ShopFactory™

SF Developer Guidelines V 9.05

June 2011

3D3.COM Pty Ltd

You can modify existing templates within ShopFactory, using Customize Design mode with point and click ease.

These Development guidelines define how templates can be created and customized on code level for website designers using ShopFactory 8 and 9.

This guide requires at least a basic understanding of HTML and CSS coding in text editors.

WARNING:

Do not use HTML or style sheet editors which reformat or add their own code.
You will break the templates and ShopFactory will not work with them.

Table of Contents

How SF creates new shops

ShopFactory uses the data entered by the user and combines it with a number of design templates to create a new shop project.

New shop Projects are usually created in:

```
C:\Documents and Settings\...\My Documents\ShopFactory V9 Websites
```

In this folder you will find a copy of the templates used as well as the Runtime folder, which contains the pages created by combining templates and data – i.e. the actual shop.

It also contains the database file containing the data entered by the user.

Back up this folder to safeguard your project.

Advanced themes introduction

Themes are designed to empower the users to customize a website a great deal with point and click ease internally. See How to create a new template and Theme generation quick guide for more details.

Advanced users may also externally change widths, heights, colours, images, spacing, fonts, etc. – provided the files have been set up correctly.

To make this possible, the code contains special SF tags, which allows the software to modify settings in the style sheets.

The software contains a sophisticated style sheet editor, which works based on the SF tags included in the various files which make up a website theme.

If you are familiar with XHTML, CSS and XML coding, this guide will allow you to further manipulate your website theme.

How ShopFactory Website themes and templates work

A ShopFactory Website theme is the lay-out design which defines the look of a website created with ShopFactory. It must support all the design elements and content elements supported by ShopFactory, as well as the customize design functions built into ShopFactory

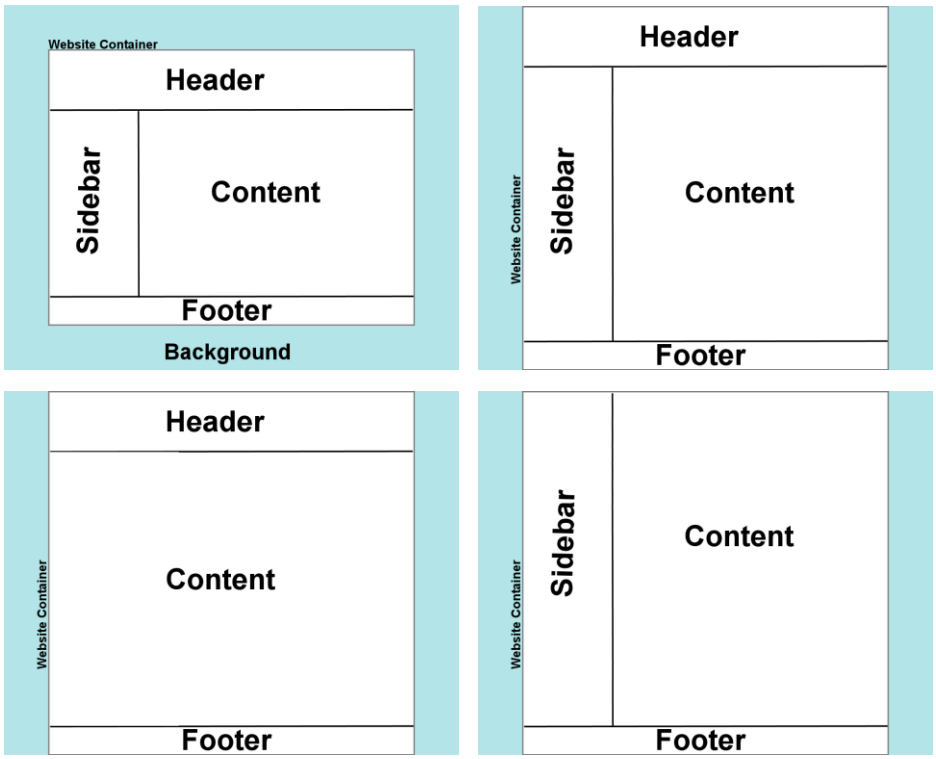
A ShopFactory website theme splits a website into multiple areas. In these areas the different website elements are displayed. Some areas are optional; other areas always have to exist, and of course you can add additional areas for design purposes.

Different templates are used to create the content of the different website areas.

The Content area is also split into several regions – see The Content Area for details.

Some possible layouts are shown below, but of course Templates can be arranged in many different ways.

If an optional lay-out area is not used, the website elements must be added to other lay-out areas.



You can add extra areas around the content area to achieve specific design goals, such as completely enclosing the Content area on all sides.

Website Lay-Out areas

The available Website lay-out areas are: Background, Website Container, Header, Sidebar, Footer, and Content

The following areas are always required:

Always required: Background, Website Container, Content, Footer

The following areas are optional, although you would usually use at least one of the two areas:

Optional: Header, Sidebar,

All elements which can go into the Header and Sidebar are interchangeable; therefore you can exclude the Header or the Index, as long as you place all elements into other areas. As all Website

elements can also be added to the Footer, only convention stops you from removing the Sidebar and the Header.

You can add extra areas for design purposes, for example to add design images to them.

Website Container

The Website container contains the complete website – it is like a parent frame around all other website lay-out areas.

Header

This area usually contains the following items:

- the website title
- a company logo
- the website slogan
- Index 1, which usually links to the special pages such as About us, Home, Basket
- The Search function
- The Log-in for members
- The language selection for multilingual shops
- Design images

However it should be understood that none of these items have to be placed into the header – the header does not even have to exist, as long as these components are placed in other lay-out areas.

Sidebar

This area usually contains **Index 2**. It also contains some Website HTML areas above and below the Index, which allows inserting additional code via ShopFactory.

This Sidebar area is optional, as long as the index is placed in another location, such as into the footer or header of the website. However you can of course also add all Header elements into the Sidebar to create a template without Header.

