

GUI Preferences

GUI Preferences - <guiprefs>

This <guiprefs> element allows you to set the layout and behavior of the GUI when the installer runs. This information will not have any effect on the command-line installers that will be available in future versions of IzPack. The attributes that can be specified are:

Attribute Name	Description	Default Value
resizable	Indicates whether the window size can be changed or not.	no
width	Sets the initial window width in pixels.	600
height	Sets the initial window height in pixels.	480

Example:

```
<guiprefs width="800" height="600"  
resizable="no" />
```

Nested Elements

<modify> - Modifying the GUI

There are some options to modify the graphic user interface. Most of them are managed with key/value pairs of the <modifier> element which is a child of the <guiprefs> element in the installation description file.

Example:

```
<guiprefs width="600" height="480"
resizable="no">
  <modifier key="useButtonIcons"
value="no"/>
  <modifier key="useLabelIcons" value="no"/>
  <modifier key="labelGap" value="2"/>
  <modifier key="layoutAnchor"
value="NORTHWEST"/>
  <modifier key="useHeadingPanel"
value="yes"/>
  <modifier key="headingImageOnLeft"
value="yes"/>
  <modifier key="headingLineCount"
value="1"/>
  <modifier key="headingFontSize"
value="1.5"/>
  <modifier key="headingBackgroundColor"
value="0x00ffffff"/>
  <modifier key="headingPanelCounter"
value="text"/>
  <modifier key="headingPanelCounterPos"
value="inHeading"/>
</guiprefs>
```

Modifying the Language Selection Dialog

The language selection dialog appears at the start of the installation before the first panel occurs.

In addition to the picture in the language selection dialog, it is possible to modify flags and the way the language name is shown:

- **useFlags** Possible are "yes" or "no". Default is "yes". If it is set to "no", no flag will be displayed in the language selection dialog. For "no" it is recommended to define also 'langDisplayType' other than "iso3".
- **langDisplayType** Possible values are "iso3", "native" and "default". Default is "iso3". With "iso3" the text for a language will be displayed as ISO 639-2:1998 code. With "native" the notation of the language will be used if possible, else the notation of the default locale. Using "default" will be presented the language in the notation of the default locale of the VM.

Modifying IzPack Panels

There are some graphic elements and behavior which are preferred by some people and deprecated by other. The following keys are related to the whole installation (all panels).

- **useButtonIcons**: possible are "yes" or "no". Default is "yes". If it is set to "no", all buttons which are created via the ButtonFactory contains no icon also a icon id was submitted. Directly created buttons are not affected.
- **useLabelIcons** Possible are "yes" or "no". Default is "yes". If it is set to "no", all labels which are created via the LabelFactory contains no icon also a icon id was submitted. Directly created labels are not affected.
- **labelFontSize** A float value used as a multiplier for the font size on labels created via the LabelFactory and IzPanel. Directly created labels are not affected.
- **layoutAnchor** This is the layout anchor for IzPanels. Valid are "NORTH", "NORTHWEST", "SOUTHWEST", "SOUTH" and "CENTER". Only panels which are using the layout helper of IzPanels are supported. These are not all standard panels. At developing custom panels it is recommended to use the layout helper with an IzPanelLayout. Note: The anchor definition will be used for all panels!
- **Gaps** There are a number of defined gaps between different components of a IzPanel that can be applied to a IzPanelLayout. The gaps can be set also via the element '<modifier>' of '<guiprefs>'. It is possible to declare different values for X and Y axis. This will be determined in the key word name. X Gaps are insert after Y gaps under the control for which the gap was declared. Following key words are defined:
 - **labelXGap | labelYGap** The gap in pixels between two labels in X or Y direction.
 - **textXGap | textYGap** The gap in pixels between two text fields.
 - **controlXGap | controlYGap** The gap in pixels between two controls other than label or textfield.
 - **paragraphYGap** The gap in pixels for a paragraph. A paragraph will be created in the panel source for controls which should be separated. **paragraphXGap is declared, but not used.???**
 - **labelToTextXGap | labelToTextYGap** The gap in pixels between a label (left or top) and a text field (right or bottom).
 - **labelToControlXGap | labelToControlYGap** The gap in pixels between a label (left or top) and a control other than a label or a textfield.
 - **textToLabelXGap | textToLabelYGap** The gap in pixels between a text field (left or top) and a label.
 - **controlToLabelXGap | controlToLabelYGap** The gap in pixels between a control other than a label or a text field and a label.
 - **controlToTextXGap | controlToTextYGap** The gap in pixels between a control other than a label or a text field and a text field.
 - **textToControlXGap | textToControlYGap** The gap in pixels between a text field and a control other than a label or a text field .
 - **firstYGap** The gap in pixels between the top border and the first control.
 - **fillerNXGap | fillerNYGap** The gap in pixels created by the layout manager **to do what???**. Fillers are used by some panels. **N** is a number between 1 and 5 to specify a different filler e.g. filler3XGap or filler1YGap.
 - **allXGap | allYGap** The gap in pixels between all controls in X or Y direction. If this is declared, this is the default for all gaps for which no own declaration has been defined.
- **layoutYStretchType | layoutXStretchType** This is the type of stretch that will be applied. The IzPanelLayout manager permits stretch factors for controls to be specified. This means, that a control will be stretched if

