



PLESK 7

CREATING AND INSTALLING CUSTOM SKINS

Distribution of this work or derivative of this work in any form is prohibited unless prior written permission is obtained from the copyright holder.

Linux is a registered trademark of Linus Torvalds. RedHat is a registered trademark of Red Hat Software, Inc. All other trademarks and copyrights are the property of their respective owners.

13800 Coppermine Road, Suite 112, Herndon, VA, 20171 USA, Ph.: 703 815-5670, Fax.: 703 815-5675

Table of Contents

1. Introduction	1
What Is a Skin?	1
Areas of the User Interface	1
Files That Compose a Skin	2
Structure Of Skin Directory	3
2. Custom Skin Creation	5
Creating the Skin Directory	5
Customizable Properties	6
3. Custom Skin Installation	19
Manual Installation	19
Using the Installation Script	19
4. Creating and Installing Skin RPM Package	21
Creating a Temporary Build Directory	21
Creating the Installation Script	21
Creating the RPM Spec File	23
Building the RPM Package	23
Installing the Custom Skin RPM Package	24

Chapter 1. Introduction

This document is a guide to creating and installing *skins* - custom interface appearance styles - for Plesk. Here you can find the structure of skin directories as well as its contents description, instructions on how to create your own custom skin and how to install it and make useable on your server.

What Is a Skin?

In Plesk a skin is in fact a set of CSS and image files. The CSS files define the style of the Plesk interface elements; the image files are the Plesk interface icons, logotype images and other pictures, used in CSS files. All these files, placed in corresponding sub-directories, compose the structure of the *skin directory*.

IMPORTANT

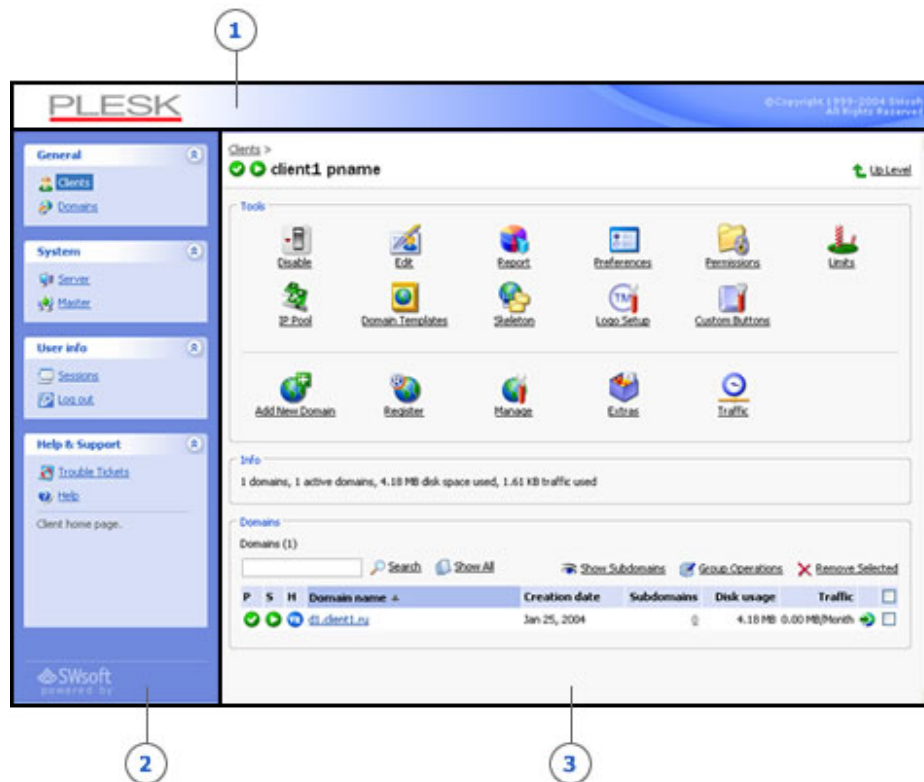
The development and/or modification of a skin require a strong knowledge of Cascading Style Sheets [<http://www.w3.org/Style/CSS/>]. It is absolutely necessary due to the fact that the process of creating a custom skin is largely a matter of editing selectors in CSS files.

Skins are an easy and flexible way to diversify your Plesk user interface appearance. Using skins you can change the colors of the interface areas, set new fonts properties, use different images for icons in the interface, etc.

It takes only a few clicks to replace one skin with another. Different skins can be used by different users at one server.