If the index is not on the left side, then the Website and Page HTML will open a new column on the left side to display the HTML website areas allocated to this region.

Website Footer

The footer usually contains the **Website Footer** and some design elements.

The website footer is added via the "Edit Website Footer" Function on the ShopFactory Central page and is shown on all Website pages. It can for example be used to add a Copyright Note to all pages.

However it is of course also possible to add other elements to the website footer, such as search, an index or any other website element.

Content

This area contains the actual website content, i.e. your product and content pages. It contains a number of additional areas, which are important to take into account – see The Content Area for more details.

List of Website elements

Below is a list of the Website elements, which are included in a Website Theme. They can be added to any Website Template area, as per your design requirements.

> There are of course many other elements that you can add to a website, but they can only be added to the content areas as part of the page and product designs.

All Website Elements MUST be included in the Website Template, as they can be enabled or disabled via ShopFactory functions. The template is not allowed to control which website elements are displayed. This is done via the ShopFactory Interface.

Website Elements	Template
Website title	Added in ShopFactory Central
Company logo	Added in ShopFactory Central
Website Slogan	Added in ShopFactory Central
Index 1	Pre-Set with the Website Theme. Can be changed via Designer. Uses its own Template and inherits colours from Website Theme.
Index 2	Pre-Set with the Website Theme. Uses its own Template and inherits colours from Website Theme.
Search function	Can be turned on or off via *Settings | Miscellaneous*
Member-Log-in	Requires Membership service to be enabled to be shown
Language Selector for multilingual shops	Is shown automatically when several shop languages are enabled
Website HTML areas 1,2,3,4,5,6	Added in ShopFactory Central. Will only be shown, if HTML code contains visible elements. See Website HTML areas
Content area	This is separate from the Website Theme and uses its own templates for the page style and the product loop. Colours are inherited from the website theme. See The Content Area
Website Footer	Added in ShopFactory Central
Website Design elements	Are defined by the Website Theme template. Design images be changed in Customize Design.
Mini-Cart	Displays the Total price of items in the shopping cart. Is selected via Designer.

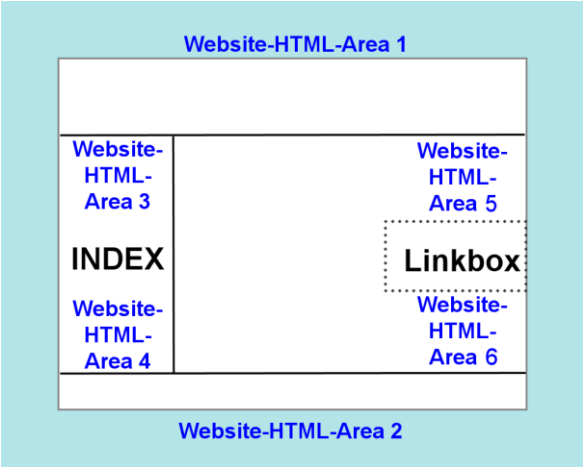
Shop wide Discount Message	
Shop wide Discount Percentage	Optional – could clash with text not included by us

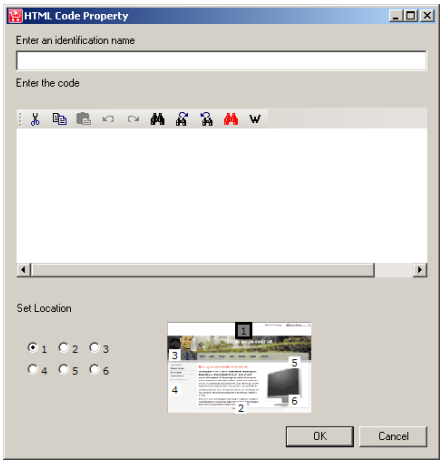
Website HTML areas

In these HTML areas ShopFactory places the content added with the "Edit Website HTML" function on the ShopFactory Central page. The text below indicates the positions of the different HTML areas.

These HTML areas can be used for display purposes such as adding a signup form as well as to place some code, such as Google Analytics or to show additional design elements.

Any code added to these areas will be added to all Website pages, but will only be visible if the HTML code includes visible elements.





The Content Area

The content area is used to display the actual page content of pages. It is split into several regions. These regions will usually only be shown, if the shop builder has added appropriate content to ShopFactory or enabled the appropriate setting.

Different templates are used to create the different areas.

The content area is located within the Content div.

Content area regions (page)

The content area is created with three templates: The page template, the product loop and – if required – sub page index linking to sub pages.

Each page has a right column, which is activated if content is available for it. This column contains the Page HTML codes 3 and 4 as well as the Linkbox. The column can be set up like a proper column going from the top to the bottom of the page, or parts thereof.

