email notifications templates are used to generate standard H-Sphere e-mail messages sent to customers or to admins on certain events related to account management, billing, and the like. It is made possible to edit these messages directly from Control Panel, in plain text or HTML and for each of the available languages, without the need of customizing the default templates.

E-mail templates are located in the `~cpanel/shiva/shiva-templates/common/mail` directory and have `.txt` extension. See the list of the system e-mail templates.

Skeletons

Skeleton templates, or skeletons, are special templates designed to generate user default sites for newly created domains in corresponding domain subdirectories of user home directories. Skeleton templates are written in HTML (without Freemarker instructions) and located in the `/hsphere/shared/skel` directory.

Web Interface Templates

Templates for generating Control Panel interface pages are of the following types: main templates, control templates, submit templates, function templates, and some special purpose templates. There are also templates for generating context help pages.

Main Templates

Main, or basic, templates are templates for generating the entire Web page in CP. The code of a main template represents a framework that contains calls of functions for generation of the page header, menu and footer, and includes control templates for processing forms.

Main templates are `.html` files located in the `~cpanel/shiva/shiva-templates/<design>/` directories for each design.

Control Templates

Contol templates, or controls, are responsible for generation and management of forms in the working area of H-Sphere interface. They represent the part of HTML code included in the main templates.

Control templates are with or without form field validation mechanism implemented:

- `.html.in` templates provide client-side form validation. They need to be compiled to apply changes made in them. The corresponding `.html` templates are generated as the result of compilation of `.html.in` templates of the same name. Thus, if there is a pair of `.html` and `.html.in` templates with the same name, it is recommended to modify the .html.in template and then to recompile it. Read more about compiling templates with client-side validation.
- No field validation mechanism is implemented in `.html` templates that do not have the initial `.html.in` templates of the same name. Changes in `.html` templates take effect immediately.

Control templates are located in the `~cpanel/shiva/shiva-templates/<design>/control` directories for each group of main templates.

Control templates assign submit templates that do not have visual HTML representation and serve solely to process form submits.

Submit Templates

Submit templates do not have visual representation. They contain instructions to be performed upon the form submit. These templates provide server-side validation of submitted data and scenarios of subsequent actions if submit is successful or if an error occurs. Submit template files have `.sbm` extension.

Function Templates

These templates contain collections of functions (or macros) used in other templates, for example, for drawing menu, footer and header.

- `~cpanel/shiva/shiva-templates/common/functions` - generic macro collection, Does not depend on designs.
- `~cpanel/shiva/shiva-templates/<design>/menu.fn` - functions for drawing menu for a particular design.
- `~cpanel/shiva/shiva-templates/<design>/design.fn` - functions for drawing interface elements for a particular design (implemented for common and XPressia/XPressia Lite designs)
- `~cpanel/shiva/shiva-templates/<design>/extra.fn` - extra functions.

- `~cpanel/shiva/shiva-templates/common/control/signup_function.html` - functions for signup templates.

Templates For Special Purposes

There are some Web interface templates that do not fall into any of the above mentioned categories. They are designed for special tasks such as to draw a menu on the left, or the page header or footer, or login page, etc. Some special purpose templates are located in the `~cpanel/shiva/shiva-templates/<design>/design/` directory, some like `signup_top.html.in` or `signup_bottom.html` in the `~cpanel/shiva/shiva-templates/<design>/signup` directory. There is no general classification for such templates.

Context Help Templates

Context help templates are special templates for generating online help message in popup windows. Each context help template has its topic header and body. They can be modified as usual H-Sphere templates.

Online help files are located in the `~cpanel/shiva/shiva-templates/common/online_help` directory. They have `.oh` extension and contain the text in HTML format. See the instructions how to add context help pages to H-Sphere interface. Also read about context help in different languages.

Designs

Design, or skin, is the Control Panel GUI representation. It provides a different look of menu (left menu or dropdown menu on the top, or no menu present at all), CSS styles, colors and images, and the Quick Access page with icon links to different CP pages.

These are basic H-Sphere designs whose templates are located in the corresponding design template directories of `~cpanel/shiva/shiva-templates` (referred to as `<design>` in the document):

- `common` - the left-menu design (Left Menu in CP). All core templates are made for this design scheme. Other templates that do not depend on design, inlcuding online help templates and system e-mail notification templates, are also located there.
- `nomenu` - the design with no left menu (No Menu in CP). It is turned on as the default user design after the H-Sphere installation.

- `text-based` is the alternative look of the No Menu design (Text-Based in CP) where only captions with no icons are provided in the Quick Access menu page.
- `xcp` - the XPressia design with dropdown menus, extensive use of CSS styles and other advancements.
- `xcpl` - the XPressia Lite design, a simpler and faster implementation of XPressia.

If a certain template is not found for a particular design, H-Sphere gets that template in the `common` directory.

The default design configuration file `design_config.xml` is located in the `~cpanel/shiva/psoft/hsphere/` directory.

Replacements

Replacements are templates that override basic templates for particular plans. Replacements' root directory for each design is the `~cpanel/shiva/shiva-templates/<design>/replacements` directory. Replacements are located in separate subdirectories specified in plan settings as the Template Directory parameter, relative to the `replacement` directory.

H-Sphere first searches for a template in the `<design>/replacements/<plan>` directory which has the same structure as the `<design>` directory. If the template is not found, it starts to look for it in the `<design>` directory. Read more about template lookup sequence.

Template Directory Structure

- `~cpanel/shiva/shiva-templates/<design>` - template directory for one of the H-Sphere basic designs.
- `~cpanel/shiva/shiva-templates/common/JS` - JavaScript functions (used for all designs)
- `~cpanel/shiva/shiva-templates/<design>/design` - contains templates for special purposes such as login page, password reminder page, header and footer templates and the like.
- `~cpanel/shiva/shiva-templates/<design>/CSS` - CSS styles for a design.
- `~cpanel/shiva/shiva-templates/<design>/<subdir>` - main templates are placed into separate subdirectories, according to their tasks, e.g., `admin`, `billing`, `MSSQL`, etc.
- `~cpanel/shiva/shiva-templates/<design>/control/<subdir>` - corresponding control templates.
- `~cpanel/shiva/shiva-templates/<design>/submit/<subdir>` - corresponding submit templates.
- `~cpanel/shiva/shiva-templates/<design>/replacements/` - template replacements for different types of plans.

- `~cpanel/shiva/shiva-templates/<design>/replacements/<plan>/<subdir>` - template replacements overriding templates in corresponding subdirectories for that design. directory.
- `~cpanel/shiva/shiva-templates/<design>/replacements/control/<resources>` - control templates for corresponding replacements.
- `~cpanel/shiva/shiva-templates/<design>/replacements/submit/<resources>` - submit templates for corresponding replacements.

Template Lookup Sequence

H-Sphere searches for a template of a particular design first in the custom template directory in replacements, then, if the template is not found there, it proceeds to the corresponding default template directory:

1. `~cpanel/shiva/custom/templates/<design>/replacements/`
2. `~cpanel/shiva/shiva-templates/<design>/replacements/`

If the template is not found in replacements, H-Sphere searches in the design directory, first in among the custom templates, then amonng the corresponding default templates:

1. `~cpanel/shiva/custom/templates/<design>/`
2. `~cpanel/shiva/shiva-templates/<design>/`

If the template is not found for this design, the search continues in the same sequence in the common design template directory:

1. `~cpanel/shiva/custom/templates/common/replacements/`
2. `~cpanel/shiva/shiva-templates/common/replacements/`
3. `~cpanel/shiva/custom/templates/common/`
4. `~cpanel/shiva/shiva-templates/common/`

Related Docs: · Template Customization

Template Customization

Related Docs: · Understanding Templates · Interface Controls and Colors · Skin and Icon Set Customization · Compiling Templates

This document will guide you through the generic step-by-step instruction on customizing templates. This implies you are already familiar with the concept of templates in H-Sphere.

It is possible to create and install packages of templates. Read more about *template packages* in Developer Guide.

Pre-Cautions

1. Advanced customization may produce unpredictable results after updating H-Sphere, since updates affect the template structure and the page generation.

2. Advanced customization performed by H-Sphere customers is done at their own risk and is not supported by the PSoft team.

3. You may order customization from the PSoft team. Such customization is supported hereafter. Please contact sales@psoft.net for pricing details.

4. **Template customization affects ALL H-Sphere accounts, regardless of their plans!**
In terms of H-Sphere customization, only two types of accounts are customized, regardless of plans: admin accounts which are H-Sphere administrative accounts, and user accounts - all other accounts. Reseller accounts are regarded as user accounts, except for the reseller administrative account which relates to the admin account type.

Pre-Requisites

- Before you do any customization, log into CP server under root as the *cpanel user*.

  Notes:

1. Template Ownership.
Make sure templates have the **cpanel:cpanel** ownership. Mind, however, that since H-Sphere 2.5 RC 3 images, CSS and JavaScript files and directories have cpanel:httpdcp ownership and you must not change their ownership to cpanel:cpanel. *H-Sphere 2.5+ updater* checks and automatically sets correct ownership and permissions on respective default and custom files and directories (this does not refer to H-Sphere packages). **We don't recommend changing manually the ownership and permissions of default templates!**

2. The `make` directive which is performed to rebuild `*.html` templates should be run ONLY under cpanel.

- Do not use **whitespaces** in the template filenames!
- 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, follow the step-by-step instructions specified below.

Step-By-Step Template Customization Procedure

Step 1. On the CP server, log in as *cpanel user*.

Step 2. In the `~cpanel/shiva/` directory, create the custom template directory `custom/templates/` if it doesn't exist.

Step 3. In the `~cpanel/shiva/psoft_config/hsphere.properties` file, find the `USER_TEMPLATE_PATH` parameter. Here, the full to your custom template directory must be specified:

```
USER_TEMPLATE_PATH=/hsphere/local/home/cpanel/shiva/custom/templates/
```

The directory name must end with a slash. Don't do anything if the directory name is already there.

WARNING:
Don't change the `TEMPLATE_PATH` variable in hsphere.properties!
`TEMPLATE_PATH` points to the default template directory. If you change it, you won't see any updates in the default templates.
Step 4. Copy the templates you would like to customize into `shiva/custom/templates/`, preserving their file paths relative to this directory. E.g., if you are going to customize the
`~cpanel/shiva/shiva-templates/path_to_template/FILE`,
copy it to

`~cpanel/shiva/custom/templates/path_to_template/FILE`.

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

WARNING:
Don't copy the **whole** 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 5. (only before 2.5!) Make sure all custom template files have correct ownership.

Step 6. Modify the templates you have copied to the `~cpanel/shiva/custom/templates/` directory.

The following documents will be helpful:

- Interface Controls and Colors
- Skin and Icon Set Customization
- Edit Interface Texts

Important:

1) We don't recommend inserting the text directly into the templates. Use text labels defined in language bundles to ensure multilingual support.

2) H-Sphere uses the **UTF-8** charset for all languages. Therefore, text directly inserted into templates (i.e., not by means of text labels defined in language bundles) **must be** in the UTF-8 encoding.

Step 7. *Restart H-Sphere*.