Areas of the User Interface

The Plesk user interface can logically be split into three parts: *top area*, *left (navigation) area* and *main area*.



1. *top area* contains the logotype image
2. *left (navigation) area* contains navigation items and context help area
3. *main area* contains the groups of available operations (based on current context), input forms, lists, and other similar interface elements

Each such area allows for individual customization of appearance within a skin.

Files That Compose a Skin

custom.css and layout.css

Each interface area has the corresponding two CSS files describing its appearance:

- *custom.css* contains selectors for visual properties (color, font, etc.) of the user interface elements
- *layout.css* contains selectors that define the layout of the user interface elements

i NOTE

The option of modifying *layout.css* is recommended only for the advanced CSS designers.

The *custom.css* and *layout.css* in the *help/* directory within the skin directory

define the appearance of the Help pages.

buttons.css

Additionally the main area uses file *buttons.css*, which defines the appearance of certain buttons in the user interface. For example, in the *XP-skins* it defines what images are used for the icons in the Tools groups.

This file is not a requirement and is not needed if the appearance of multiple buttons is not redefined in the skin. *buttons.css* is addressed from *main/custom.css*, its contents were separated only for the sake of ease of use.

general.css

The file *general.css* contains style settings general for all interface elements. The style specified here will be applied when displaying an interface element unless it was specifically redefined for the corresponding area of the user interface.

Image files

Image files are stored in two directories:

- *icons/* contains image files required for the user interface (state/status icons, list operations, etc.)
- *images/* contains image files used with the specific skin for customizing elements, set of these can be different for different skins. Links to these images are provided in the CSS files

Structure Of Skin Directory

The skin directories are located in */usr/local/psa/admin/htdocs/skins/*

The structure of the skin directory:

- *custom_skin/* - a custom skin directory
 - *css/* - all CSS files
 - *top/*
 - *custom.css*
 - *layout.css*
 - *left/*
 - *custom.css*
 - *layout.css*
 - *main/*
 - *custom.css*
 - *layout.css*
 - *help/*

- *custom.css*
- *layout.css*
- *general.css*
- *icons/* - all of the Plesk interface icons
- *images/* - all image files, referenced in the CSS files

Chapter 2. Custom Skin Creation

This chapter provides instructions on how to compose a new skin. The first main step in this process is the creation of the structure of the skin directory along with all its files, another is the actual customization of the visual appearance of elements of the user interface by editing the properties in the corresponding CSS files of the skin.

Creating the Skin Directory

In order to create the skin directory you need to create the structure of directories along with the corresponding CSS files as described in the section Structure Of Skin Directory. When this task is complete, you can proceed to editing the style properties.

Using Existing Skin As Template

In order to speed up and simplify the creation of the skin directory you can make use of an already existing skin (one of the default ones) installed on your server as a template for your own custom skin.

Create your future skin directory (e.g. my_skin):

```
# mkdir ~/my_skin
```

Copy to this directory one of the default skins:

NOTE

The directory where all skins are located in Plesk is
/usr/local/psa/admin/htdocs/skins/.

```
# cp -r /usr/local/psa/admin/htdocs/skins/winxp.blue/*  
~/my_skin
```

At this point you will have in your skin directory (*~/my_skin/*) the complete skin directory structure along with the CSS and image files of the Plesk default skin *winxp.blue*.

The skin template is ready to be used. Now you can begin editing the CSS files and adding in the necessary image files within the skin directories creating your unique style of Plesk user interface appearance.

Customizable Properties

Every type of element of the user interface that can be customized is described by the corresponding selectors within the appropriate CSS files of the skin. This section considers the areas of the user interface and gives the listing of these selectors and elements they correspond to, as well as provides simple examples of using these selectors in the CSS files.

General

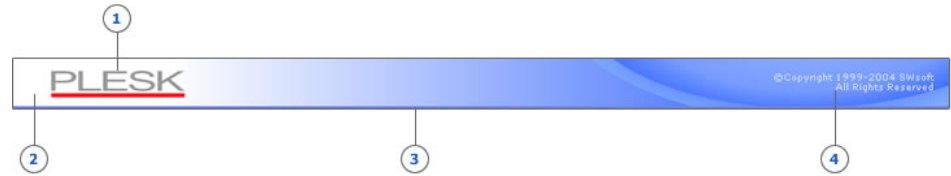
The properties that are common for all areas of interface, assigned in the file *general.css*.