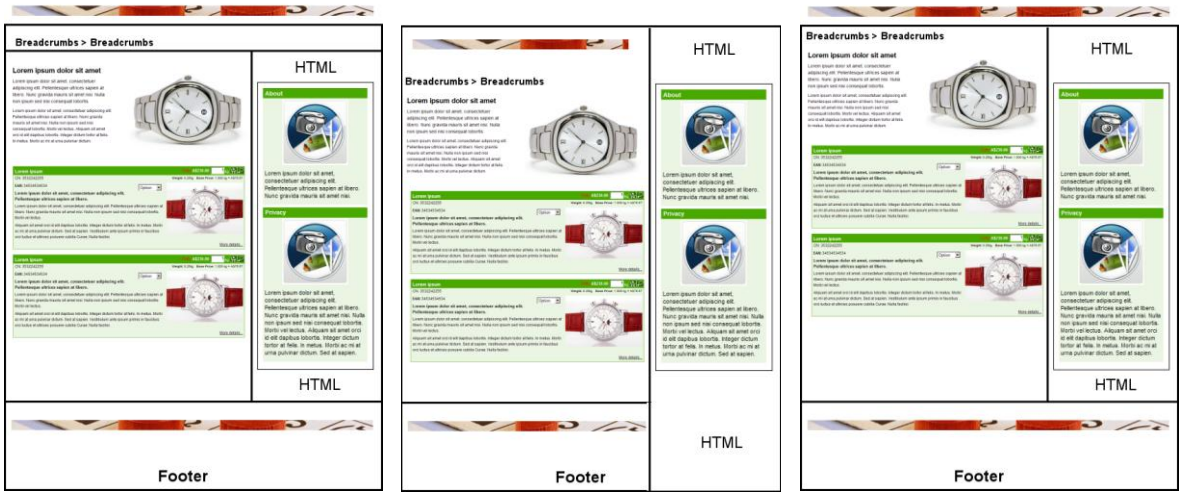
Content area Lay-outs

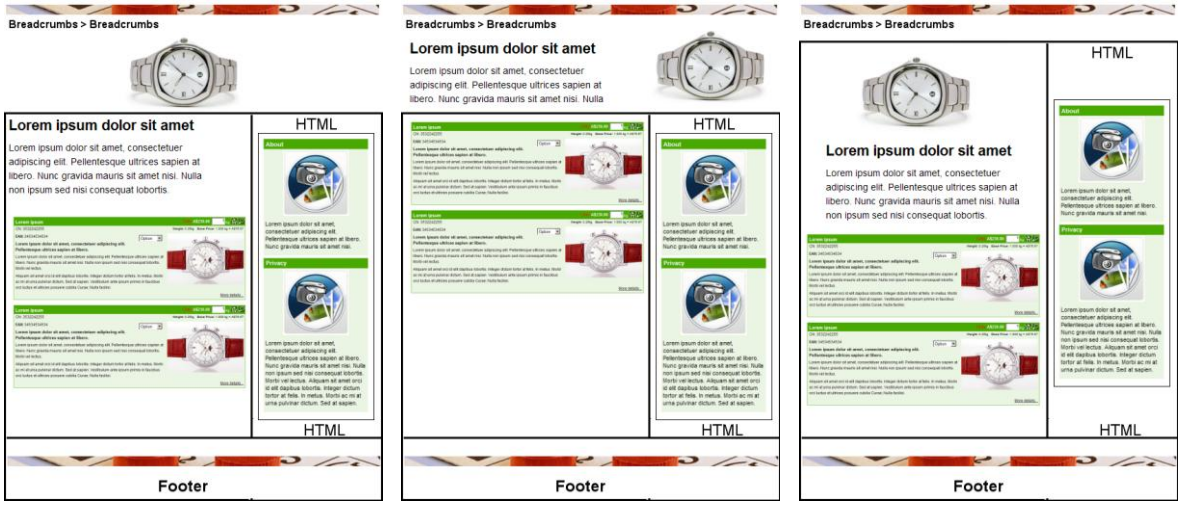
Below are the most likely page scenarios we need. HTML Page codes 1 and 2 would be above the top item and below the footer.

The following lay-outs do not include the Page Description, to make the lay-outs easier to understand. However the description must be placed below the Introduction. Where the Introduction is above the Linkbox, the Description must go next to the Linkbox.

Sub-Page Navigation is either placed at the very top of the page (even above HTML code), so it can extend from the Website header if required by the design, or below the Introduction.

Design images added to a theme will of course influence these placements.





HTML areas have to be cut off in width if they display content, so they do not force the side bar to grow too big. It is up to the user to make sure they don't create too big an area.

List of Page (content area) elements

This is a list of the elements on a page. The page also contains the product loop which shows the product details. See Product Loops and Products templates (More details) for a list of product elements.

All page elements MUST be included in a template and can not be removed. You can however change the way in which they are ordered and displayed and add design elements.

Page Elements	Explanation
Details for Search engines	Provided with the *Enter Details for Search Engines* functions in Page Properties. Creates Meta tags for each page.
Page HTML code area 1,2,3,4	Uses the *Add HTML code* function in Page properties.
Banner link	Uses the *Add Banner Link* function in Page Properties. Banner links can be inserted by the user on top or bottom of a page.
Breadcrumbs	Breadcrumbs are path of pages listed on top of a page, such as *Page 1 > Child Page 1* to help with site navigation. They are turned on or off via *Setting* | *Miscellaneous*.
Change Currency	Usually integrated into the Bread crumbs. Allows changing the currency in the shop if multiple currencies are defined.
Shopping cart	This shopping cart is hidden until a product is added to it. Used only if the appropriate style is on. Style is selected in *Designer* | *Select MiniCart Style*.
Page Content	Contains the **Page Title**, **Page Description** and **Page Image** (Image, Caption, Screen tip). The page title can be turned off in Page Properties in ShopFactory.
Product Loop	Contains the product information. See Product Loops
Linkbox	This contains links to other pages and products, as well as page HTML code areas
Subpage navigation	Contains the links to sub-pages. Is only visible if an appropriate navigation style is selected and sub-pages exist. Depending on the selected sub-page navigation style it is inserted either at the top of the page or after the introduction.
Auto-Split Page counter	ShopFactory automatically splits pages if they have more than the defines number of paragraphs or products (See Settings | Miscellaneous). This allows customers to navigate to the next pages.
Footer	Displays the content of the Page footer field.

Product Loops

All product elements can be included in a product loop or on the *More Details* page, the actual product page.

The structure is entirely up to the designer. A product loop can be as little as just the product image linking to the Product page or include all product elements.

If an element is not included in the product loop and the ShopFactory user enters value for this element – for example ***Feature List*** – then the Product Page is generated with this additional field on it. Otherwise the Product Page is not generated.

Product Pages are only created, if customers add content to product elements, which can not be displayed in the product loop.

The Product Page MUST support all product elements.

List of Product elements

Loop elements can be used in the loop and on the product page.