Related Docs: · Understanding Templates · Interface Controls and Colors · Skin and Icon Set Customization · Compiling Templates

Skeleton Customization

Related Docs: · Understanding Templates

When an end-user adds a new domain, the system generates initial web site based on skeleton templates. Skeleton templates are written in HTML. Unlike FreeMarker templates, Skeleton templates are customized right in the directory they are located.

To modify Skeleton templates:

1. Log in as *cpanel user*
2. Enter the Skeleton template directory:

   cd /hsphere/shared/skel
3. Customize the template files directly according to your needs

Please only pay attention that with each automatic upgrade of web boxes, all the customizations get lost. So, after performing the customization, backup the templates into a safe location to get them back working after performing box upgrades.

Related Docs: · Understanding Templates

Interface Controls And Colors

Related Docs: · Template Customization · Skin And Icon Set Customization

This document describes the rules for template customization:

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 approprate 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: · Template Customization · Skin And Icon Set Customization

Adding Custom Icons

Related Docs: · Template Customization · Interface Controls and Colors · Menu Customization · Skin and Icon Set (Design) Customization

To provide multiple design support, H-Sphere uses the `design_config.xml` file, which can be found in the `/hsphere/local/home/cpanel/shiva/psoft/hsphere/` directory. Its structure is explained in the Design XML Configuration guide.

The current document provides an example of how to customize this file to add your own custom icons. For this, you need to:

1. Create custom `design_config` file in custom location that will be merged with the default XML configuration file. Your design_config.xml should include only differences in relation to the original H-Sphere one.
2. As H-Sphere supports several icon sets, create an icon for each icon set and include information about this icon location into your custom `design_config.xml`. To do this:
    I. Declare the url and the image id that will be used for your icon:

```
<design_config>
<icons>
<icon id="custom_mail_icon" url_param="template_name=email/custom_email.html"
        rtype="" platform=""
        label="quick.your_custom_mail_web_app"
        tip="quick.your_custom_mail_web_app" help=""/>
</icons>
</design_config>
```

Where:
-The email/custom_email.html is the hsphere page (custom template) from which a customer will be redirected to your url.
-The quick.your_custom_mail_web_app label should be described in the customized `hsphere_lang.properties` file.

II. Specify the place where the icon will be displayed. It can be included into the Quick Access page for the Admin Account or end user Quick Access page:

```
<design_config>
...
...
<skill_icon_sets account_type="user">
    <skill_set id="advanced" label="icon_skill_set.advanced_user_l">
        <icon_group id="mail" label="icongroups.mail">
```



```
                <icon id="custom_mail_icon"/>
            </icon_group>
        </skill_set>
    <skill_set id="standard" label="icon_skill_set.standard_user_l">
        <icon_group id="mail" label="icongroups.mail">
            <icon id="custom_mail_icon"/>
        </icon_group>
    </skill_set>
    </skill_icon_sets>
    </design_config>
```

So, your icon will be included into standard and advanced **Quick Access** of the **Mail** section.

III. Describe which image will be used for each icon set. Use the custom image directory to store your new icons `/hsphere/local/home/cpanel/shiva/custom/images`:

```
<design_config>
    ...
    ...
    <icon_image_sets base_dir="/CUSTOM_IMAGES">

        <icon_image_set id="default" label="iconsets.default" dir="">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="46"
height="49"/>
        </icon_image_set>

        <icon_image_set id="blue_haze" label="iconsets.blue_haze" dir="blue_haze">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="46"
height="49"/>
        </icon_image_set>

        <icon_image_set id="cocoa" label="iconsets.cocoa" dir="pebble">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="46"
height="49"/>
        </icon_image_set>

        <icon_image_set id="marsh" label="iconsets.marsh" dir="marsh">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="46"
height="49"/>
        </icon_image_set>
```



```
         <icon_image_set id="wooden" label="quick.iconsets.wooden"
dir="ICONS/wooden">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="48"
height="48"/>
        </icon_image_set>

        <icon_image_set id="square_set" label="quick.iconsets.square_set"
dir="ICONS/square_set">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="48"
height="48"/>
        </icon_image_set>

        <icon_image_set id="cartoon_set" label="quick.iconsets.cartoon_set"
dir="ICONS/cartoon_set">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="48"
height="48"/>
        </icon_image_set>

        <icon_image_set id="bubble_set" label="quick.iconsets.bubble_set"
dir="ICONS/bubble_set">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="48"
height="48"/>
        </icon_image_set>

        <icon_image_set id="xcp1_set" label="quick.iconsets.xcp1_set"
dir="ICONS/xcp1">
        <image id="custom_mail_icon" file="custom_mail_icon.png" width="48"
height="48"/>
        </icon_image_set>

        <icon_image_set id="xcpl_set" label="quick.iconsets.xcpl_set"
dir="ICONS/xcp1">
        <image id="custom_mail_icon" file="custom_mail_icon.gif" width="48"
height="48"/>
        </icon_image_set>

    </icon_image_sets>
</design_config>
```

3. Create images for each icon set and place them into the corresponding directories. The base directory for your images is `/hsphere/local/home/cpanel/shiva/custom/images`

Related Docs: · Template Customization · Interface Controls and Colors · Menu Customization · Skin and Icon Set (Design) Customization

Skin And Icon Set Customization

Related Docs: · Template Customization · Interface Controls and Colors · Menu Customization · Adding Custom Icons

H-Sphere interface design has a broader meaning than just configuration of certain color schemes and the corresponding icon sets, what is called the skin. It also determines the set of skins available for this design, specifies the sets of icons in the **Quick Access** page and enables to override the standard settings with the custom ones.

To provide multiple design support, H-Sphere uses the `design_config.xml` file, which can be found in the `/hsphere/local/home/cpanel/shiva/psoft/hsphere/` directory. Its structure is explained in the Design XML Configuration guide.

The current document shows you how to customize this file to add your own designs, color schemes, colors and images. For this, you need to:

1. customize `design_config.xml`
2. implement custom design templates

1. Design XML Customization

A designer should have access to the H-Sphere server as the "cpanel" user. To implement customization correctly, all template files and directories should have cpanel:cpanel ownership.

Important: Since 2.5 RC 3 and up, templates, images, CSS and JavaScript files and directories must have cpanel:httpdcp ownership.

IMPORTANT:
We don't recommend you to modify the default `~cpanel/shiva/psoft/hsphere/design_config.xml` file. These modifications will be lost with the next H-Sphere updates.
There are the following ways of customizing design XML configuration:

I. With packages

You can customize design_config.xml by means of *packages*. In this case, within a package you create custom XML file that will be *merged* with default XML configuration.

II. Merging custom XML configuration

Instead of creating a package, create a custom XML configuration file to be merged with the default XML configuration and a custom package XML if installed.

III. Overrriding default XML configuration with custom XML configuration

You create the custom design configuration file and perform modifications there:

1. Login to cpanel user under root:

   su -
   su -l cpanel
2. copy the standard `design_config.xml` file to a certain custom directory (it may be `/hsphere/local/home/cpanel/shiva/custom/xml`):

   cp ~cpanel/shiva/psoft/hsphere/design_config.xml ~cpanel/shiva/custom/xml/design_config.xml
3. make changes into the custom design_config.xml structure.
4. in `~cpanel/shiva/psoft_config/hsphere.properties`, change the `DESIGN_SCHEME_CONFIG` variable to point to this new file:

   DESIGN_SCHEME_CONFIG = /hsphere/local/home/cpanel/shiva/custom/xml/design_config.xml

2. Implementation of Custom Design Templates

Custom design templates are created in the `~cpanel/shiva/custom/templates` custom template directory. In order to implement these custom designs, the symlinks to them should be put into the Apache DocumentRoot directory which is set by default to the `~cpanel/shiva/shiva-templates` standard templates directory.

However, on the subsequent H-Sphere update all symlinks in `~cpanel/shiva/shiva-templates` would be lost, and custom designs would not be displayed correctly.

To avoid this, we suggest to use another directory as DocumentRoot and to create there the symlinks to **ALL** design directories, for both custom designs and the H-Sphere built-in designs.

WARNING:
Don't change the `TEMPLATE_PATH` variable in hsphere.properties!
`TEMPLATE_PATH` points to the default template directory. If you change it, you won't see any updates in the default templates.
1. Create the `/hsphere/local/home/cpanel/shiva/web` directory.

2. Create the symlinks to the design directories using the `ln -s` command.

You should have something similar to this:

bash-2.05a$ pwd
/hsphere/local/home/cpanel/shiva/web
bash-2.05a$ ls -la

...

lrwxrwxrwx 1 cpanel cpanel 55 Jun 4 08:55 common -> /hsphere/local/home/cpanel/shiva/shiva-templates/common
lrwxrwxrwx 1 cpanel cpanel 46 Jun 2 13:39 counter -> /hsphere/shared/SiteStudio/public_html/counter
lrwxrwxrwx 1 cpanel cpanel 47 Jun 2 13:39 custom-images -> /hsphere/local/home/cpanel/shiva/custom/images/
lrwxrwxrwx 1 cpanel cpanel 50 Jun 2 13:39 custom-templates -> /hsphere/local/home/cpanel/shiva/custom-templates/
lrwxrwxrwx 1 cpanel cpanel 48 Jun 2 13:40 guestbook -> /hsphere/shared/SiteStudio/public_html/guestbook
lrwxrwxrwx 1 cpanel cpanel 55 Jun 2 13:42 IMAGES -> /hsphere/local/home/cpanel/shiva/shiva-templates/IMAGES
lrwxrwxrwx 1 cpanel cpanel 46 Jun 2 13:40 masonry -> /hsphere/shared/SiteStudio/public_html/masonry
lrwxrwxrwx 1 cpanel cpanel 55 Jun 4 08:55 nomenu -> /hsphere/local/home/cpanel/shiva/shiva-templates/nomenu
lrwxrwxrwx 1 cpanel cpanel 43 Jun 2 13:40 poll -> /hsphere/shared/SiteStudio/public_html/poll
lrwxrwxrwx 1 cpanel cpanel 48 Jun 2 13:41 shiva-templates -> /hsphere/local/home/cpanel/shiva/shiva-templates
lrwxrwxrwx 1 cpanel cpanel 59 Jun 4 08:55 text_based -> /hsphere/local/home/cpanel/shiva/shiva-templates/text_based
lrwxrwxrwx 1 cpanel cpanel 62 Jun 2 15:19 YourDesign1 -> /hsphere/local/home/cpanel/shiva/custom/templates/YourDesign1
lrwxrwxrwx 1 cpanel cpanel 62 Jun 2 15:19 YourDesign2 -> /hsphere/local/home/cpanel/shiva/custom/templates/YourDesign2

Here, the `counter`, `guestbook`, `masonry`, and `poll` directories are SiteStudio-related directories; `YourDesign1` and `YourDesign2` are custom design directories.

3. Make the `/hsphere/local/home/cpanel/shiva/web` directory in the DocumentRoot directory.

To do this, in the `~cpanel/apache/etc/httpd.conf` Apache configuration file change the DocumentRoot global definition line:

DocumentRoot "/hsphere/local/home/cpanel/shiva/shiva-templates"

to the following line:

DocumentRoot "/hsphere/local/home/cpanel/shiva/web"

Then, delete all other instances of the DocumentRoot definition (for virtual hosts) in this file. Also, delete all DocumentRoot definitions in all configuration files located in the `~cpanel/apache/etc/sites` directory.