there is space in the line. The amount of stretching will be determined by the stretch factor. **But what to do if the whole stretch factor for a line or column is not 1.0?** To determine this these settings are exist. Valid values are

- **RELATIVE** The values will be normalized ????
- **ABSOLUTE** The values will be used as they are. A part of the line will be clipped if the sum is greater than 1.0.
- **NO** No stretch will be performed.
- **layoutFullLineStretch** | **layoutFullColumnStretch** There are controls which should be stretched. Beside fixed values there are the symbolic values FULL_LINE_STRETCH and FULL_COLUMN_STRETCH which are computed at layout. E.g. MultiLineLabels has this stretch factor for x direction. **But what to do if a centered layout is chosen?** With a control like this the lines will be stretch to the hole size. With this settings it can be changed. E.g. a factor of 0.7 creates a nice centered layout. The default is 1.0, valid are 0.0 up to 1.0.

Alternative Frame Title

It is possible to use an alternative frame title. Normally the title is "IzPack - Installation of " + '\$APP_NAME'. If the langpack key `installer.reversetitle` is defined, the value of that key will be used instead of the key `installer.title`. There is no string added, but it is possible to use IzPack variables. The **third heading example** contains such a alternatively frame title. It is only possible to use predefined variables like '\$APP_NAME' because the title will be created before the frame will be shown. It is common to use the name of the installation toolkit in the frame title.

Using a Separated Heading Panel

Some standard panels have headings (e.g. ShortcutPanel). These headings are integrated in the IzPanel. **In opposite to this following heading will be displayed in a separated panel potential for all panels with the same design** .(not parsable in English) There is no need to modify existing Java classes. The declaration of some key/value pairs is enough.

There can be one real head and zero or more info lines. The headline will be written bold, the font size can be changed. Info lines will be indented and written with the normal used font. The heading message has to be written into the langpack (or custom langpack) file with the key '<panel class name>.headline'. Examples can be seen in eng.xml. Maybe the entries for standard panels are not present in other languages. Messages for info lines have the key '<panel class name>.headinfo<info line number>'. First info line has number zero. If no or empty headline messages will be declared in the chosen language no heading panel will be shown. This behavior can be used to suppress heading for special panels.

It is also possible to declare head and info lines additional dependent on the 'panelid'. The result is, that it is possible to declare different messages for panels which are shown more than one time (e.g. the UserInputPanel. In this case the key for heading is

```
<panel class name>.headline.<panelid>
```

and for info lines

```
<panel class name>.headinfo<info line number>.<panelid>
```

Panel IDs are declared in the [<panel> element](#). The standard strings are declared in the standard langpack file. For customized panels it is common to declare text in the custom language pack.