Loop and product page elements

Product Elements Loop	Explanation
Product ID	Allows ShopFactory to identify the product and places a bookmark on the page.
Product Price	As per name
Discount Note	As per name
Price	As per name
Quantity Box	As per name
Quantity Unit	
Currency Symbol	As per name. Is displayed together with the price always as ShopFactory controls the position of the symbol depending on the currency before or after the price.
Discount Price	As per name
Special Discount Message	As per name
Base Price	As per name
Weight	As per name
Catalog Number	As per name
Product Image	As per name
Product Image Screen Tip	As per name
Product Image Caption	As per name
Multimedia Link	As per name
Add to Basket button	There are multiple types available – Link to more details page only or actual Add to Cart button. See Add to Basket Button Options
Product Headline	As per name
Product Description	As per name
Options and Choices	As per name
Cross Promotions	Inserts a link box with links to related products and pages, when entered by user.
Highlights	As per name
Longer Description	As per name
Features	Table with Feature Name and Optional Description.
Slideshow	Now combined with larger image, but can be shown separately
Product Discounts	As per name

Link to Shipping charges	As per name
Tax message	As per name
Stock Message	As per name
Stock Level	As per name
Manufacturer	As per name
Manufacturer Code	As per name
Product Code	As per name
Distributor Code	As per name
Price Code	As per name
EAN-UCC	As per name
Design images	As per name

Product Page only elements

While these elements are part of a product, they are only shown on the actual product page(more details page) not as part of a product loop.

Product page only	Explanation
Mata Name	Allows defining a Meta Name for Search engine optimization.
Meta Description	Allows defining a Meta Description for Search engine optimization.
Meta Keywords	Allows defining Meta Keywords for Search engine optimization.

Linkbox

The link-box is displayed on the right side of the page – but only if content has been added to it with the "Link to other pages or products" via the ShopFactory.

Depending on the width of the content area, the width of the Linkbox and the image size in the Linkbox is automatically adjusted.

There are three sizes preset:

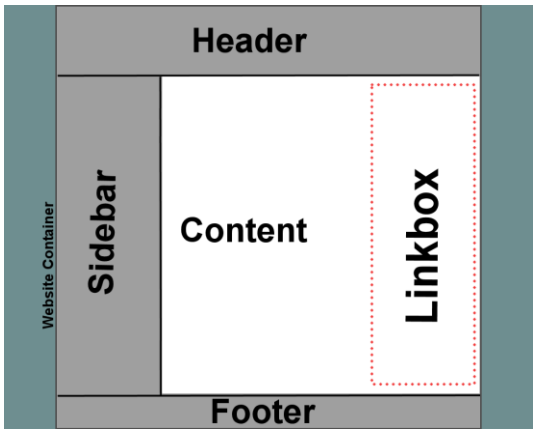
Small (S) = 160 px wide

Medium (M) = 196 px wide

Large (L) = 196 px wide

If you change from a **L**arge website theme to a **S**mall website theme, ShopFactory will automatically resize the width of the Link boxand the link image based on the presets of the page template. The website theme is set as part of the Website template.

The Linkbox is also shown if you add HTML code to the associated HTML areas; see Page HTML areas and Website HTML areas



Page Footer

Each Page also has a footer, which allows you load information to the bottom of the page, below the Product Loop. This is separate from the Website Footer

The templates used by ShopFactory

The different lay-out areas of ShopFactory are generated from different templates.

When you customize a website, you can decide how many of these templates you want to adjust.

- Index (Horizontal / Vertical)
- Pages
- Product Loops
- Products
- Special Pages
- Website
- Object Fragments

Where are the templates

If you have installed ShopFactory without changing the installation location, you will find the templates in the different folders located at

C:\Program Files\ShopFactory V9\ShopFactory\Templates

Here you will find a number of different folders. Each Folder contains template variations for the appropriate type of template.

The folder .templates does not contain any templates and should be ignored.

Element	Template
Index (Horizontal / Vertical)	Contains horizontal and vertical index templates in appropriately named folders. ShopFactory only allows the user to select the correct type of Index based on the folder – so it is important to put the right type of index into the right folder. Each Index has two style sheets. The style sheet used depends on which index the design is applied to – i.e. either Index 1 or Index 2
Pages	Contains a folder for each different page style available
Product Loops	Contains a folder for each different product loop available
Products	Contains a folder for each different product page style (more details) available
Special Pages	Contains a folder for each special page type. Each of these folders contains a number of folders with different styles for each of these special pages.
Website	Contains a folder for each different website theme.
Object Fragments	Contains a folder for each type of object fragment. Each folder contains at least one template style for the object.
~~.Template~~	**NOT to be used**

How to create a new template

The recommended way to create a new ShopFactory Website Theme or style is to edit an existing template (**Name must start with the number 8**).

Simply pick a Website theme or page, product or index style with a name starting with the number 8 which is close to what you are trying to achieve.
Then all you have to do is to adjust it, rather than to create a completely new theme.
This will ensure that your theme will properly work in ShopFactory.

Theme generation quick guide

The following steps will explain how to create a new website theme

1. Open a website

Open  a Demo Website or another website you want to work with.

2. Select a website theme

To create a new website theme it is easiest to start from an existing website theme.

You can either select one of the themes provided, or we have also provided a Website Developer Website Theme as base for your new website themes.

Start by selecting a Website or Developer Theme that most closely resembles the design you want to achieve.

3. Select page and product styles

Select a page style for the Home Page and for the **first page** in Index 2.

Click on a page in the tree with your RIGHT mouse button

On the menu coming up click on 'Select a page style …'

Choose a page style from the themes shown.

Repeat to select a product style (Websites with e-commerce functionality), also for the Homepage and for the first page in Index 2.

When you later save your template, these page and product styles will become the default settings for the new Website Theme.

4. Change theme colours

Click on the "Customize Design" tab where a colour panel and a preview of your website will appear.

To change colours, simply click on a colour you want to change within the colour panel and select the colour from the colour picker. You may also set a hexadecimal colour value via the field for more precise colour choices.

The results of your colour changes will be shown in the website preview.