4. Logout from cpanel back to root and *restart H-Sphere*.

Related Docs: · Template Customization · Interface Controls and Colors · Menu Customization · Adding Custom Icons

Design XML Configuration

Related Docs: · Skin and Icon Set (Design) Customization · Menu Customization · Template Customization · Interface Controls and Colors · Adding Custom Icons

H-Sphere design configuration is represented in the `design_config.xml` file, which can be found by default in the `/hsphere/local/home/cpanel/shiva/psoft/hsphere/` directory. You can customize this XML configuration to add your own designs, color schemes, colors and images.

This document explains all important parts of the design configuration file, including:

- Icons
- Skill Icon Groups
- Icon Image Sets
- Common Images
- Color Types
- Designs (Skins)

IMPORTANT:
1) Do not make changes into the default XML configuration file! Instead, follow instructions on design.xml customization.
2) Changing design XML configuration implies proper knowledge of XML. Errors in XML structure may badly damage your Control Panel interface!

Icons

By icon we mean an H-Sphere control that provides quick access to a certain functional page of the control panel. All H-Sphere icons are displayed only on the Quick Access page, which is based on the `quick/quick_view.html` template.

An icon description includes the following:
- `id` - the mnemonic system name of the icon
- `url_param` - typically, the name of the base template for the functional page this icon links to. Multiple url parameters are separated with the `&` delimiter
- `rtype` - the list of H-Sphere resource types whose availability determines whether the icon will be drawn. It may include simple resource types separated by semicolons and commas in the following way:

- `res1` - simple resource type, icon will appear if the resource is available;

- `res1,res2,res3` - group of resources where all of them must be available to show an icon (operation 'AND');
- `res1; res2; res3` - icon will be shown if any of the resources is available (operation 'OR'). Example: `rtype="webalizer; modlogan; urchin"`
- `res1,res2; res3,res4; res5,res6` - combination of groups where an icon will be shown if any of the groups - res1,res2 or res3,res4 or res5,res6 includes both available resources. If translated to C or Java, this would mean `(res1 & res2) || (res3 & res4) || (res5 & res6)`

- `platform` - operating system (unix|win2k) on which plans using this icon are based. If platform=unix or platform=win2k, this means that the icon is displayed for Unix-based or Windows-based plans, respectively. If the platform attribute is left empty, the icon is available for all plans.
- `label` - mnemonic id of the caption under the icon image. It must be defined in the `hsphere_lang.properties` file
- `tip` - mnemonic id of the HTML tooltip of the icon. It must be defined in the `hsphere_lang.properties` file
- `help` - reserved for future use.

One icon can have many visual representations. This means that it will have a different look depending on which icon image set was selected by the user. You can specify which icon image sets will be available for each individual design, e.g.:

```
  <allowed_icon_image_sets>
    <set id="wooden"/>
    <set id="square_set"/>
    <set id="cartoon_set"/>
    <set id="bubble_set"/>
  </allowed_icon_image_sets>
```

Skill Icon Groups

In terms of the H-Sphere interface visual settings, only two types of accounts are customized, regardless of plans: `admin` accounts which are H-Sphere administrative accounts, and `user` accounts - all other accounts. Reseller accounts are regarded as user accounts, except for the reseller administrative account which relates to the admin account type.

Skill icon groups determine the structure of the icon groups in the Quick Access page for these two types of accounts and are defined as follows:

```
<skill_icon_sets account_type="account_type">
  <skill_set id="standard" label="icon_skill_set.standard_account_type_l">
    <icon_group id="group_id" label="icongroups.group_id">
      <icon id="icon_id1"/>
    . . .
    <icon_group>
```



```
 . . .
 <skill_set>

 <skill_set id="advanced" label="icon_skill_set.advanced_account_type_l">
    <icon_group id="group_id" label="icongroups.group_id">
      <icon id="icon_id1"/>
          . . .
    <icon_group>
 . . .
 <skill_set>

 <skill_set id="simplified" label="icon_skill_set.simplified_account_type_l">
    <icon_group id="group_id" label="icongroups.group_id">
      <icon id="icon_id1"/>
    . . .
    <icon_group>
 . . .
 <skill_set>
<skill_icon_sets>
```

Account types: `admin` for admin accounts and `user` for user accounts.

Skill set ids are of the following types: usually, `standard` for the common ('left menu') interface, and `advanced` for the 'no menu' interface. `simplified` skill set may be chosen for any of the two types of accounts.

Icon groups are defined within the `skill_set` element structure. The icon group `id` attribute corresponds to menu groups, such as Info, FTP, mail, etc., and is a mnemonic identifier of the icon group (`mail`, `admin_mail`, and the like).

`icon_group` construction enlists the set of icons which are displayed in this icon group. Each icon is defined in the `icons` construction described above in the previous section.

Icon Image Sets

By icon image set we mean the set of icons corresponding to a certain H-Sphere color scheme:

```
<icon_image_sets base_dir="/IMAGES">
  <icon_image_set id="image_set_id" label="iconsets.image_set_id" dir="relative_image_set_dir">
```



```
    <preview_image file="/IMAGES/previews/icons_default.gif" width="375" height="60"/>

    <image id="icon_id" file="filename_with_relative_path" width="xx" height="yy"/>
    . . .
  <icon_image_set>
  . . .
<icon_image_sets>
```

base_dir attribute defines the directory where the H-Sphere images, both standard and custom, are to be stored. Typically, it is the IMAGES directory in the Apache document root directory (usually, ~cpanel/shiva/shiva-templates).

Note: The base image directory is actually relative to the alternative directory for images which is located in the document root. This directory is set in the IMAGES variable in the hsphere.properties file. If it is not set there, base_dir contains the path relative to the document root.

icon_image_set element sets the list of images corresponding to a color scheme. Attributes:

- id attribute refers to mnemonic name of the color scheme. It is default for the default color scheme; cocoa, bubble and some other schemes go with H-Sphere installation, while others may be created manually.
- dir is the path relative to the base_dir directory. If it is empty, images are located in the base images directory.

preview_image tag refers to the preview image appeared in the H-Sphere Look and Feel interface when choosing the available design.

The following attributes are determined for the preview image:

- file - filename and path to the preview image relative to the document root directory.
- width, height - image width and height.

image - image description tag.

Please keep in mind that the image should be set for EACH color scheme. In order to add a new image, first, add the image definition to the icons tag, and then add image elements with the same id attribute to EACH icon_image_set element.

The following attributes should be set:

- id attribute value refers to the icon described in the icons section.
- file is the image filename with the path relative to the icon image set subdirectory of the image set basic directory.
  For example, if base_dir="/IMAGES" and dir="wooden", then images for wooden scheme will be located in /IMAGES/wooden directory.

However, if `dir=""`, then, to do so that H-Sphere would find, let say, a GIF image, you should set the `file` attribute as `file="wooden/name_of_the_file.gif`

- `width`, `height` - image width and height. Unless these parameters are not changed, the user custom image would be displayed by H-Sphere with this width and height, regardless of the image size parameters.

Common Images

Common images are the set of images that have the same look in all designs, such as, arrows, bullets, home icon, etc.

The `common_images` element structure is as follows:

```
<common_images base_dir="/IMAGES/">
  <image id="spacer" file="spacer.gif" width="1" height="1"/>
  <image id="arrow" file="arrow.gif" width="22" height="22"/>
  . . .
</common_images>
```

The `base_dir` attribute is defined in the same way as for the icon image sets.

The image `id` attribute is an image mnemonic name used in templates.

Color Types

Color types comprise all possible interface entities for which colors are set: basic text, background, header, menu, error messages, etc.

The following attributes are present in the `color_type` tag:

- `id` is a mnemonic color type identifier later used in designs description;
- `label` is a mnemonic id to the caption under which this color type is configured in the CP interface.

Designs

Design, or skin, is a GUI representation including certain icon image sets and color schemes. There are 3 basic H-Sphere designs:

- `common` is the left-menu design. All basic templates are made for this design scheme.
- `nomenu` is the design with no left menu. It is turned on as the default user design after the H-Sphere installation.
- `text-based` is the alternative look of the nomenu design where only captions with no icons are provided in the Quick Access menu page.

Attributes of the `design` element:

- `id` is a design mnemonic identifier: nomenu, common, text-based.
- `label` is a mnemonic id to the design name in the CP interface.
- `template_dir` is a directory relative to the document root directory where where template files for this design are located.
- `default_color_scheme` is the default color scheme identifier (see below, `color_scheme` tag description).

`preview_image` tag defines the design preview image settings. The structure is the same as for the icon image sets.

`colors` element defines all available colors for this design. They are taken by default for every color scheme. Individual color settings for each color scheme could be defined in the corresponding `color_scheme` constructions (see description below).

Color `id` attribute refers to the color type identifier (see Color types).

`base_images` is the set of images that look the same for all color schemes in this design. The `base_images` tag structure is the same as for the common images.

The `image_sets` element contains the settings for each image set enabled for this design. `base_dir` attribute points to the images directory (usually, `IMAGES`). The element includes the following tags:

- `set_images` element contains the list of images in a standard `image` description which are taken by default for this design.
- `image_set` tag contains image definitions for a color scheme to override the `set_images` default definitions. Here you may set an alternative directory where images for this color scheme will be searched, or modify settings so that an image will be taken from a file different from the default image file. `image_set` may contain `image` tags for images with alternative settings (for example, if the image file name is different from the default one).

  Attributes:
  - `id` - identifier for a color scheme (for example, default or cocoa);
  - `label` - color scheme label;

- `dir` - alternative image directory relative to the `base_dir` directory; if it is empty the image search would be performed in the `base_dir` directory (usually, `IMAGES`).

Example: If we need to change the banner.jpg image to the banner1.jpg image in the default color scheme so that this image would be searched in the base directory, the `image` tag should be inserted into the `image_set` construction in the following way:

```
<image_set id="default" label="imagesets.default" dir=""/>
   <image id="banner" file="banner1.jpg" width="468" height="60"/>
</image_set>
```

`allowed_skill_icon_sets` tags are used to determine a skill icon set for user and admin accounts (see Skill icon groups):

```
<allowed_skill_icon_sets account_type="user">
  <set id="standard"/>
</allowed_skill_icon_sets>
```

`allowed_icon_image_sets` tags determine icon image sets that would be available for each account type (see Icon sets):

```
<allowed_icon_image_sets account_type="admin">
  <set id="default"/>
</allowed_icon_image_sets>
```

`color_scheme` sets page colors, image sets and icon image sets in this design.

Attributes:

- `id` - color scheme identifier (for example, blue_haze or default);
- `label` color scheme label defined in hsphere.properties;
- `image_set` - image set identifier (for example, blue_haze or default);
- `icon_image_set` - icon image set identifier (for example, blue_haze or default).

Color scheme element may include the `color` tags to set colors for this color scheme. If these colors are not set here, default colors would be taken from the `colors` tag definition for this design.