Example:

```
<panels>
  ...
  <panel classname="UserInputPanel"
id="one"/>
  <panel
classname="UserInputPanel" id="two"/>
  ...
</panels>
```

Then the messages can be declared in 'CustomLangpack.xml_eng' like this:

```
<langpack>
  ...
  <str id="UserInputPanel.headline.one"
txt="User Data one"/>
  <str id="UserInputPanel.headline.two"
txt="User Data two"/>
  <str id="UserInputPanel.headinfo0.one"
txt="Info 1 one"/>
  <str id="UserInputPanel.headinfo1.one"
txt="Info 2 one"/>
  <str id="UserInputPanel.headinfo0.two"
txt="Info 1 two"/>
  <str id="UserInputPanel.headinfo1.two"
txt="Info 2 two"/>
  ...
</langpack>
```

It is possible to place an icon on the right side of the heading (see below to display on left side). To do this a simple resource entry will be needed:

```

<resources>
    ...
    <res id="Heading.image" src="[path to the
image in the source tree]"/>
    ...
</resources>

```

There are some **guiprefs** modifier keys used to modify heading (see above). Additionally it is possible to count the general not hidden panels in the heading or navigation panel.

- **useHeadingPanel** General switch for heading. If this key does not exist or does not have the value "yes" no heading panel will be shown.
- **headingImageOnLeft** Option to allow displaying the heading image on the left of the header instead of the default (right side). Only valid if heading panel is used.
- **useHeadingForSummary** In the language files there are entries for the heading text ('[Panel name](#).headline') and the summary caption ('[Panel name](#).summaryCaption'). If this modifier is set to "yes", the text of the heading will be also used for the summary caption.
- **headingLineCount** Number of heading lines. If no info lines should be shown the value should be one (not zero).
- **headingFontSize** A float value used as multiplier for the standard font size.
- **headingBackgroundColor** Background color of the heading panel as integer. Often used is 0x00ffffff (white).
- **headingForegroundColor** Font color of the heading panel as integer. Often used is 0x00ffffff (white).
- **headingPanelCounter** Draw a panel counting. Possible values are "text" or "progressbar". inHeading the progressbar will be not the best choice.
- **headingPanelCounterPos** Declares where the counter will be shown. Possible are "inHeading" or "inNavigationPanel". If "inNavigationPanel" is chosen, the panel counter can be used also no heading was selected.

Example: Modifiers to create an IzPack installation with heading, no button and label icons and a panel text counter in the heading panel:

```
<guiprefs width="600" height="480"
resizable="no">
  <modifier key="useButtonIcons"
value="no"/>
  <modifier key="useLabelIcons" value="no"/>
  <modifier key="labelGap" value="2"/>
  <modifier key="layoutAnchor"
value="NORTHWEST"/>
  <modifier key="useHeadingPanel"
value="yes"/>
  <modifier key="headingImageOnLeft"
value="yes"/>
  <modifier key="headingLineCount"
value="1"/>
  <modifier key="headingFontSize"
value="1.5"/>
  <modifier key="headingBackgroundColor"
value="0x00ffffff"/>
  <modifier key="headingPanelCounter"
value="text"/>
  <modifier key="headingPanelCounterPos"
value="inHeading"/>
</guiprefs>
```

Example: Changed resources and langpack keys to create IzPack installation with an alternative frame title, a heading, no button and label icons and a text counter in the heading panel.

install.xml

```
<installation version="1.0">
  ...
  <resources>
    ...
    <res src="border4.png"
id="Installer.image.3" />
    ...
  </resources>
</installation>
```

In the resource file `<ISO3>.xml` or `'CustomLangpack.xml_<ISO3>`, add:

```
<langpack>
  ...
  <str id="installer.reversetitle"
txt="$APP_NAME $APP_VER - IzPack Wizard " />
  ...
</langpack>
```

Example: Changed key/value pairs to create IzPack installation with heading, no button and label icons and a panel progressbar counter in the navigation panel.

```
<guiprefs width="640" height="480"
resizable="no">
  <modifier key="useButtonIcons" value="no"/>
  <modifier key="useLabelIcons" value="no"/>
  <modifier key="layoutAnchor"
value="NORTHWEST"/>
  <modifier key="labelGap" value="2"/>
  <modifier key="useHeadingPanel" value="yes"/>
  <modifier key="headingLineCount" value="1"/>
  <modifier key="headingFontSize" value="1.5"/>
  <modifier key="headingBackgroundColor"
value="0x00ffffff"/>
  <modifier key="headingPanelCounter"
value="progressbar"/>
  <modifier key="headingPanelCounterPos"
value="inNavigationPanel"/>
</guiprefs>
```

PackSize

The PacksPanel dialog supports the modifier **doNotShowPackSizeColumn**. With **doNotShowPackSizeColumn** set to true, the third column will not be shown. This does not affect the display of the required size of all packs.