You can also click on any area in the preview window to bring up a small dialog which allows you to set the values specifically for this area.

Text colours are automatically adjusted to make sure text is always readable – however you can override the settings by selecting the area directly.

When you have finished customizing the theme, click on the 'OK' button. You will be taken to the 'Normal' tab to make other adjustments to content.

Advanced users:
The colours are defined by a CSS file called website.css. You may find it in your website project folder in the following location:

My Documents ** Name of software and Version number **Websites ** Name of Website you are working on **\ Templates \ Website ** Name of theme **\styles

Colour Mapping is defined by an XML file called mapping.xml. You may also find it in your website project folder in the following location:

**My Documents ** Name of software and Version number **Websites ** Name of Website you are working on **\ Templates \ Website ** Name of theme

When you click "OK," the files will be overwritten in the runtime directory.

5. Preview website

Click on the "Preview" tab to preview your website, with the new colour theme applied to it. You may also click the "Preview" button above to view changes in your web browser. To preview the website in a different browser, select it from the drop down menu of the Preview button.

6. Change the index style

Click on the "Normal" tab, select navigation styles for Index 1 and Index 2 by right clicking on the Indexes and customize them.

You can change:

The style by switching to different navigation method, such as a navigation tree instead of a fold-out menu.

The look of the navigation themes by changing colours, adjusting the fonts or introducing images.

Switch the index style:

In the "Normal" tab, right click the index you want to adjust: index 1 or index 2.

Click on the "Select a Theme ..." button on the right toolbar.

Select a theme you want to use from the styles available.

Customize an Index Page:

In the "Normal" tab, select the index you want to adjust: Index 1 or Index 2.

Below the Edit Window click on the tab for "Customize Design".

In the Customize Design window coming up you can change the settings of the appropriate index.

You can customize the Top and Sub Level menus and the page settings:

Top-Level Menu: This is the first level of the index, representing all pages branching directly of the index in the Tree Window.

Sub-Level Menu: These are the index levels below the Top-Level. You can customize it in the same way as the top-level menu.

Page: These are the direct page settings of the index page such as background color and can be changed like any other page.

See also:

Customize page and product styles.

Customize a Menu on an Index Page

This is another very powerful function, as it allows you to completely redefine the look of the menu you have selected. You can create your own button simply by adjusting the settings and adding images and colours.

Click on the tab for "Customize design".

Click on the Index you would like to modify.

Open the tree until you find "Button" and "Rollover" and change the settings as required. **Button:** The settings here define the look of the button as it appears on the website. **Rollover:** This is the look of

the button when the mouse is hovering over it. To preview it, you must put your mouse over a button on the index you are changing.

Each button is split into three sections:

Left | Right | Center

The Center Area always contains the text, if this is used.

Left and Right areas allow you to put images or colours before or after each button.

By adjusting the settings for each area, you can redefine the look of a button and how it behaves when a mouse hovers above it.

7 Customize page, paragraph and product styles

You can change the entire look of your website by selecting new page, paragraph and product (websites with e-commerce functionality) styles.

To change the Page style:

In the document tree of "Normal" tab, right click the page you would like to change.

In the following menu, click "Select a page style…" and select a desired page.

To change a Paragraph style:

In "List" tab, select a paragraph or multiple paragraphs

Right click and select a desired paragraph style

To change a Product style (websites with e-commerce functionality):

In "List" tab, select a paragraph or multiple paragraphs

Right click and select a desired paragraph style

To modify these elements:

Click on the "Customize Design" tab.

Click on the item you would like to change.

A menu will appear with names of elements selected. Click on a name and you may apply new colours or add background images to the item.

8 Changing images

You may easily change the images used within a website, by clicking on "Customize Design" tab and by clicking a theme image, a menu will appear with a list of modifiable images will follow.

Advanced users:

Themes using Flash elements do not support progressive or interlaced images.

If you use transparent PNG images as design images, they may be converted to Flash elements to reduce their size. This is not the case for background images in DIVs or other DOM elements.

9 Save website theme

To save your new website theme click on "Save as new Template" in the Designer menu. Give your new website theme a name and save it.

You have now created a new Website Theme which you can use with any website you create.

(If you do not see the new Theme in your selection list, click on the Refresh button.)

One more thing

To give your website theme the finishing touch you can take a screen shot from your website. Reduce the screenshot to a size of around 276x200 pixels and save it in your website theme folder as preview.gif in the website theme folder.

The location of your website theme folder may be found here:

```
C:\Program Files\Name of software\Templates\Website\Name of new theme\
```

The picture will appear as the new preview image for your new website theme when selected.

Adjusting the theme size

Sometimes it is helpful to be able to change the theme itself, to allow your own artwork to fit, for example. See: Adjusting website dimensions with CSS to do this.

How to edit a template

WARNING:

Do not use HTML or style sheet editors which reformat or add their own code.
You will break the templates and ShopFactory will not work with them.

1. Open the Demo Shop and save it with a different name with Save As in the File menu.
2. Select the Website Theme which most closely match the design you want to create.
3. Select the First Page in Index 2 and select the Page style, Product loop, Detailed View and Indexes from the ShopFactory Themes.
4. Go to (C:\Program Files\ShopFactory V9\ShopFactory\Templates).
5. Identify the templates you have selected and copy, paste and rename the templates you want to edit as variations of the selected styles or, if different enough, as new styles (see How to name ShopFactory templates)
6. In ShopFactory select the renamed templates you want to edit:
7. You must click on the Refresh button in the template selection folder to see your renamed templates
8. To edit the website theme, select the renamed website theme

9. To edit the page style select the Page Template for the **First Page in Index 2**
10. To edit the product loop Template select the product theme you want to edit for products on the first Page in Index 2
11. To edit the Products Page (Detailed View Style, More Details Page) select the one you want to change
12. To edit the Indexes select the index styles you want to use
13. To edit the templates for special pages, select the special page templates
14. Edit the renamed templates to match your requirements. Do not forget to adjust the default and maximum image sizes where required.
15. To preview a change, do the following:
16. Reselect the template
17. Select Preview (if you do not see the change, make sure you have edited the correct template and you have selected it.)
18. **NOTE**: You MUST use the preview mode to review your changes, as Normal Mode is set to editing, which shows text for example without wrapping it, making it look wrong.
19. Test the new templates according to the test plan attached to this document.
20. When everything is ok, save the Website theme with a new name with the "Save as New Website Theme" function in the Designer Menu. This will adjust the presets for the Website Theme, and will tell it which templates to use when creating a new shop.
21. When you create a new shop, all the templates you have assigned should now be automatically assigned to the new shop
22. Delete ONLY the website template which you used for editing, as it has now been replaced with the name of the Template saved with the "Save as New Website Theme" function.