Example:

```
<color_scheme id="marsh" image_set="marsh" icon_image_set="marsh" label="imagesets.marsh">
```



```
    <color id="logo_bgcolor" value="#C9D1CA"/>
    . . .
</color_scheme>
```

Related Docs: · Skin and Icon Set (Design) Customization · Menu Customization · Template Customization · Interface Controls and Colors · Adding Custom Icons

System E-Mail Customization

Related Docs: · Template Customization

System e-mail notifications are messages H-Sphere automatically sends to customers.

It is possible to *edit system emails* in the admin CP interface, in the **Settings/Notifications/E-Mail Notifications** menu. Both default messages and messages in each interface language can be customized there. Custom modifications are stored in the H-Sphere database and do not affect the system email templates. On the contrary, default settings can be restored from the templates.

This document explains where to find the default system email templates and how to customize them.

Note: Support info and checks info is modified in the **Settings -> Look and Feel -> Misc.Text** menu by filling in the Customer Support Info and Checks Info forms.

IMPORTANT:
It is strongly recommended not to touch the default system email templates; instead, you should *edit notifications via Control Panel* to be able to restore default texts from the templates.

System Notification Templates

Here is the list of default templates for system email messages that can be customized via CP interface:

Template	**Location** (in `shiva/shiva-templates/common/mail/`)	**Notification Sent**
Welcome Letter	`new_account.txt`	to customer on account activation
Welcome Letter for Moderated Accounts	`new_account_moderated.txt`	to customer on moderated check account registration (accounts waiting activation)
Welcome Letter For Moderated Account with CC	`new_account_moderated_cc.txt`	to customer on moderated credit card registration
	`trial_moderated.txt`	to customer on moderated trial account registration

Welcome Letter For Moderated Trial Account		
Trial Registration	trial_account.txt	to customer on account trial period expiration
Invoice	invoice.txt	to customer: - on each paid operation - at the beginning of the next billing period - on switching to another billing period
Money Back	money_back.txt	to admin when a user chooses to cancel hosting and wants his/her money back
Overlimit Notification	overlimit.txt	to customers when they reach traffic or disk usage limit
Account Suspended Notification	suspended_account.txt	to customers when their accounts get suspended
Account Resumed Notification	resumed_account.txt	to customers when their accounts get suspended
Accounting Error letter	accounting_error.txt	to admin on accounting error
Lost Password	forgot_passwd.txt	to customers after they enter their email address on the "forgot your password" page
Failed Signup Notification	you_have_files_signups.txt	to admin when customer signup fails
Domain Transfer Message	tranfer_domain.txt	to customers explaining how to transfer an external domain
Internal Ticket	ticket_internal.txt	to admin in case of internal problems
Shell Access Notificaton	ssh_notification.txt	to the customer when Shell Access is granted or refused (disabled)
Welcome Letter (Tax Exemption)	new_account_tax_exemption.txt	to new customers awaiting approval of their Tax Exemption Codes
Tax Exemption Approved Notification (Moderated Accounts)	tax_exemption_approved_new.txt	to new customers signed up to tax exemption plan, awaiting moderation
Tax Exemption Rejected Notification (Moderated Accounts)	tax_exemption_rejected_new.txt	to new customers signed up to tax exemption plan, on failing to verify tax exemption data
Tax Exemption Approved Notification (Live Accounts)	tax_exemption_approved.txt	to customers on tax exemption approval

Tax Exemption Rejected Notification (Live Accounts)	`tax_exemption_rejected.txt`	to customers on failing to verify tax exemption data

Below is the list of templates for standard texts sent as mass mail. Messages in these templates cannot be customized via H-Sphere interface:

Template	**Location** (in `shiva/shiva-templates/common/mail/`)	**Notification Sent**
Welcome Letter	`welcome.txt`	optionally from mass mail
User login and password	`login_psw.txt`	optionally from mass mail
User balance	`balance.txt`	optionally from mass mail

Related Docs: · Template Customization

Context Help Implementation

Related Docs: · Template Customization · Changing Language of Context Help

Context help (or online help) is implemented through special templates, each with a topic header and a body. Context help files are located in the `/hsphere/local/home/cpanel/shiva/shiva-templates/common/online_help` directory. They have `.oh` extension and contain text in HTML format. Context help may be implemented in different languages.

This document explains how to add context help pages to H-Sphere interface.

Procedure

1. Create an online help file and put it anywhere inside `~cpanel/shiva/shiva-templates/common/online_help/`. You can create new subdirectories for your files where necessary.
2. In `~cpanel/shiva/psoft/hsphere/online_help.xml`, add an id/file correspondence, where file is the path + name of the context help file, and id is the string that will be used in the template.
3. Find the template where the context help icon will be added. The easiest way to find the name of the template is view html page source code. The templates you need are located in `~cpanel/shiva/shiva-templates/common/control/`. More about templates
4. Add context help function call to the template. The function call has the following syntax:

   <call draw_help("HELP_ID","LABEL")>

   where HELP_ID is the id of the file specified in `~cpanel/shiva/psoft/hsphere/online_help.xml`, and LABEL is the description used as the title in the html link.
   If the second parameter is left empty, the default text ("Click to get help") is used. For example:

   <call draw_help("admin-emanager-l_availableforsignup","")>

   Alternatively, you can call the function that draws help PLUS gives a link to send a trouble ticket:

   <call draw_tt_help("RESOURCE_ID", "HELP_ID", "LABEL")>

In plan creation and edit wizard templates, context help file IDs are passed with resource calls:

<call service(TAG,PARENT,STRMOD,CAN_ENABLED,OH_ID)>

where OH_ID is the id of the file specified in `~cpanel/shiva/psoft/hsphere/online_help.xml`. For example:

<call service("asp", "hosting","","1","admin-editwizard-w_asp")>

5. If the edited template is in *.html.in format, run `make` in this template's directory.
6. *Restart H-Sphere*.

Related Docs: · Template Customization · Changing Language of Context Help

User Signup Customization

This document explains how to modify standard user signup (order) forms or replace them with custom forms.

Before you begin 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.

Signup fields:

- Basic (service fields required for signup)
- User Contact Info
- Billing Info (not used for trial registration)

- Credit Cards
- Billing Period
- Domains
- Domain Registration
- Domain Regisration Contact Info

Basic (service fields required for signup)

Field name	Possible Values	Explanation
_eul_accept	"1"	Accept terms of End User License Agreement
_mod	"signup" - transfer domain, "opensrs" - domain registration, "nodomain" - stopgap domain, "3ldomain" - third level domain, "service" - service domain, "empty" - signup without domain	Signup mode
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 Domain registration)	The so-called submit template located in the `/hsphere/local/home/cpanel/shiva/shiva-templates/common` directory and used to perform server-side form validation.
admin_signup	"yes" - if we sign user up from the admin panel	Service parameter

User Contact Info

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 `_ci_state='NA'`.
_promo_code	alphanumeric	PROMO code for subsidized plan. Contains 2-20 chars and starts from the letter.

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 `_bi_state='NA'`.
_bi_type	"CC" - credit card, "Check" - check or bank transfer, "PayPal" - PayPal, "2CheckOut" - 2CheckOut, "TRIAL"	Payment type

Credit Cards

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 Settings link for this plan and scroll down to the Billing configuration section.

Domains

Field name	Possible Values	Explanation
type_domain	"transfer_new_misc_domain" - transfer domain without registrar changes "domain_transfer" - transfer domain with registrar changes "without_domain" - stopgap domain "3ldomain" - third level domain "service_domain" - service domain	Type of new domain

	"empty_domain" - signup without domain "new_opensrs_domain" - register new domain	
_mod	"signup" - transfer domain without registrar changes "dtransfer" - transfer domain with registrar changes. It accepts the same fields as OpenSRS registration except the **period** field and extra contact/billing info for domain registration. "nodomain" - stopgap domain "3ldomain" - third level domain "service" - service domain "empty" - signup without domain "opensrs" - register new domain	Domain registration mode.
domain_name	alphanumeric	Domain name; may be omitted if `type_domain="empty_domain"`

Domain Registration

Field name	Possible Values	Explanation
period	numeric	Registrar's periods (years)
_srs_owner_first_name	alphanumeric	User's first name
_srs_owner_last_name	alphanumeric	User's last name
_srs_owner_address1	alphanumeric	User's address 1
_srs_owner_address2	alphanumeric	User's address 2
_srs_owner_city	alphanumeric	User's city of residence
_srs_owner_org_name	alphanumeric	User's company name
_srs_owner_country	alphanumeric	User's country code
_srs_owner_email	alphanumeric	User's contact e-mail address

_srs_owner_phone	numeric	User's phone number
_srs_owner_postal_code	numeric	User's zip code
_srs_owner_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`.
_srs_owner_state2	alphanumeric	User's state or province for non US or Canada residents. Should be present in the custom form only in case `_srs_owner_state='NA'`.

Domain Registration Contact Info

Field name	Possible Values	Explanation
_srs_billing_first_name	alphanumeric	User's first name
_srs_billing_last_name	alphanumeric	User's last name
_srs_billing_address1	alphanumeric	User's address 1
_srs_billing_address2	alphanumeric	User's address 2
_srs_billing_city	alphanumeric	User's city of residence
_srs_billing_org_name	alphanumeric	User's company name
_srs_billing_country	alphanumeric	User's country code
_srs_billing_email	alphanumeric	User's contact e-mail address
_srs_billing_phone	numeric	User's phone number
_srs_billing_postal_code	numeric	User's zip code
_srs_billing_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`.
_srs_billing_state2	alphanumeric	User's state or province for non US or Canada residents. Should be present in the custom form only in case

		_srs_billing_state='NA'.

Compiling Templates With Client-Side Form Validation

Related Docs: · Understanding Templates · Template Customization

There are two types of templates that are responsible for generating Control Panel web content: *.html.in templates with client-side form validation that require compilation before modifications in them would take effect, and *.html templates that provide server-side form validation and don't need to be recompiled after their modification.

This document provides step-by-step instructions on how to compile control panel templates with the client-side validation of HTML form input fields.

1. Log into the control panel server as `cpanel` user:

   ```
   # su -l cpanel
   ```

   To implement customization correctly, all template files and directories should have the `cpanel:cpanel` ownership.
2. Check settings in `/hsphere/local/home/cpanel/shiva/psoft_config/hsphere.properties`:

   1) Check the `TEMPLATE_PATH` parameter. It should point to the default template directory. Default setting is:

   ```
   TEMPLATE_PATH = /hsphere/local/home/cpanel/shiva/shiva-templates/
   ```

   2) Uncomment the `USER_TEMPLATE_PATH` parameter and set it to your custom templates directory, e.g.,:

   ```
   USER_TEMPLATE_PATH = /hsphere/local/home/cpanel/shiva/custom/templates/
   ```

   3) Check the `JS` (JavaScripts) and `IMAGES` parameters:

   ```
   JS =
   IMAGES =
   ```

   By default, `JS` and `IMAGES` are left blank. It means that javascripts and images are placed inside each design directory. You don't need to change these parameters if you have default system settings.
3. Go to the default templates directory (DocumentRoot, `/hsphere/local/home/cpanel/shiva/shiva-templates` by default). Check parameters in the `configure` file:

(1) `SHIVA_ROOT` - H-Sphere Control Panel's root directory (`/hsphere/local/home/cpanel/shiva` by default)
(2) `HSPHERE_PROPERTIES` - path to the hsphere.properties file.