Table 2.1. General properties

UI Element	Selector	CSS code sample
common background, font	body, td, th	<pre>body { font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif; font-size: 11px; font-weight: normal; color: #000000; background-color: #f6f6f6; } td, th { font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif; font-size: 11px; }</pre>
form elements	input, select, textarea	<pre>input, select, textarea { font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif; font-size: 11px; }</pre>
link	a	<pre>a:link, a:visited, a:hover { color: #0240a3; }</pre>

Further, for the specific elements, these properties can be re-defined as desired.

Top Area



1. *logotype image*
2. *background*
3. *frame separator line*
4. *top left image*

Table 2.2. Top area properties

N	Selector	CSS code sample
1.	can be set through the user's interface, but the default image is contained in the skin (<i>images/def_plesk_logo.gif</i>)	-
2, 3.	body	<pre>body { background-color: #ffffff; background-image: url(../images/top_bg.jpg); background-repeat: repeat-x; background-position: left bottom; }</pre>
4.	.body	<pre>.body { background-image: url(../images/top_body_bg.jpg); background-repeat: no-repeat; background-position: top right; }</pre>

Left Navigation Area



1. *background*
2. *navigation sections header background*
3. *navigation sections header*
4. *expand/collapse navigation section*
5. *navigation section area*
6. *navigation item*
7. *selected navigation item*
8. *logged in user info*
9. *context help*
10. *'powered by' logotype image*

Table 2.3. Left navigation area properties

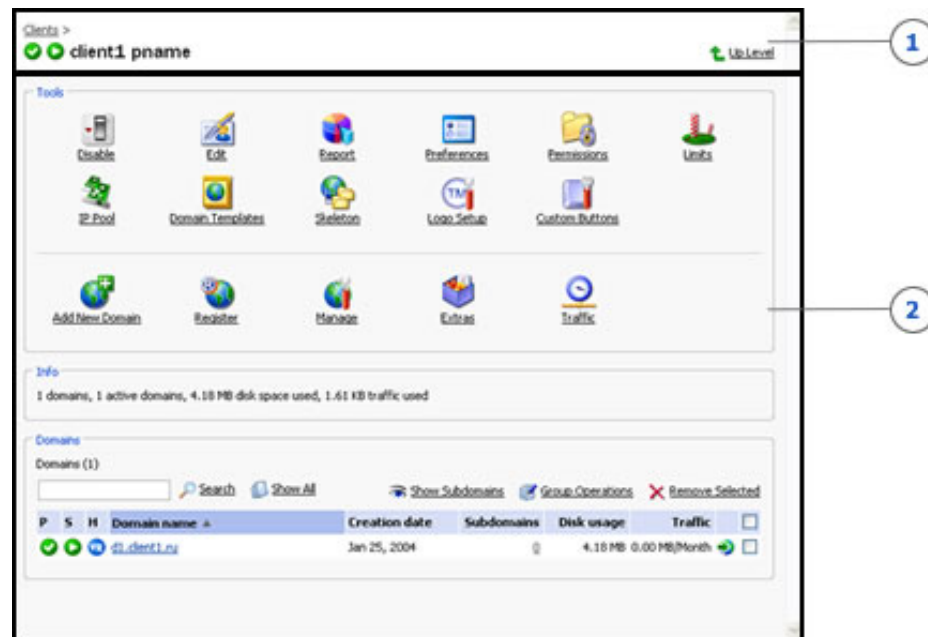
N	Selector	CSS code sample
1.	body	<pre>body { background-color: #6e89dd; }</pre>
2.	.navOpened .navClosed	<pre>.navOpened, .navClosed { background-color: #ffffff; }</pre>

N	Selector	CSS code sample
3.	<pre>.navOpened .navTitle .titleText .navClosed .navTitle .titleText</pre>	<pre>.navOpened .navTitle .titleText, .navClosed .navTitle .titleText { color: #215dc6; }</pre>
3.	<p>Mouse over</p> <pre>.navOpened .navTitleOver .titleText .navClosed .navTitleOver .titleText</pre>	<pre>.navOpened .navTitleOver .titleText, .navClosed .navTitleOver .titleText { color: #428eff; }</pre>
4.	<pre>.navTitle .titleHandle</pre>	<pre>.navTitle .titleHandle { background-color: #215dc6; }</pre>
4.	<p>Mouse over</p> <pre>.navTitleOver .titleHandle</pre>	<pre>.navTitleOver .titleHandle { background-color: #428eff; }</pre>
5.	<pre>.tree</pre>	<pre>.tree { background-color: #d6dff7; }</pre>
6.	<pre>.name</pre>	<pre>.name a:link, .name a:visited, .name a:active { color: #215dc6; } .name a:hover { color: #428eff; }</pre>
7.	<pre>.nodeActive .name</pre>	<pre>.nodeActive .name { background-color: #3878bf; } .nodeActive .name a:link, .nodeActive .name a:hover, .nodeActive .name a:visited, .nodeActive .name a:active { color: white; }</pre>