How to name ShopFactory templates

ShopFactory allows you to create a Template and then variations thereof.

The Name of the template and the variation are separated by a **Hyphen**, such as **AAA-AA**, **AAA-BB**.

AAA is the name of the template, **AA** and **BB** are variations thereof. This applies to website themes and all other templates.

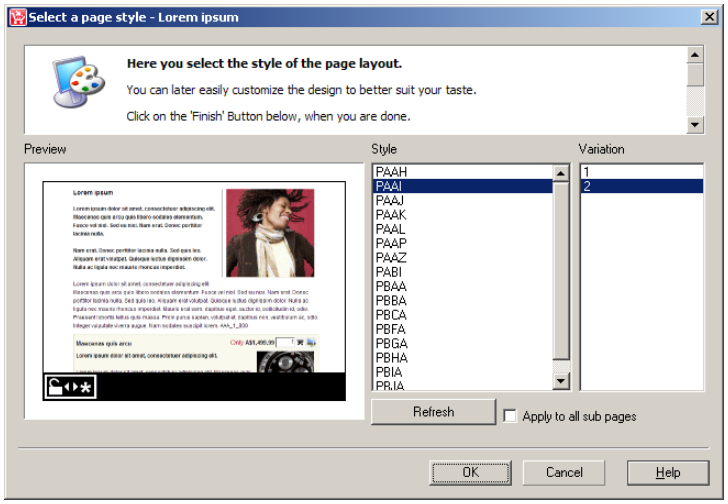
There are additional values which can be attached to the name to further differentiate it based on its size. For example the template

AAA-AA_1024 x –

will fit into an browser window which is **1024** wide or wider and will grow unlimited in height, depending on its content: the **–** stands for unlimited. A template for a window size of 1024 x 768 would be represented as **AAA-AA_1024 x 768.**

We do not recommend using proper words such as colours or moods, as these would need to be translated into other languages, since ShopFactory is multi-lingual.
The preview should provide the user with enough information to make the selection.

ShopFactory uses this **Hyphen** to make the theme selection easier for users by showing the theme and its variations in different selection areas.



The **Style** is in the left window, the **Variation** in the right window.

The icons in the preview image mean the following:

800 will be 800 pixels wide.	width grows with the browser width.
768 height is limited to 768 pixels.	height grows with content.
# height grows with the browser height.	uses Flash elements.

We will no longer support # style themes.

This naming convention is used for all templates – from website template to indexes, product loops, products, pages and object fragments.

How to add a template to ShopFactory

Simply copy the templates you have required into the appropriate folder in ShopFactory Software Templates Folder. To be able to see the template in ShopFactory, you must click on the Refresh button in the appropriate template selection dialog. This will display the template in ShopFactory ready for you to select.

I can't see my new template in ShopFactory

You must click on the Refresh button in the template selection folder before you can see your renamed templates. Once refreshed they will continue to be visible.

How the templates work

Before you can edit templates, you should understand the different files involved. Following are the files and folders included in each template.

WARNING:

Do not use HTML or style sheet editors which reformat or add their own code.
You will break the templates and ShopFactory will not work with them.

What's in a Template folder

Each Template is a parent containing a number of files and other folders as follows. The green background in the tables below denotes the main template files in a folder.

Not all files in Template folders can be edited. Some are used by ShopFactory to function correctly. These files are required, but should not be changed. The have a RED background in the following tables.

Website

Name	What it does
media (folder) (*optional*)	superseded – replaced by defining images in the **Build.ini file**.
parsLang (folder)	Contains the HTML template for the Website Design.
styles (folder)	Contains the CSS files for the website Template and the Indexes
mapping.xml	This defines the colours, colour mappings and fonts assigned to the template. This file is best edited by changing colours and fonts with ShopFactory Customize Design
preview.gif	A small preview image which is displayed in ShopFactory to allow selecting the Template
build.ini	Defines which files have to be copied to the Project folder and which ones have to be converted. You must understand how this works. See **Build.ini files** for more details.

Pages, Special Pages

Name	What it does
preview.gif	A small preview image which is displayed in ShopFactory to allow selecting the Template
page.html	The HTML template for the page design.
stylesheet.css	The CSS file for the html page template
prices.js	Required by ShopFactory – ignore, but **must be kept**
prices.js .CDB	Required by ShopFactory – ignore, but **must be kept**
build.ini	Defines which files have to be copied to the Project folder and which ones have to be converted. You must understand how this works. See **Build.ini files** for more details.

Product loops

Name	What it does
productloop.html	The HTML template for the product loop design.
stylesheet.css	The CSS file for the product loop template
preview_pa.gif	A small preview image for the look of the template as **paragraph** only, displayed in ShopFactory to allow selecting the template
preview_pr.gif	A small preview image for the look of the template as **product**, displayed in ShopFactory to allow selecting the template
build.ini	Defines which files have to be copied to the Project folder and which ones have to be converted. You must understand how this works. See **Build.ini files** for more details.

productloop_html.CDB	Required by ShopFactory – ignore, but **must be kept**

Products (More Details view, detailed view)

Name	What it does
preview.gif	A small preview image which is displayed in ShopFactory to allow selecting the Template
product.html	The HTML template for the product page design.
stylesheet.css	The CSS file for the product page template
alias.ini	Advanced users only - allows creating new conditions
build.ini	Defines which files have to be copied to the Project folder and which ones have to be converted. You must understand how this works. See **Build.ini files** for more details.