4. Run `./configure` in the templates directory. This will create Makefiles for all of designs.

   Warning: Running `./configure clean` would remove **ALL** the compiled templates in the nested directories and delete **ALL** Makefiles created by the previous `configure` execution! After that, your control panel interface would not show up correctly!
5. To compile all modified templates, run `make` or `make all` in your default templates directory (`gmake` for FreeBSD). If you need just to modify one template, run `make` from the directory where this template is located.

   Warning: Running the `make clean` command from a certain template directory would clear all the compiled template files (`*.html`) in the nested directories! After that, your control panel interface would not show up correctly!

Related Docs: · Understanding Templates · Template Customization

Introduction to Menu Customization

Related Docs: · Template Customization · Skin And Icon Set Customization

Control Panel menu customization affects the following elements:

- **menu structure:** by customizing the menu XML configuration file, you can restructure menu, add/delete menu groups and items, configure menu layouts for individual plans, add external links to menu.
- **menu texts:** texts for menu item labels are stored in menu language bundles.
- **menu design:** you can change the menu appearance: color, bullets, etc.

Related Docs: · Template Customization · Skin And Icon Set Customization

Changing Menu Structure

Related Docs: · Menu XML Customization · Menu Design

H-Sphere's control panel menu structure is defined in XML file, `~cpanel/shiva/psoft/hsphere/menu.xml` by default.

This document explains how you can make changes into the XML structure in order to:

- modify menu items and groups
- configure individual menu layouts for different plans
- add external links to menu items

Location

The default location of `menu.xml` is set in `~cpanel/shiva/psoft_config/hsphere.properties`:

MENU_CONFIG = /hsphere/local/home/cpanel/shiva/psoft/hsphere/menu.xml

Texts for menu elements are set in language bundles:

MENU_BUNDLE = psoft.hsphere.lang.menu

IMPORTANT:
You should not make changes into the default `~cpanel/shiva/psoft/hsphere/menu.xml` file!
There are several ways how to customize menu XML data correctly, and they are explained in a separate Menu XML Customization document.
Here in the text we refer to `menu.xml` assuming it is correctly customized according to `menu.xml` customization rules.

XML Structure

`menu.xml` consists of the following blocks going one after another:

1. DTD scheme;
2. definition of menu groups (tag <menus>)
3. definition of menu layouts for different hosting plans (tag <interface>)

Here is an example of the menu.xml file.

Modifying Menu Groups And Items

Groups of menu items are set within the menus tag. Each group is defined by a menu tag and comprises definitions of items in this group (menuitem tags), as well as inclusions of submenus (initmenu tags).

```
<menus>

<menu name="SomeMenu" label="somemenu.label" defaultitem="SomeMenu-Item1" tip="somemenu.tip">
    <menuitem name="SomeMenu-Item1" label="somemenu.edit.label"
            URL="template1.html" resource="" tip="somemenu.edit.tip"/>
    <menuitem name="SomeMenu-Item2" label="somemenu.add.label"
            URL="template2.html" resource="" tip="somemenu.add.tip"/>
    . . .
    <initmenu name="SomeSubmenu">
    . . .
</menu>
.
.
.
<menu name="SomeSubmenu" label="somesubmenu.label" defaultitem="SomeSubmenu-Item1" tip="somesubmenu.tip">
    <menuitem name="SomeSubmenu-Item1" label="somesubmenu.item1.label"
            URL="submenu_template1.html" resource="" tip="somesubmenu.item1.tip"/>
    . . .
</menu>
.
.
.
</menus>
```

These menu groups are grouped into different menu layouts defined below in menu.xml within the interface container.

Attributes of the `menu` tag:

- `name` - the name of the group.
- `label` - the mnemonic identifier of the text label for the menu group name displayed in CP. This label is set in the `menu.properties` bundle, for example:

  ```
  somemenu.label = Sample Menu Group
  ```

  See more about interface text bundles.
- `defaultitem` - the name of the menu item that becomes active by default when the menu group is opened; contains the menu item name (the `name` attribute of the `menu` tag).
- `tip` - the menu group tooltip label set in the `menu.properties` bundle.

Attributes of the `menuitem` tag:

- `name` - the name of the item.
- `label` - the menu item mnemonic identifier from the `menu.properties` bundle (see explanation above for the `label` attribute of the `menu` tag).
- `URL` - the template file the item refers to, the pathname is relative to each design directory in the template directory. Read more in Understanding Templates for template directory structure.
- `resource` - the resource name that must exist in the account for this item to be shown in the menu.
- `tip` - the menu item tooltip label set in `menu.properties`.

Attributes of the `initmenu` tag:

- `name` - name of the menu group referred to the `name` attribute of the `menu` tag described among other menus in the `menus` container.

According to `menu.xml` customization rules, you can do the following modifications in the structure of menu items and groups:

1. add/delete menu groups, items and submenus, and edit their attributes.
2. choose menu items to be activated by default when menu group is opened in CP.
3. restructure the order of menu items within a group.

Configuring Individual Menu Layouts For Different Hosting Plans

Each hosting plan in H-Sphere may have its own menu configuration set in `menu.xml` after the description of menu groups. Plan menu structure is set by the `menudef` tag within the `interface` container. For example:

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

The `menudef` tags contain sets of menu groups and separate items in order of their appearance in H-Sphere interface. The `name` attribute of `initmenu` tags points to the name of the respective menu group defined above in `menu.xml` in the `menus` container.

The value for the `id` attribute of the `menudef` tag (`TestPlan` in the example above) must be the same as of the menuId parameter in the Plan Settings form (the Info->Plans menu, the Settings icon for the plan, in admin CP). Please refer to the *Plan Settings* document in Admin Guide.

These are default menu identifiers for the standard H-Sphere plan types:

- `unix` - Unix plan;
- `admin` - Admin plan;
- `ttadmin` - Trouble Ticket Admin plan;
- `bill` - Billing plan;
- `reseller` - Reseller plan;
- `winduz` - Windows plan;
- `real` - Real Server plan;
- `mysql` - MySQL Only plan;
- `email_only` - E-Mail Only plan;

`vps` - VPS plan.

To add your custom menu and assign it to a specific plan, do the following:

1. According to `menu.xml` customization rules, add the new `menudef` structure for the plan and fill it with menu groups and items as shown in the above example:

   ```
   <menudef id="custom_menuId">
   ...
   </menudef>
   ```
2. In the admin panel, in the plan Settings form, set the menuId custom value to custom_menuId (read the *Plan Settings* document in Admin Guide for details.)

Assigning External Links to Menu Items

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 design subdirectories of the template directory (`~cpanel/shiva/shiva-templates` by default) and will be searched by H-Sphere according to its template lookup sequence.

A solution is in creating a template (specially-formatted HTML document) redirecting to an external URL.

Consider an example of menu customization where to a certain external page on logout from admin CP.

We assume that `menu.xml` and the template created are customized properly according to the respective customization rules.

Note: Alternatively, instead of creating a new `logout_redirect.html` template with an external link, you may customize the standard logout.html template without customizing menu.xml. The example below is given merely to illustrate menu.xml customization.

1) In `menu.xml`, find the element corresponding to the admin plan:

```
<menudef 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 `menudef` 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. In this case, the URL attribute should be set as:

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

4) In the specified directory, create the redirecting HTML file. H-Sphere will search for it in `~cpanel/shiva/custom/templates/common/misc/logout_redirect.html`.
See more about template lookup sequence.

The HTML redirecting document will be organized as follows:

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

5) *Restart H-Sphere* after the customization is dome.

6) Refresh browser to see the changes. It is better to logout and login again.

Related Docs: · Menu XML Customization · Menu Design

Menu XML Customization

Related Docs: · Menu Customization · Menu Structure · Menu XML Customization

The Control Panel menu structure is represented in an XML file. Its default location is set in `~cpanel/shiva/psoft_config/hsphere.properties` by the `MENU_CONFIG` parameter:

```
MENU_CONFIG=/hsphere/local/home/cpanel/shiva/psoft/hsphere/menu.xml
```

You should not make changes in the default menu.xml file, because these changes will be lost with the next H-Sphere upgrade. Instead, you create custom XML files that either override or merge changes with default configuration.

Below are the alternative ways to customize menu.xml.

1. With Packages

You create a custom menu.xml file within an *H-Sphere package*. After the package is installed, this custom XML configuration will be *merged* with default XML configuration.

2. Merging Custom XML Configuration

If you need minor modifications in menu configuration, you don't need to rebuild a package. Instead, you create a custom XML configuration file to be merged with the default XML configuration and a custom package XML if installed. In this case, you avoid changing the default `MENU_CONFIG` property in hsphere.properties and use the `CUSTOM_MENU_CONFIG` parameter to specify your custom menu.xml location.

Related Docs: · Menu Customization · Menu Structure · Menu XML Customization

Menu Design Customization

Related Docs: · Menu Customization · Template Customization · Skin And Icon Set Customization

This document 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 and with H-Sphere templates.

The template to be customized is `~cpanel/shiva/shiva-templates/common/menu.fn`. The instruciton below is the regular way of template customization. This customization can be also performed by means of *H-Sphere packages* explained in Developer Guide.

1. Login as cpanel, under root:

   su -
   su -l cpanel
2. Create the `~cpanel/shiva/custom` directory if it doesn't exist yet.
3. Inside the `custom/` directory, create the following directories if they aren't there:

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

   Note: If the `images/` directory is created, there must be a symlink to this directory from the Apache document root (which is the default template directory `~cpanel/shiva/shiva-templates`); if not, users should place their images into the `IMAGES` directory.
4. Put your images into the `images/` directory and your interface text files (bundles) into the `bundles/` directory.
5. In the `templates/` directory, create the `common/` directory if it doesn't exist and copy the `menu.fn` file from the `shiva-templates/common/` directory into that directory.
6. Make changes in custom `menu.fn`. The following Freemarker functions are used in `menu.fn` to draw the navigation menu:

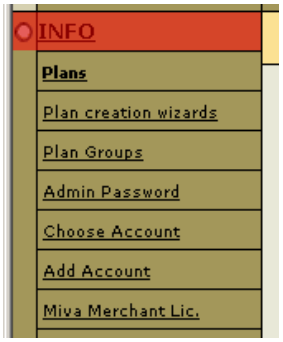
   - `<function draw_menu(activeItem)>;`
   - `<function draw_sub(item, level)>;`
   - `<function draw_item(item, level)>;`
   - `<function draw_blank_menu()>.`

   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:

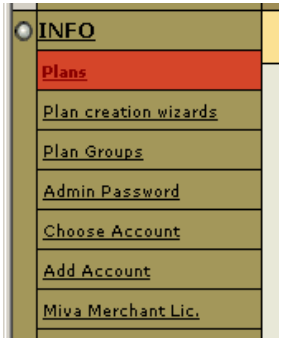
```
<TABLE WIDTH="100%" BORDER="0" CELLPADDING="0" CELLSPACING="0">.
```

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

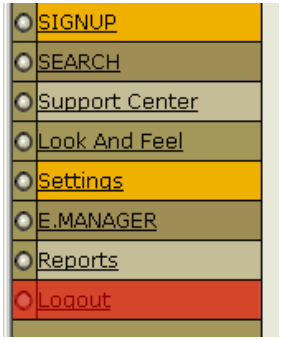


The menu_was_drawn variable is used to figure out how to show the next item.

(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:

```
<IMG SRC="path_to_your_image/replacing_image" WIDTH="xx" HEIGHT="yy"></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/`.