N	Selector	CSS code sample
8.	#userInfo	#userInfo { color: #555555; }
9.	#contexthelp	#contexthelp { color: #555555; border-top: 1px solid #A7B8EB; }
10.	body	body { background-image: url(../images/powered_by.gif); background-position: left bottom; background-repeat: no-repeat; }

Main Area

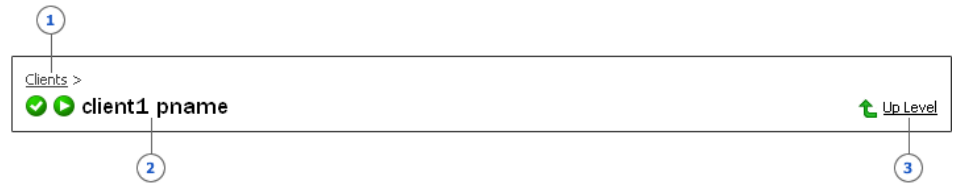
The main area consists of two smaller areas:



1. *screen title* - the title of the currently displayed screen
2. *screen content* - the currently available (visible) set of operations, input forms, lists, etc.

Following is the description of sub-areas that compose the main area and of their elements in detail.

Screen Title

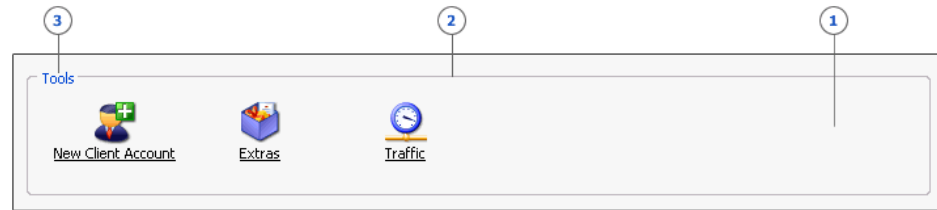


1. *path bar*
2. *title*
3. *'up level' link*

Table 2.4. Screen title properties

N	Selector	CSS code sample
1.	.pathbar	<pre>.pathbar { background: #ffffff; color: #444444; } .pathbar a:link, .pathbar a:visited, .pathbar a:hover { color: #444444; }</pre>
2.	.screenTitle	<pre>.screenTitle { background: #ffffff; border-bottom: 1px solid #cccccc; } .screenTitle td { font-size: 18px; font-family: "Franklin Gothic Medium", Verdana, Arial, sans-serif; color: #000000; }</pre>
3.	.uplevel .commonButton span icon can be changed using #bid-up-level in buttons.css	<pre>.uplevel .commonButton span { text-decoration: underline; } in buttons.css: #bid-up-level span { background-image: url(../images/btn_uplevel_bg.gif); }</pre>

General Screen Content



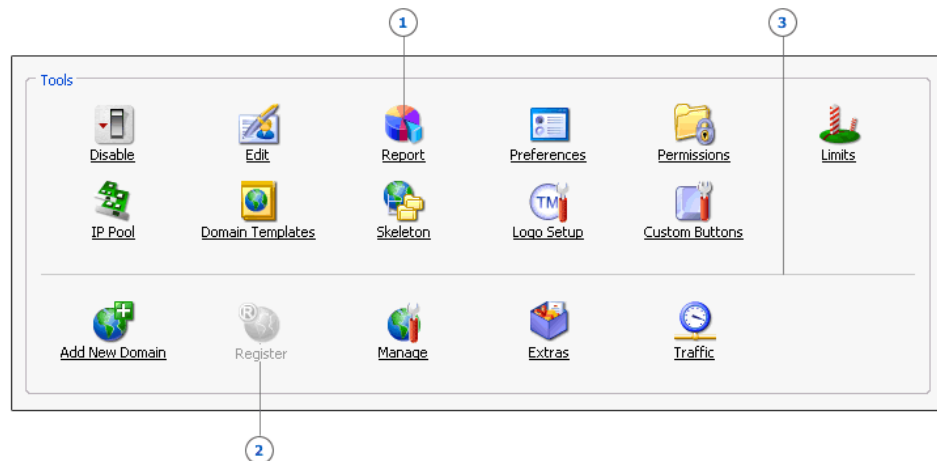
1. *screen content background*
2. *fieldset for grouping ui elements*
3. *fieldset title*