Indexes

Indexes are sorted by horizontal or vertical. ShopFactory uses this distinction to allow users to only select appropriate index templates. This way you can not accidentally select a horizontal template for a vertical space.

Name	What it does
preview.gif	A small preview image which is displayed in ShopFactory to allow selecting the Template
build.ini	Defines which files have to be copied to the Project folder and which ones have to be converted. You must understand how this works. See **Build.ini files** for more details.
styles	Contains the CSS files for the index.
sublevels.html	The template to display the indexes for child pages
toplevel.html	The main index template

Object fragments

Name	What it does
objectfragment.html	The HTML template for the design.
preview.gif	A small preview image which is displayed in ShopFactory to allow selecting the Template. Not all fragment types have a preview image, as currently not all can be selected via ShopFactory
build.ini	Defines which files have to be copied to the Project folder and which ones have to be converted. You must understand how this works. See **Build.ini files** for more details.

What does a template do

Templates have a number of tasks in ShopFactory.

First of all they define the look and feel of a Website. However they also provide ShopFactory with information which lets ShopFactory know how to deal with a template.

This includes defining image sizes, so users can not add images to a template design which would break it.

Other presets tell ShopFactory which templates to combine when creating a new shop.

Templates also pull in the content data from ShopFactory entered by the user to create the final pages. To do this the templates use SF-SmartTags and Macros.

Creating the look and feel

To create the look and feel of a website, ShopFactory combines a number of templates with the content provided by the user, as described above.

To do this it uses the HTML files and CSS Style sheets of the different templates in a certain order, so the different templates have the ability to adjust default settings provided by ShopFactory or by templates earlier in the order.

The order is as follows:

ShopFactory Settings → Website → Page → Product loop

How does this work? If a value is set in the Website CSS file, or a design image added, then the same value or file later in the order will override that value.

Example:

ShopFactory automatically adds the file **add2basket.png** as add to basket button to shops. If you create a new website, you can add your own **add2basket.png** file to the media folder of the website template.

Your image will then override the image added by ShopFactory and will be use for all products.

If you were to add the same image to a specific product loop, it would override even the image added to the website template. However if you give it the same file name, the image added to the product loop would also change the product image in all other product designs selected on other pages.

ShopFactory Template Presets

Presets tell ShopFactory image sizes, template combinations and other details. They also influence the way ShopFactory behaves when a user interacts with a template via selection dialogs or the Customize Design function.

Presets can be found in the various templates. Some examples are listed below.

By changing the presets you will affect how a shop will behave, when it is either created with or converted to this website theme or page or product style.

Website presets

Example – Defining the company image size

The following preset tells ShopFactory that this Website theme supports a company logo with the maximum size of **784x80** pixels, but that the recommended size should be **196x80** pixels.

ShopFactory uses this information to automatically resize any company image selected by the user to the recommended image size.

However if users choose to override the recommended image size, they will not be able to go beyond the maximum size permitted by the designer. This prevents that ShopFactory users can break the template.

It requires however that the designer selects accurate presets.

```
<sf:macro object="CompanyImage" recwidth="196" recheight="80" maxwidth="784" maxheight="80" />
```

Example pre-setting an object fragment template

The following preset ensures that the template uses the code fragment **OFA1** to create the log-in function displayed on the page.

By creating a new log-in fragment and pointing the website template to it, the Designer can change the look of the log-in and pre-set it, so this fragment will always be chosen for this template.

Of course the fragment must be available on the computer of the user.

```
<!-- This element is available within the 'Object fragment' folder-->
<!-- BEGIN: Login -->
<sf:macro object="Start_DivLogin" />
<sf:macro object="LoadLogin" design="OFA1" />
<sf:macro object="End_DivLogin" />
<!-- END: Login -->
```

Example pre-setting the index style to be used and its values

The following preset calls in the index design VSE_4. It defines the index orientation as Horizontal, so if ShopFactory users want to change the Index style, they will only be able to select horizontal styles.

showhomelink=”true” means the Index will include a link to the home page.

showlinkimagesublevels="false" means the Link image of the page or product will not be shown in the Sub-Page navigation levels. This can be changed by the user in ShopFactory.

showlinkimagetoplevel="false" means the Link image of the page or product will not be shown in the main navigation level. This can be changed by the user in ShopFactory.

```
<!-- Start Index1 -->
<sf:macro object="Start_DivIndex1" class="GC2" />
<sf:macro object="LoadIndex1" design="VSE_4" orientation="Horizontal" scroll="976" showhomelink="true"
showlinkimagesublevels="false" showlinkimagetoplevel="false" />
<sf:macro object="End_DivIndex1" />
<!-- End Index1 -->
```

Example presetting the page styles to be used

This following preset tells ShopFactory which templates to use for the content area of the website theme selected.

Based on this preset the Shop about to be created will use the page template **PAAI_2**, the product loop template **PRDV_2**, the page template **PAAi_2** for the home or welcome page, the product Style **PRDV_2** on the home page and the product page which displays the complete product details **PDDV_1.**

While in this example the

```
<!-- Content -->
<sf:macro object="Start_DivContent" class="GC22 ContentBody EqualHeight" />
<sf:macro object="LoadContent" detailedproductdesign="PDDV_1" pagedesign="PAAI_2"
productdesign="PRDV_2" welcomepagedesign="PAAI_2" welcomeproductdesign="PRDV_2" />
<sf:macro object="End_DivContent" />
<!-- End Content -->
```

Page Pre-sets

The following presets tell ShopFactory how to deal with images on a page.

ShopFactory has three general widths for the Content Area, Large (L), Medium (M) and Small (S).