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

   USER_TEMPLATE_PATH = /hsphere/local/home/cpanel/shiva/custom/templates/
   CUSTOM_MENU_BUNDLE = custom.bundles.menu

   See more about template customization and menu.xml customization.

   Note: Don't initialize the variables you are not using!
8. Login as root and *restart H-Sphere*.

Related Docs: · Menu Customization · Template Customization · Skin And Icon Set Customization

Understanding Interface Text (Language) Bundles

Related Docs: · Understanding Templates · Interface Text Customization · Adding New Languages To H-Sphere · Language Bundle Compiler

Interface texts are defined in key=value sets collected in separate files for each language and encoding, where key is a mnemonic identifier for the text, and value is the text itself that is different for each language and encoding. Such files are called resource bundles, or language bundles, or simply bundles. H-Sphere templates include mnemonic identifiers and thus generate language-independent dynamic content.

Default Bundles (Installed With H-Sphere)

Default bundles are bundles containing text values H-Sphere takes by default (see bundle lookup sequence). Default bundle location is set in `~cpanel/shiva/psoft_config/hsphere.properties`:

Important: Starting with H-Sphere 3.0 RC 1, `menu.properties` and `messages.properties` become deprecated, and all labels are merged into a single `hsphere_lang.properties` for each language!

TEMPLATE_BUNDLE=psoft.hsphere.lang.hsphere_lang
MENU_BUNDLE=psoft.hsphere.lang.menu
USER_BUNDLE=psoft.hsphere.lang.messages

It means that default bundles are set in the following files of the `/hsphere/local/home/cpanel/shiva/psoft/hsphere/lang/` directory:

1. Texts in English

Important: Starting with H-Sphere 3.0 RC 1, `menu.properties` and `messages.properties` become deprecated, and all labels are merged into a single `hsphere_lang.properties` for each language!

- **`hsphere_lang.properties`** - template and tooltip texts;
- **`menu.properties`** - the navigation menu texts;
- **`messages.properties`** - system e-mail notification texts. It is recommended *changing them in CP interface*.

2. Texts in other languages

Important: Starting with H-Sphere 3.0 RC 1, `menu.properties` and `messages.properties` become deprecated, and all labels are merged into a single `hsphere_lang.properties` for each language!

```
hsphere_lang_<language>_<COUNTRY>_<ENCODING>.properties
menu_<language>_<COUNTRY>_<ENCODING>.properties
messages_<language>_<COUNTRY>_<ENCODING>.properties
```

Here, `<language>`, `<COUNTRY>`, and `<ENCODING>` are Java-compliant language, country and encoding identifiers.

Please refer to the tables of canonical identifiers:

- Languages
- Countries
- Encodings (aka charsets or variants).
  Note: For CP server under FreeBSD, use canonical encodings for Java 1.3.

All source bundles, in whatever encoding they were created, are compiled into Unicode (UTF-8). Therefore, in most cases `<ENCODING>` and `<COUNTRY>` identifiers may be omitted in bundle names.

1) Some languages have different character sets, e.g., standard Chinese and simplified Chinese. In such case you cannot omit the encoding in bundle names.

For example, standard Chinese (Big5 encoding) bundles will look like:


```
hsphere_lang_zh_CN_Big5.properties
menu_CN_Big5.properties
messages_CN_Big5.properties
```

And for simplified Chinese (EUC_CN encoding) bundles will be:

```
hsphere_lang_zh_CN_EUC_CN.properties
menu_CN_EUC_CN.properties
messages_CN_EUC_CN.properties
```

2) Some languages have variations in different countries, e.g., Portuguese (Portugal) and Portuguese (Brazil). Then, you specify language and country for each case:

For example, Portuguese (Portugal) bundles will be:

```
hsphere_lang_pt_PT.properties
menu_pt_PT.properties
messages_pt_PT.properties
```

Portuguese (Brazil) bundles will be:

```
hsphere_lang_pt_BR.properties
menu_pt_BR.properties
messages_pt_BR.properties
```

3) If you know for sure the language won't be used for other countries, you may omit the country identifier, e.g., for Russian:

```
hsphere_lang_ru.properties
menu_ru.properties
messages_ru.properties
```

Important: In any case, you must fully specify the correct language, country and encoding identifiers in the `LANG_LIST` parameter in *hsphere.properties* or in *package properties file* for H-Sphere packages. In particular, H-Sphere should know in what encoding the bundles were created to be able to convert them to Unicode.

`LANG_LIST` contains definitions for the languages, delimited with whitespace, each including the following components:

```
<language>_<COUNTRY>_<ENCODING>|<HTML_ENCODING>:misc.langs.<LABEL>lang
```


Here:

- <HTML_ENCODING> is HTML-compliant encoding (parameter deprecated since 2.4 and is not really used but must still be specified for the sake of compatibility);
- misc.langs.<LABEL>lang is the label for the language name in H-Sphere interface. These labels are set in hsphere_lang bundles. For example:

```
misc.langs.rulang = Russian
```

Custom Bundles

Default bundle directory is being rewritten with each H-Sphere update. So, if you make changes in the default bundles or add new bundles to the default bundle directory, you will lose all such modifications.

Instead, you use custom bundles to modify and expand default bundles.

Custom bundle location is set in hsphere.properties in the CUSTOM_TEMPLATE_BUNDLE, CUSTOM_MENU_BUNDLE, and CUSTOM_USER_BUNDLE parameters. For example:

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

It means that custom bundles will be searched by H-Sphere in the ~cpanel/shiva/custom/bundles/ custom bundle directory.

Custom bundle files are of the same filename format as in the default bundle directory. To customize texts in the default bundles, files with the same names are created in the custom bundle directory; they contain only labels to be added or to override the default texts. New language bundles are also created in the custom bundle directory, and you don't worry they'll be lost with the next updates.

See the following documents for customization instructions:

- Interface Text Customization
- Adding New Language To H-Sphere

It is also possible to build installable *packages* with custom bundles.

Resulting Internal Bundles

Default and custom bundles, along with bundles coming with H-Sphere packages, are compiled and merged into the so-called internal bundles located in the `~cpanel/shiva/languages` directory. It is there that H-Sphere takes bundles from.

Internal bundles location can be changed. You either set `INT_LANGBUNDLE_DIRECTORY` to override the whole internal bundles directory (`~cpanel/languages`), or set `INT_TEMPLATE_BUNDLE`, `INT_MENU_BUNDLE`, `INT_USER_BUNDLE` to override the respective internal bundles.

Please refer to Compiling Language Bundles for details.

Bundle Lookup Sequence

H-Sphere searches for text labels in language bundles in the following order:

The search is made in the internal bundle directory (`~cpanel/shiva/languages` by default):

1. `<bundle_name>_<language>_<COUNTRY>_<ENCODING>.properties`
2. `<bundle_name>_<language>_<COUNTRY>.properties`
3. `<bundle_name>_<language>.properties`
4. `<bundle_name>.properties` (default English text)
5. If no appropriate labels are found, the user will get a corresponding notification and a possibility to send a trouble ticket.

Note that no different encodings for languages are present in internal bundles. Language bundle compiler converts all regional data in default and custom language bundles into UTF-8 format.

Related Docs: · Understanding Templates · Interface Text Customization · Adding New Languages To H-Sphere · Language Bundle Compiler

Interface Text Customization

Related Docs: · Understanding Templates · Introduction to Bundles · Adding New Languages To H-Sphere

This document dwells on interface text customization methods. It implies you are familiar with the concept of language bundles.

These are the following alternative ways to customize texts:

- ## With packages

  Packages are helpful when you want to *add a new language to H-Sphere*. In this case, you build new language bundles into a portable package easily installed on other H-Sphere's.

- ## Compiling bundles

  This new scheme of customizing bundles enables you to apply modifications by merging custom bundles with default bundles without building and installing packages. Read more about compiling bundles.

  1. Log into CP server as *cpanel user*.

     All affected files must have `cpanel:cpanel` ownership.
  2. Create custom bundle directory, e.g., `~cpanel/shiva/custom/bundles` if it is not created yet.
  3. Set custom bundle location in hsphere.properties (if it's not set):

     CUSTOM_TEMPLATE_BUNDLE=custom.bundles.hsphere_lang
     CUSTOM_MENU_BUNDLE=custom.bundles.menu
     CUSTOM_USER_BUNDLE=custom.bundles.messages
  4. Find the string you want to modify among default or custom bundles. Create the corresponding custom bundle file if it doesn't exist to place modified string there.

     For example, you are going to change the label **Shell Access** to **SSH Access**. It is set in the `~cpanel/shiva/psoft/hsphere/lang/hsphere_lang.properties` file. Create the file `~cpanel/shiva/custom/bundles/hsphere_lang.properties` if it isn't there already.
  5. Copy the line with the identifier and the value you want to change into the custom bundle 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}

6. Run language bundle compiler to implement customization:

```
java psoft.hsphere.LangBundlesCompiler
```

Related Docs: · Understanding Templates · Introduction to Bundles · Adding New Languages To H-Sphere

Language Bundle Compiler

Related Docs: · Introduction to Bundles · Interface Text Customization · Adding New Languages

Language bundles - default, custom, and those installed with H-Sphere packages - are compiled and **merged** into the resulting **internal bundles** located in the internal bundle directory, `~cpanel/shiva/languages` by default. Instead of parsing default and custom bundles, H-Sphere takes interface texts directly from this directory.

Merging language bundles is performed by a Java tool called **language bundle compiler**:

java psoft.hsphere.LangBundlesCompiler

Implications

You need to run bundle compiler if you customize language bundles, namely:

- add new languages to H-Sphere;
- edit interface texts.

Also, bundle compiler is launched when you install a package with language bundles.

How Does Language Bundle Compiler Work

1. Bundle compiler parses the `LANG_LIST` parameters set in:

   - *package properties files* for all packages installed in H-Sphere: usually, `~cpanel/shiva/packages/`PackageName`/default.properties`, where PackageName is the name of the package without its version and build number;
   - the default `~cpanel/shiva/psoft_config/hsphere.properties` file;

   `LANG_LIST` contains definitions for the languages, delimited with whitespace, each including the following components:

<language>_<COUNTRY>_<ENCODING>|<HTML_ENCODING>:misc.langs.<LABEL>lang

Here:

- <language>, <COUNTRY>, and <ENCODING> are Java locale identifiers. For detailed description and the tables of canonical identifiers, please refer to Understanding Language Bundles"). <ENCODING> is an encoding in which bundles were created and saved,
- <HTML_ENCODING> is the HTML-compliant encoding (this parameter is deprecated since 2.4 and is not really used but must still be specified for the sake of compatibility);
- misc.langs.<LABEL>lang is a label for language name in H-Sphere.

According to Java locale identifiers set in the LANG_LIST parameters, bundle compiler looks for the bundles:

Important: Starting with H-Sphere 3.0 RC 1, menu.properties and messages.properties become deprecated, and all labels are merged into a single hsphere_lang.properties for each language!