Table 2.5. General screen content properties

N	Selector	CSS code sample
1.	body	<pre>body { background: #F9F8F8; }</pre>
2.	fieldset	<pre>fieldset { }</pre> <p>Presently not available, using default value.</p>
3.	legend	<pre>legend { color: #0046D5; }</pre>

Tools

The set of operations at the current screen:



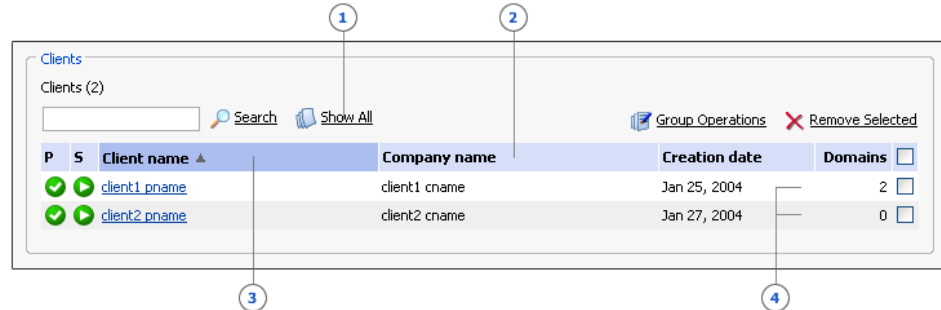
1. *tool*
2. *tool (disabled)*
3. *separator*

Table 2.6. Tools properties

N	Selector	CSS code sample
1.	<p><code>.toolsArea .commonButton</code></p> <p>icons are customized through id's (e.g. <code>#bid-report</code>) in <code>buttons.css</code></p>	<pre>.toolsArea .commonButton { text-decoration: underline; } in buttons.css: #bid-report { background-image: url(../../images/btn_report_bg.gif); }</pre>
2.	<p><code>.toolsArea span.commonButton</code></p> <p>icons are customized through id's (e.g. <code>#bid-report</code>) in <code>buttons.css</code></p>	<pre>.toolsArea span.commonButton { color: #999999; text-decoration: none; } in buttons.css: #bid-register-disabled { background-image: url(../../images/btn_register-disabled_bg.gif); }</pre>
1.	<p><code>hr</code></p>	<pre>hr { color: #cccccc; background-color: #cccccc; height: 1px; }</pre>

Lists

The list of objects:



1. *operations on lists*
2. *table header*
3. *table header (list sorted by selected parameter)*
4. *table's rows*

Table 2.7. Lists properties

N	Selector	CSS code sample
1.	<p>.buttons .commonButton span</p> <p>icons are customized through id's (e.g. #bid-report) in buttons.css</p>	<pre>.buttons .commonButton span { text-decoration: underline; } in buttons.css: #bid-show-all span { background-image: url(../../images/btn_show-all_bg.gif); }</pre>

N	Selector	CSS code sample
2.	th	<pre>th { text-align: left; background: #D6DFF7; border-right: 1px solid #ffffff; border-bottom: 1px solid #ffffff; } th a:link, th a:visited { color: #000000; text-decoration: none; } th a:hover { text-decoration: underline; }</pre>
3.	.sort	<pre>.sort { background-color: #ABBEEF; }</pre>
4.	.oddrowbg - for odd rows .evenrowbg - for even rows	<pre>.evenrowbg { background-color: #F0F0F0; } .oddrowbg { background-color: #ffffff; }</pre>

Dialog Forms

The screenshot shows a dialog box titled "Client form" with the following fields and controls:

- Company name
- Contact name *
- Login *
- Password *
- Confirm Password *
- Phone
- Fax
- E-mail
- Address
- City
- State/Province
- Postal/ZIP code
- Country (dropdown menu, currently showing "United States")
- Interface language (dropdown menu, currently showing "English")
- Select template (dropdown menu, currently showing "Create client without template")
- Proceed to client's IP pool configuring
- * Required fields (footnote)
- OK button
- Cancel button