The required size can be hidden by setting the **doNotShowRequiredSize** to true.

Example: Don't show pack size in PacksPanel.

```

<guiprefs width="640" height="480"
resizable="no">
    ...
    <modifier key="doNotShowPackSizeColumn"
value="true" />
    <modifier key="doNotShowRequiredSize"
value="yes" />
    ...
</guiprefs>

```

The PacksPanel will not show the column with the sizes of each pack, but will show the total required space.

Alternative Cancel Dialog

The cancel dialog will be shown if the **cancel** button or the **close** button of the frame is pushed.

In the standard dialog, the title contains the question and the confirmation message.

Often, in other dialogs, the title is a common heading and the question will be called in the dialog as a message. The standard behavior will be modified if the messages `installer.quit.reversemessage` or `installer.quit.reversetitle` are declared.

In the resource files `<ISO3>.xml` or `CustomLangpack.xml_<ISO3>`, add

```

<langpack>
    ...
    <str id="installer.quit.reversemessage"
txt="Are you sure you want to cancel
installation?"/>
    <str id="installer.quit.reversetitle"
txt="$APP_NAME $APP_VER"/>
    ...
</langpack>

```

<splash> - Adding a Splash Page

The `<splash>` element specifies the path to the image to be used as a splash screen. The path is relative to the installation root.

The image can be any bitmap format such as png, jpg, jpeg, gif or bmp.

Sample IzPack installation description

```
<installation version="5.0"
```

```
xmlns:izpack="http://izpack.org/schema/installation"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://izpack.org/schema/installation  
http://izpack.org/schema/5.0/izpack-installation-5.0.xsd">
```

```
  <info>
```

```
    <appname>Test</appname>
```

```
    <appversion>0.0</appversion>
```

```
    <appsubpath>myapp</appsubpath>
```

```
    <javaversion>1.6</javaversion>
```

```
  </info>
```

```
  <guiprefs width="800" height="600"  
resizable="no">
```

```
    <splash>images/peas_load.gif</splash>
```

```
  </guiprefs>
```

```
</installation>
```

The look and feel can be specified on a per-OS basis. For instance you can use the native look and feels on Win32 and OS X but use a third-party one on Unix-like platforms. To do that, you have to add some **<laf>** child elements to the **<guiprefs>** element.:

laf: This tag specifies a look and feel. It has a **name** attribute that defines the look and feel name.

Each **laf** element needs at least one **os** child element.

Each **laf** element can also contain any number of **param** elements to customize a look and feel. A **param** element has two attributes: **name** and **value**. The valid names and values will depend on the **laf** being described.

The available look and feels are:

- Kunststoff: `kunststoff`
- Liquid: `liquid`
- Metouia: `metouia`
- JGoodies Looks: `looks`
- Substance: `substance`
- Windows: `windows`
- Aqua: `aqua`
- Metal: `metal`

If you don't specify a look and feel for a particular operating system, then the default native one will be used: Windows on Windows, Aqua on Mac OS X and Metal on the Unix-like variants.

Sample IzPack installation description

```
<installation version="5.0"
```

```
  xmlns:izpack="http://izpack.org/schema/installation"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation="http://izpack.org/schema/installation
```

```
    http://izpack.org/schema/5.0/izpack-installation-5.0.xsd">
```

```
<info>
  <appname>Test</appname>
  <appversion>0.0</appversion>
  <appsubpath>myapp</appsubpath>
  <javaversion>1.6</javaversion>
</info>

<guiprefs width="800" height="600"
resizable="no">
  <splash>images/peas_load.gif</splash>
  <laf name="substance">
    <os family="windows" />
    <os family="unix" />
    <param name="variant"
value="mist-silver" />
  </laf>
  <laf name="substance">
    <os family="mac" />
    <param name="variant"
value="mist-aqua" />
  </laf>
  <modifier key="useHeadingPanel"
value="yes" />
```

```
</guiprefs>
```

```
</installation>
```