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

hsphere_lang_<language>.properties
menu_<language>.properties
messages_<language>.properties

hsphere_lang_<language>_<COUNTRY>.properties
menu_<language>_<COUNTRY>.properties
messages_<language>_<COUNTRY>.properties

hsphere_lang_<language>_<COUNTRY>_<ENCODING>.properties
menu_<language>_<COUNTRY>_<ENCODING>.properties
messages_<language>_<COUNTRY>_<ENCODING>.properties
```

in the following directories, in order of priority:

- **installed package bundles:** ~cpanel/shiva/packages/PackageName.

  Package bundle location is set in the package properties file ~cpanel/shiva/packages/PackageName/default.properties:

TEMPLATE_BUNDLE=packages.PackageName.hsphere_lang
MENU_BUNDLE=packages.PackageName.menu
USER_BUNDLE=packages.PackageName.messages

- **custom bundles:** ~cpanel/shiva/custom/bundles.

  Custom bundle location is set in hsphere.properties:

  CUSTOM_TEMPLATE_BUNDLE=custom.bundles.hsphere_lang
  CUSTOM_MENU_BUNDLE=custom.bundles.menu
  CUSTOM_USER_BUNDLE=custom.bundles.messages
- **default bundles:** ~cpanel/shiva/psoft/hsphere/lang.

  Default bundle location is set in hsphere.properties:

  TEMPLATE_BUNDLE=psoft.hsphere.lang.hsphere_lang
  MENU_BUNDLE=psoft.hsphere.lang.menu
  USER_BUNDLE=psoft.hsphere.lang.messages

See Introduction To Bundles for details.

2. Bundle compiler merges default, custom and package bundles with the same filenames and save the resulting bundles in UTF format into the internal bundle directory ~cpanel/shiva/languages.

   See Bundle Lookup Sequence.

   Notes:

   - If the ~cpanel/shiva/languages directory is not found, language bundle compiler will be launched automatically and will create this directory with the resulting bundles.
   - To override the default internal bundle directory, set the INT_LANGBUNDLE_DIRECTORY parameter in hsphere.properties. Or, set INT_TEMPLATE_BUNDLE, INT_MENU_BUNDLE, INT_USER_BUNDLE to override the location of the respective internal bundles.
   - When a text label is set several times, e.g., in a package bundle and in a custom/default bundle, package bundle takes precedence and override labels set in custom or default bundles. The order of priority is: package bundles, custom bundles, default bundles.
   - Depending on bundle encodings specified in the LANG_LIST parameters, bundle compiler converts all regional language bundles into UTF-8. This is done due to the Java restrictions in relation to regional encodings: Java can work only with ISO-8859-1 and UTF symbols.

     Hints:

a) To convert regional symbols into Unicode, you can use the `native2ascii` JDK tool.

b) To convert a lang file into Unicode, you can also run `make` in the `~cpanel/shiva/psoft/hsphere/lang` directory.

Related Docs: · Introduction to Bundles · Interface Text Customization · Adding New Languages

Adding New Languages To H-Sphere

Related Docs: · Understanding Templates · Understanding Bundles · Interface Text Customization · Updating Translation of H-Sphere Interface

This document introduces methods of adding languages to H-Sphere interface. We presume you are familiar with the concept of language bundles.

Translating Language Bundles

NOTE: H-Sphere developers are not responsible for adding languages to H-Sphere interface and updating translation of the default English bundles. It is entirely up to H-Sphere owners. All language bundle translations included into H-Sphere interface are kindly provided by our customers. You are also welcome to contact PSoft team and share your new and updated translations, in files or packages, to have them incorporated into the upcoming versions of H-Sphere.

Default H-Sphere interface language is English. The default English files are located in `~cpanel/shiva/psoft/hsphere/lang`:

Important: Starting with H-Sphere 3.0 RC 1, `menu.properties` and `messages.properties` become deprecated, and all labels are merged into a single `hsphere_lang.properties` for each language!

- `hsphere_lang.properties` - template and tooltip texts;
- `menu.properties` - the navigation menu texts;
- `messages.properties` - system e-mail notification texts.

You can also download default (English) language bundles from our site.

Translate these files and save new language bundles to a separate location as:

```
hsphere_lang_<language>_<COUNTRY>_<ENCODING>.properties
menu_<language>_<COUNTRY>_<ENCODING>.properties
```



```
messages_<language>_<COUNTRY>_<ENCODING>.properties
```

where <language>, <COUNTRY>, and <ENCODING> are Java locale identifiers. For detailed description and the tables of canonical identifiers, please refer to Understanding Language Bundles.

For example, for Portuguese (Brazil) these files will be:

```
hsphere_lang_pt_BR.properties
menu_pt_BR.properties
messages_pt_BR.properties
```

Please take into account the following considerations on translating the bundles:

- **Using quotes in the text:** 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}
- **Updating the translation:** With upoming versions of H-Sphere, default English texts may be changed and new labels may be added, so you need to update translation of your language bundles accordingly.
- **Encoding:** Translation can be performed in any encoding. All language bundles are converted into UTF-8 during their compilation.
- **Right-to-left languages (Arabic, Hebrew, etc.):** H-Sphere supports right to left languages, such as Arabic or Hebrew. On this step, you need to specify whether the target language is "left to right" or "right to left":
  In the file hsphere_lang_<LANGUAGE_CODE>.properties, add the following lines:

  ```
  ######################################################
  # "text_direction" is a special label used to define
  # which text direction is appropriate for the current
  # language bundle.
  # There are 2 possible values:
  # - "ltr" means "Left to right";
  # - "rtl" means "Right to left".
  text_direction = ltr
  ```


setting `text_direction` to the value which is appropriate for the target language.

Adding New Language Bundles Into H-Sphere

H-Sphere provides the following alternative ways of adding new languages:

With packages

H-Sphere provides an easy way to add languages by installing packages (.hsp files) to the system. You may install ready-to-use and portable packages to any of your H-Sphere control panels or *build language packages yourself*.

Compiling bundles

Instead of building a package, you create new language bundles in custom bundle directory and run language bundle compiler.

1. Login as the *cpanel user*. All affected files must have `cpanel:cpanel` ownership.
2. Create custom bundle directory, e.g., `~/shiva/custom/bundles` if it is not created yet.
3. Set custom bundle location in `~cpanel/shiva/psoft_config/hsphere.properties` (if it's not set):

   Important: Starting with H-Sphere 3.0 RC 1, `menu.properties` and `messages.properties` become deprecated, and all labels are merged into a single `hsphere_lang.properties` for each language!

   ```
   CUSTOM_TEMPLATE_BUNDLE=custom.bundles.hsphere_lang
   CUSTOM_MENU_BUNDLE=custom.bundles.menu
   CUSTOM_USER_BUNDLE=custom.bundles.messages
   ```
4. Create the files `hsphere_lang.properties`, `menu.properties`, and `messages.properties` in the `~/shiva/custom/bundles` directory if they are not there. Even if you don't need to modify them, they are important for correct bundle compilation:

   ```
   cd ~cpanel/shiva/custom/bundles
   touch hsphere_lang.properties menu.properties messages.properties
   ```
5. Copy the translated language bundles to `~cpanel/shiva/custom/bundles`. For example, for Portuguese (Brazil):

cd ~cpanel/shiva/custom/bundles
cp /some/location/hsphere_lang_pt_BR.properties .
cp /some/location/menu_pt_BR.properties .
cp /some/location/messages_pt_BR.properties .

6. 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.<LABEL>lang = <LANGUAGE> (<COUNTRY>)

   For example, for Portuguese (Brazil):

   misc.langs.ptlang = Portugu\u00eas (Brasil)

   Notes:

   1) Default (English) language bundles are written in the ISO-8859-1 encoding. Special Latin and non-Latin characters must be presented in Unicode. Use the native2ascii JDK tool to convert these symbols into Unicode characters (`\uxxxx` notations, like "`\u00ea`" instead of "ê" in the example above).

   2) If the original language name significantly differs from that in English, (especially for non-Latin alphabets), we recommend adding its English transcription, e.g.:

   misc.langs.de_atlang = Deutch (\u00d6sterreich) - German (Austria)
7. Add new language to the list of languages available in H-Sphere. For this, add the language and encoding of the translated files to the `LANG_LIST` parameter in `~cpanel/shiva/psoft_config/hsphere.properties`. For example:

   LANG_LIST = en_US_ISO8859_1|ISO-8859-1:misc.langs.englang pt_BR_ISO-8859-15|ISO-8859-15:misc.langs.ptlang

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

   <language>_<COUNTRY>_<ENCODING>|<HTML_ENCODING>:misc.langs.<LABEL>lang

   Here, `misc.langs.<LABEL>lang;` is the label with the language name, which is set in the previous step.
8. In `hsphere.properties`, 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 Portuguese (Brazil) language will look as follows:


```
# Override system locale
LOCALE = pt_BR
# Encoding
ENCODING = UTF-8
```

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

9. Run language bundle compiler to implement new bundles into H-Sphere:

```
java psoft.hsphere.LangBundlesCompiler
```

10. *Restart H-Sphere*

Related Docs: · Understanding Templates · Understanding Bundles · Interface Text Customization · Updating Translation of H-Sphere Interface

Changing Language of Context Help

Related Docs: · Understanding Templates · Interface Text Customization · Adding New Languages To H-Sphere

Online help texts are hard-coded in special H-Sphere templates with .oh extension located in the `~cpanel/shiva/shiva-templates/common/online_help/` directory.

Carefully follow the procedure outlined in the Customizing Templates document, with the following considerations specific to context help templates:

- Copy the online help files from the `~cpanel/shiva/shiva-templates/common/online_help/` folder to the custom template directory `shiva/custom/templates/common/online_help/`, to files whose names contain identifiers for target language, country and encoding, in the format similar to that of language bundles:

  cp ~cpanel/shiva/shiva-templates/common/online_help/some/dir/helpfile.oh
  ~cpanel/shiva/custom/templates/common/online_help/some/dir/helpfile_<LANGUAGE>_<COUNTRY>.oh

  Consider the following example for the Russian language:

  cp ~cpanel/shiva/shiva-templates/common/online_help/user/b_invoice/01balance.oh
  ~cpanel/shiva/custom/templates/common/online_help/user/b_invoice/01balance_ru_RU.oh

  IMPORTANT:
  H-Sphere uses the **UTF-8** charset for all languages. Therefore, text directly inserted into templates (i.e., not by means of text labels defined in language bundles) **must be** in the UTF-8 encoding. Other regional encodings (for example, Windows-1251 for Russian) must not be used anymore in context help templates, and texts there must be converted into UTF-8.
- Translate the heading and the body of each help file in the corresponding encoding.

Related Docs: · Understanding Templates · Interface Text Customization · Adding New Languages To H-Sphere

Updating Translation of H-Sphere Interface

Related Docs: · Template Customization · Introduction To Bundles · Adding New Languages To H-Sphere

With every update of H-Sphere, your translation of the control panel interface becomes outdated, and you need to update translation in language bundles.

Note: H-Sphere developers are not responsible for adding languages to H-Sphere interface and updating translation of the default English bundles. It is entirely up to H-Sphere owners. All language bundle translations included into H-Sphere interface are kindly provided by our customers. You are also welcome to contact PSoft team and share your new and updated translations, in files or packages, to have them incorporated into the upcoming versions of H-Sphere.

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 `~cpanel/shiva/psoft/hsphere/lang/` default bundle directory.

To compare two files, do the following:

Step 1. Log into your CP server as the *cpanel user*.

Step 2. Go to the default bundle directory:

cd shiva/psoft/hsphere/lang/

Step 3. Execute the following command:

./diff_labels.pl hsphere_lang.properties /path/to/hsphere_lang_<LOCALE>.properties > /path/to/hsphere_lang_<LOCALE>.diff
./diff_labels.pl menu.properties /path/to/menu_<LOCALE>.properties > menu_<LOCALE>.diff
./diff_labels.pl messages.properties /path/to/messages_<LOCALE>.properties > /path/to/messages_<LOCALE>.diff

Here, `/path/to/` is a path to language bundles in another language (they may not be located in the default bundle directory), `<LOCALE>` contains language, country and encoding identifier in accordance with language bundle filename format.

The tool will write all the differences between these updated default language bundles and their previous translation into the corresponding `.diff` files.

Step 4. Customize your language bundles by adding translated labels from the resulting `.diff` files.

Related Docs: · Template Customization · Introduction To Bundles · Adding New Languages To H-Sphere

Package Installation

Related Docs: · Package Upgrade · Package Uninstallation · Installation of Missing H-Sphere Package Components

This document explains how to install custom packages on standard H-Sphere.

1. Login to CP server as *cpanel user*.
2. If a package contains templates, prepare your custom template directory according to steps 1-3 of Template Customization.
3. Install the package using the **Package Installer** tool. Use the `--help` option for the syntax:

   ```
   java psoft.hsp.tools.PkgInstaller --package=/path/to/package/file [--upgrade] [--check-only] [--force]
   ```

   Options:

   - `--package=/path/to/package/file`

     Specifies the path to the built package. H-Sphere package is a Java archive file (JAR) with `.hsp` extension.
   - `[--check-only]`

     Makes utility not install the package but only perform a check routine if the package can be installed.
   - NEW! (HS 2.4.3+) `[--upgrade]`

     Use this option if you upgrade the package already installed on your H-Sphere. With this option, you can easily upgrade the package without uninstallation of the older package and restarting H-Sphere after the uninstallation.
   - `[--force]`

     Forces the package installation even if conflicts were detected. For example, if you are installing two packages that use exactly the same file, install the first one without, and the second one with the `--force` option. The first instance of the file will be replaced with the second. Use this option only if you are sure this won't damage other packages.
4. Check whether all H-Sphere package components have been successfully installed.
5. *Restart H-Sphere*

If something goes wrong during or after the installation, you can uninstall the package.

The files installed with the package will be copied to the `~cpanel/shiva/packages/`PackageName directory, where PackageName is the name of the package without its version and build number. For example, after installation of the `MyPackage-1.0.1-2.hsp` package its files will be stored in the `~cpanel/shiva/packages/MyPackage` directory.

The List of Installed Packages in CP

After you installed a package, it will be added to the list of installed packages in your admin Control Panel, in the Settings->Packages menu:

Click on a package to check its details and files installed by that package:

This is helpful to find out whether all package files are installed correctly.

Related Docs: · Package Upgrade · Package Uninstallation · Installation of Missing H-Sphere Package Components

Package Upgrade

Related Docs: · Package Installation · Package Uninstallation · Installation of Missing H-Sphere Package Components

This document explains how to upgrade H-Sphere packages on standard H-Sphere in versions:

- 2.4.3 and higher
- before 2.4.3

Updating H-Sphere packages in versions 2.4.3 and higher

H-Sphere 2.4.3 introduces the `--upgrade` option to the Package Installer tool. With this option, you don't need to care about uninstalling the older package and restarting H-Sphere before updating the package. To upgrade a package in 2.4.3 do:

1. Login to CP server as *cpanel user*.
2. Install the new package with the `--upgrade` option:

   ```
   java psoft.hsp.tools.PkgInstaller --package=/path/to/package/file --upgrade
   ```

   More Package Installer options
3. *Restart H-Sphere*

The `--upgrade` option removes the older installation and installs the updated package files instead.

The package files will be installed to the `~cpanel/shiva/packages/`PackageName directory, where PackageName is the name of the package without its version and build number. For example, after installation of the `MyPackage-1.0.1-2.hsp` package its files will be stored in the `~cpanel/shiva/packages/MyPackage` directory.

To ensure the package is upgraded, check the Settings->Packages menu in admin interface for the package version and the package files installed.

Updating H-Sphere packages in versions before 2.4.3

Before 2.4.3, to upgrade package you needed to do the following:

1. Login to CP server as *cpanel user*.
2. Uninstall the older package.
3. *Restart H-Sphere*
4. Install the new package.
5. *Restart H-Sphere*

Related Docs: · Package Installation · Package Uninstallation · Installation of Missing H-Sphere Package Components

Package Uninstallation

Related Docs: · Package Installation

To uninstall a package:

1. Log into CP server as *cpanel user*.
2. Run package uninstaller:

   ```
   java psoft.hsp.tools.PkgUnInstaller --pkg-name=PackageName [--force]
   ```

   Options:

   - `--pkg-name`=PackageName - specify the package name. It should be without the version and build number, as in the Package Name column in the list of installed packages (Settings/Packages menu in admin panel).
     For example, if the package installed is `MyPackage-1.0.1-2.hsp`, package name to be specified is MyPackage:

     ```
     java psoft.hsp.tools.PkgUnInstaller --pkg-name=MyPackage
     ```
   - `--force` - force the uninstallation.
3. *Restart H-Sphere*.

Related Docs: · Package Installation

Installation of Missing H-Sphere Package Components on New Physical Servers

Related Docs: · Package Installation · Package Uninstallation

This document explains how to check and install missing components (scripts, 3rdparty bundles, etc.) of H-Sphere *.hsp packages on new H-Sphere physical boxes.

For H-Sphere 3.0 and later: run update with option *3rdpackages* NEW!

For H-Sphere before 3.0:

1. Login to CP server as *cpanel user*.
2. Run:

   ```
   java -Xms128m -Xmx512m psoft.hsp.tools.PkgRecheck --force
   ```

   Execute this script without the `--force` option to see what components of H-Sphere *.hsp packages are missing on a new H-Sphere box.
   If run with the `--force` option as shown above, the script checks H-Sphere database for missing components of all required *.hsp packages and installs them on a new physical box.
3. *Restart H-Sphere*

This will install missing parts of H-Sphere *.hsp packages on a new physical box.

Related Docs: · Package Installation · Package Uninstallation

Merging XML Configuration Files

Customizable XML Configuration Files

The following H-Sphere components are configured by means of XML files:

Component	**Property** (in *hsphere.properties*)	**Default Location**
CP Skins (Designs)	DESIGN_SCHEME_CONFIG	/hsphere/local/home/cpanel/shiva/psoft/hsphere/design_config.xml
CP Menu	MENU_CONFIG	/hsphere/local/home/cpanel/shiva/psoft/hsphere/menu.xml
CP Crons	CRON_CONFIG	/hsphere/local/home/cpanel/shiva/psoft/hsphere/cron_config.xml
Online (Context) Help	HELP_CONFIG, ONLINE_HELP_CONFIG	/hsphere/local/home/cpanel/shiva/psoft/hsphere/help.xml /hsphere/local/home/cpanel/shiva/psoft/hsphere/help_url.xml
Promotion Validators And Calculators	PROMO_CONFIG	/hsphere/local/home/cpanel/shiva/psoft/hsphere/promotion/xml/promotions.xml
Merchant Gateways	MERCHANT_GATEWAYS_CONF	/hsphere/local/home/cpanel/shiva/psoft/hsphere/merchants.xml
Domain Registrars	REGISTRAR_CONF	/hsphere/local/home/cpanel/shiva/psoft/hsphere/registrar.xml
E-Mail Notifications	USER_EMAILS	/hsphere/local/home/cpanel/shiva/psoft/hsphere/user_emails.xml

XML Customization Step By Step

Custom XML files can be merged with default XML files by means of *XML merger*. Instead of moving and changing a default XML file, a small custom file is created containing only the changes to be implemented, and its location is specified in `hsphere.properties` in the parameter with the "CUSTOM_" prefix added to the default parameter name. For example:

```
MENU_CONFIG = /hsphere/local/home/cpanel/shiva/psoft/hsphere/menu.xml
CUSTOM_MENU_CONFIG = /hsphere/local/home/cpanel/shiva/custom/xml/menu.xml
```

Go through the following steps to customize your XML configuration files:

1. Login as *cpanel user*.
2. Create a directory for custom XML configuration files if it does not exist, for example, `~cpanel/custom/xml`.
3. In the custom directory, create a custom XML file if it hasn't been created yet. Here, you add only those tags that need to be added or modified with relation to the default XML file. Please follow the rules for *merging XMLs*.

   For example, if you need just to add a new item to the menu, the custom menu.xml file will look like:

```
<?xml version="1.0"?>
<!DOCTYPE config [
 <!ELEMENT config (menus,interface)>
  <!ELEMENT menus (menu+)>
  <!ELEMENT menu (menuitem*,initmenu*)>
  <!ELEMENT menuitem (#PCDATA)>
  <!ELEMENT initmenu (#PCDATA)>
  <!ELEMENT interface (menudef+)>
  <!ELEMENT menudef (initmenu*,menuitem*)>


  <!ATTLIST menudef id CDATA #REQUIRED>

  <!ATTLIST menu name CDATA #REQUIRED>
  <!ATTLIST menu label CDATA #REQUIRED>
  <!ATTLIST menu platform_type CDATA "">
  <!ATTLIST menu resource CDATA "">
  <!ATTLIST menu defaultitem CDATA #REQUIRED>
  <!ATTLIST menu tip CDATA "">

  <!ATTLIST menuitem name CDATA #REQUIRED>
```



```
  <!ATTLIST menuitem label CDATA #REQUIRED>
  <!ATTLIST menuitem URL CDATA #REQUIRED>
  <!ATTLIST menuitem platform_type CDATA "">
  <!ATTLIST menuitem resource CDATA "">
  <!ATTLIST menuitem tip CDATA "">
  <!ATTLIST menuitem check_type CDATA "1">
  <!ATTLIST menuitem new_window CDATA "0">

  <!ATTLIST initmenu name CDATA #REQUIRED>
]>

<config>
<menus>
<menu name="info" label="info.label" defaultitem="info-plans" tip="info.tip">
  <menuitem name="new_item" label="NEW PAGE" URL="/newpage.html" resource="" tip="Positive Software Corporation"/>
</menu>
</menus>
</config>
```

IMPORTANT:

In the custom XML file to be merged with the default one, you must define the same DTD structure!

4. In `~cpanel/shiva/psoft_config/hsphere.properties`, add the location for the custom XML file, for instance:

   CUSTOM_MENU_CONFIG = /hsphere/local/home/cpanel/shiva/custom/xml/menu.xml
5. Login as root (log off from cpanel) and *restart H-Sphere*.