1. *parameter name*
2. *'required' indicator*
3. *footnote*
4. *button*

Table 2.8. Dialog forms properties

N	Selector	CSS code sample
1.	.name	.name { font-weight: bold; color: #555555; }
2.	.required	.required { color: #cc0000; }
3.	.footnote	.footnote { color: #666666; }

N	Selector	CSS code sample
4.	<p>.commonButton</p> <p>.buttons .commonButton span</p> <p>icons are customized through id's (e.g. #bid-report) in buttons.css</p>	<pre>.commonButton button { font-family: Tahoma, Verdana, Arial, Helvetica, sans-serif; font-size: 11px; color: #000000; background-color: transparent; background-image: url(../images/btn_bg.gif); border: 0 solid white; background-repeat: no-repeat; } in buttons.css: #bid-ok button { background-image: url(../images/btn_ok_bg.gif); padding-left: 8px; }</pre>

Help

The Help pages properties that can be customized:

Table 2.9. Help properties

UI Element	Selector	CSS code sample
Header level 1	h1	<pre>h1 { font-size: 16px; }</pre>
Header level 2	h2	<pre>h2 { font-size: 14px; }</pre>

Chapter 3. Custom Skin Installation

There are two ways you can install your custom skin. One is the manual installation – this method requires of you only a certain knowledge and experience in working with databases and files, it is recommended to more advanced users. The other one requires you to download the installation script, yet it is less complicated then the first one.

Manual Installation

In order to install a skin, the first thing you should do is place the whole created structure of sub-directories and files to a directory `htdocs/skins/<skin_name>`. After that you must let Plesk know that it is there.

NOTE

You must be logged in as user `root` to install the skin.

Plesk learns what skins are available from the 'Skins' table of its database 'psa'. So, in order to make a custom skin available for using, you need to add a corresponding record to this database. A record `<skin_name>`, `<source-dir>` must be added to the 'Skins' table. It can be done like this for example (for the custom skin 'my_skin'):

```
insert into Skins (name, place) values ('My  
skin', '/skins/my_skin');
```

Next, you need to set proper ownership and access privileges for the directories and files of the skin:

Use the utilities `chmod` and `chown` to set permissions for all files of the skin to `444`, permissions for all directories of the skin to `555`, and the ownership of files and directories to user `root`, group `psadm`.

The custom skin is installed. You can now apply it from the Plesk user interface.

Using the Installation Script

Another way to install a custom skin is to make use of the installation script. The script will automatically perform all actions required for installing the script, all you need to do is execute it with the proper parameters.

The skin installation script can be downloaded from the SWsoft, Inc. web site:

ftp://download1.sw-soft.com/Plesk/Plesk7/extensions/skin_install.sh

Download the installation script to your Plesk server.

 NOTE

You must be logged in as user *root* to install the skin.

Execute the following command:

```
# sh skin_install.sh <skin-name> <source-dir>
```

Here, <skin-name> is the name of your skin (e.g. 'My skin'), <source-dir> is the location of your custom skin (e.g. ~/my_skin).

The custom skin is installed. You can now apply it from the Plesk user interface.

Chapter 4. Creating and Installing Skin RPM Package

If you plan to often install your custom skin on Linux servers it is advisable to build an RPM package of your custom skin. This way it can be easily distributed and installed. Follow the instructions in this section to build the RPM package for the skin you created.

Creating a Temporary Build Directory

The first thing you need to do is to create a temporary build directory. It can be located anywhere on your hard disk. Let it be `/tmp/skin_builds/` for example:

```
# mkdir /tmp/skin_build
```

Next, inside the temporary build directory create the complete path to where the skin will be located when installed in Plesk.

NOTE

The directory where all skins are located in Plesk is `/usr/local/psa/admin/htdocs/skins/`.

Thus in our example you need to create the directory `/tmp/skin_build/usr/local/psa/admin/htdocs/skins/my_skin/`, where `my_skin` is your custom skin directory name.

Copy the contents of your skin to this location. For example, assuming that your custom skin directory is `~/my_skin/`:

```
# cp ~/my_skin/*  
/tmp/skin_build/usr/local/psa/admin/htdocs/skins/my_skin/
```

When you are done you should have the temporary build directory (`/tmp/skin_build/`) contain your custom skin just as it will be located on server once installed.