Liquid Look and Feel

The *Liquid Look and Feel* supports the following **variant** attributes:

`decorate.frames: yes` means that it will render the frames in Liquid style

`decorate.dialogs: yes` means that it will render the dialogs in Liquid style

JGoodies Looks

The *JGoodies Looks* look and feel can be specified by using the **variant** attributes. The values can be one of:

`windows`: use the Windows look

`plastic`: use the basic Plastic look

`plastic3D`: use the Plastic 3D look

`plasticXP`: use the Plastic XP look (default).

Here is a small sample:

```

<guiprefs height="600" resizable="yes"
width="800" splash="images/splash.png">
  <laf name="substance">
    <os family="windows" />
    <os family="unix" />
    <param name="variant"
value="mist-silver" />
  </laf>
  <laf name="substance">
    <os family="mac" />
    <param name="variant" value="mist-aqua"
/>
  </laf>
</guiprefs>

```

Substance 6.1

The *Substance* look and feel *toned-down* themes can be specified using the `variant` parameter, with the value being one of:

Attribute	PushingPixels Theme Name
<code>default</code>	<code>org.pushingpixels.substance.api.skin.SubstanceBusinessLookAndFeel</code>
<code>business</code>	<code>org.pushingpixels.substance.api.skin.SubstanceBusinessLookAndFeel</code>
<code>sahara</code>	<code>org.pushingpixels.substance.api.skin.SubstanceSaharaLookAndFeel</code>
<code>business-blue</code>	<code>org.pushingpixels.substance.api.skin.SubstanceBusinessBlueSteelLookAndFeel</code>
<code>business-black</code>	<code>org.pushingpixels.substance.api.skin.SubstanceBusinessBlackSteelLookAndFeel</code>
<code>creme</code>	<code>org.pushingpixels.substance.api.skin.SubstanceCremeLookAndFeel</code>
<code>creme-coffee</code>	<code>org.pushingpixels.substance.api.skin.SubstanceCremeCoffeeLookAndFeel</code>

graphite	org.pushingpixels.substance.api.skin.SubstanceGraphiteLookAndFeel
moderate	org.pushingpixels.substance.api.skin.SubstanceModerateLookAndFeel
nebula	org.pushingpixels.substance.api.skin.SubstanceNebulaLookAndFeel
nebula-brick-wall	org.pushingpixels.substance.api.skin.SubstanceNebulaBrickWallLookAndFeel
autumn	org.pushingpixels.substance.api.skin.SubstanceAutumnLookAndFeel
mist-silver	org.pushingpixels.substance.api.skin.SubstanceMistSilverLookAndFeel
mist-aqua	org.pushingpixels.substance.api.skin.SubstanceMistAquaLookAndFeel
dust	org.pushingpixels.substance.api.skin.SubstanceDustLookAndFeel
dust-coffee	org.pushingpixels.substance.api.skin.SubstanceDustCoffeeLookAndFeel
gemini	org.pushingpixels.substance.api.skin.SubstanceGeminiLookAndFeel
mariner	org.pushingpixels.substance.api.skin.SubstanceMarinerLookAndFeel
officesilver	org.pushingpixels.substance.api.skin.SubstanceOfficeSilver2007LookAndFeel
officeblue	org.pushingpixels.substance.api.skin.SubstanceOfficeBlue2007LookAndFeel
officeblack	org.pushingpixels.substance.api.skin.SubstanceOfficeBlack2007LookAndFeel

The Substance web site has been taken off line but is still available in the Substance project source code found at <http://java.net/projects/substance/sources/svn/show/trunk/www>.

To view the gallery of the different toned-down themes, you need to check out the source code and then navigate to the `www/docs/skins/toneddown.html` file from your browser..

Operating System Restrictions

The `<laf>` element can be restricted to operating system conditions by a nested `<os>` element, in which case the look and feel applies only in case the OS conditions apply

See [OS Restrictions](#) describes the use of `<os>` in more detail.