Creating the Installation Script

The installation script is used for updating appropriate records in Plesk database when the skin is installed (uninstalled). You need to create such a script for your own custom skin.

Below is a sample installation script that could be used for the skin considered in our example:

```
#!/bin/sh
admin_passwd=`cat /etc/psa/.psa.shadow`
info="$1"
errorlog=`mktemp /tmp/psa_my_skin_install.log`
case $1 in
    install)
        echo "=====>install skin" >>"$errorlog"
        mysql -uadmin "-p$admin_passwd" >>"$errorlog"
2>&1 psa <<-EOF
            insert into Skins(name,place)
values('My Skin','/skins/my_skin');

            EOF
            ;;
    uninstall)
        echo "=====>take old id" >>"$errorlog"
        query="select id from Skins where
place='/skins/my_skin';"
        old_id=`echo "$query"|mysql -B -r -uadmin
"-p$admin_passwd" psa 2>>"$errorlog"|tail +2`
        echo "=====>take new id" >>"$errorlog"
        info="$info old_id=$old_id"
        query="select id from Skins where
place<>' /skins/my_skin' limit 1;"
        new_id=`echo "$query"|mysql -B -r -uadmin
"-p$admin_passwd" psa 2>>"$errorlog"|tail +2`
        info="$info new_id=$new_id"
        echo "=====>uninstall skin" >>"$errorlog"
        mysql -uadmin "-p$admin_passwd" psa
>>"$errorlog" 2>&1 <<-EOF
            delete from Skins where id=$old_id;
            update misc set val='$new_id' where
param in ('def_skin_id','admin_skin_id') and
val='$old_id';
            update mn_param set val='$new_id' where
param='skin_id' and val='$old_id';
            update cl_param set val='$new_id' where
param='skin_id' and val='$old_id';
            EOF
            ;;
    *)
        echo "$0 {install|uninstall}" >/dev/stderr
        exit 1
        ;;
esac

if test "$?" != 0; then
    echo " An error occured while
(un)registering theme in Plesk database."
    echo " please update the database
manually"

    echo -n " info: $info "
    cat "$errorlog"
    true
else
    rm "$errorlog"

```

fi

Creating the RPM Spec File

The RPM spec (specification) file contains data required for building the RPM package. Below is a sample spec file that could be used for the skin considered in our example. In this sample file:

- SKINVERSION is the version number of the skin,
- SKINRELEASE is the release number of the skin,
- License type can be GPL, Freeware or other common type,
- SCRIPTNAME is the name of the installation script described in the previous section.

```
Name: my_skin
Version: SKINVERSION
Release: SKINRELEASE
License: Commercial
Group: Networking/Daemons
Requires: psa = 7.0.0
Summary: My cool custom skin
BuildRoot: /tmp/skin_build

%description
plesk 7 new skin

%post
if test "$1" = 1; then
    sh
    /usr/local/psa/admin/htdocs/skins/my_skin/SCRIPTNAME
install
fi

%preun
if test "$1" = 0; then
    sh
    /usr/local/psa/admin/htdocs/skins/my_skin/SCRIPTNAME
uninstall
fi

%files
%defattr(-,root,root)
/usr/local/psa/admin/htdocs/skins/my_skin
```

To complete preparing the spec file for the custom skin from our example (my_skin), the above items need to be replaced with proper values.

Building the RPM Package

Once you have the temporary build directory with the skin files all set and the installation script and spec file complete you can proceed to build the RPM

package for your skin.

First, copy the installation script and the spec file to the skin directory in the temporary build directory. In our example:
`/tmp/skin_build/usr/local/psa/admin/htdocs/skins/my_skin/`.

NOTE

You must be logged in as root to build the RPM package.

Execute the following command:

```
# rpmbuild -bb  
/tmp/skin_build/usr/local/psa/admin/htdocs/my_skin/SKIN.spec
```

Here SKIN.spec should be replaced with the name of your custom skin spec file described in the previous section.

Once the process of building the RPM package is complete you will find the package (in our example it will be `my_skin.rpm`) in the following directory:
`/usr/src/redhat/RPMS/noarch`.

Installing the Custom Skin RPM Package

Upload the custom skin RPM package to Plesk server. Execute the following command (we consider the `my_skin.rpm` from our example):

```
# rpm -Uvh my_skin.rpm
```

The custom skin `my_skin` is installed.

In case of problems with the installation, refer to the log file `/tmp/psa_my_skin_install.log` for information.

NOTE

This log file is specified in the installation script.