As a generic page style may be called into templates of varying width, ShopFactory must know what to do, depending on the available space.

ShopFactory also has to deal with users switching from a wide website style to a narrow website style.

The following pre-sets help with this by defining the image sizes for the different content width areas used in the templates.

You may want to change for example the maximum width settings for images (maxwidth), to suit your new design. However usually you should not have to touch these values.

When changing these values you have to allow for the width of the Linkbox, the width of the text and the width of the image, unless your design does not have these elements next to each other.

```
<!-- BEGIN: Page parameters -->
<!-- begin: do not use tabs to indent the attributes -->
<sf:macro object="SetBannerLinkImageSizes"
  maxwidth_L="950" maxheight_L="300"
```

```
 maxwidth_M="750" maxheight_M="300"
 maxwidth_S="560" maxheight_S="300"
/>
<sf:macro object="SetPageLinkBoxImageSizes"
 recwidth_L="184" recheight_L="184"
 recwidth_M="184" recheight_M="184"
 recwidth_S="148" recheight_S="148"
 maxwidth_L="184" maxheight_L="400"
 maxwidth_M="184" maxheight_M="400"
 maxwidth_S="148" maxheight_S="400"
/>
<sf:macro object="SetPageImageSizes"
 recwidth_L="350" recheight_L="500"
 recwidth_M="300" recheight_M="400"
 recwidth_S="200" recheight_S="300"
 maxwidth_L="540" maxheight_L="1000"
 maxwidth_M="400" maxheight_M="1000"
 maxwidth_S="250" maxheight_S="1000"
/>
<sf:macro object="SetSizes" name="SideBar_R"
 width_L="196"
 width_M="196"
 width_S="160"
/>
<!-- end: do not use tabs to indent the attributes -->
<!-- END: Page parameters -->
```

There are also presets for the width of the Linkbox and for Banner images.

Product presets

Product pre-sets are also mainly concerned with image sizes.

You may want to change for example the maximum width settings for images (maxwidth), to suit your new design.

```
<!-- begin: do not use tabs to indent the attributes -->
<sf:macro object="SetProductCrossPromotionImageSizes"
 recwidth_L="165" recheight_L="165"
 recwidth_M="110" recheight_M="110"
 recwidth_S="55"  recheight_S="55"
/>
<sf:macro object="SetProductImageSizes"
 recwidth_L="180" recheight_L="180"
 recwidth_M="130" recheight_M="130"
 recwidth_S="80"  recheight_S="80"
 maxwidth_L="600" maxheight_L="600"
 maxwidth_M="400" maxheight_M="400"
 maxwidth_S="200" maxheight_S="200"
/>
<!-- end: do not use tabs to indent the attributes -->
```

Adjusting website dimensions with CSS

In most cases you will be able to create a new theme by changing the design images and adjusting the size of the areas assigned to header, footer, sidebar and content to cater for the new image sizes. It is not very difficult to change them.

This is one of the tasks you can perform in the website.css template file.

C:\Program Files*Name of software*\Templates\Website*NewTheme*\styles\website.css

WARNING:

Do not use HTML or style sheet editors which reformat or add their own code.
You will break the templates and ShopFactory will not work with them.

Look for the following code in the website.css file:

Blue values relate to adjusting width. **Green** values must result in the blue website width when added together. **Red** values relate to height

Alterations will modify the entire website.

```
/* ** BEGIN: Website width and height ** */
	/* Normally set to 770px (Fits on 800 x 600 screens) or 980px (Fits on 1024 x 768 screens). Larger or smaller sizes are possible, with sacrifice to legability of website text and loading times. */
	#WebSite, #WebSiteHeader, #WebSiteContent, #WebSiteFooter, #WebSiteContent, #DesignImage1, #AppLogo, .WebSiteFootnote {width:980px;}
/* Sum of the following elements width must equal WebSite width. Check other values further down this file for other elements which my be affected by these settings. */
	#SideBar_L{width:196px;} #Content {width:784px;}
	/* Minimum website height */
	#WebSite, #SideBar_L, #WebSiteContent, #Content {min-height:415px;}
/* ** END: Website width and height ** */
```

Once modified, save the website.css file.

In a resized website theme design images or flash elements will likely no longer fit and have to be relaced. You can quickly do this in ShopFactory's Customize Designe mode. Flash elements will have also have to replaced, as they are usually designed for specific area sizes.

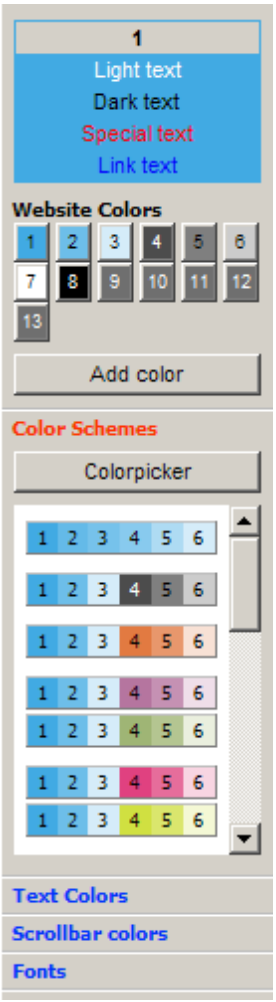
Website Colours

ShopFactory uses a sophisticated colour management system which allows adjusting the colours via customize design with point and click ease.

The colour system is controlled via CSS style sheets, based on Global Colours, CSS classes and a colour mapping file, which you should **never** have to touch.

Colours are assigned globally – that is on a website wide level.

How to change Website colours



The ShopFactory designer makes changing colours easy. You can easily change colours, switch colour schemes or remap colours to create a new look.

To do this simply select the template you want to change and switch to Customize Design Mode.

On the left side of the Designer you find the colour tools which make changing colours easy.

Initial colour settings

ShopFactory Templates usually come with 8 colours predefined.

Colour 1 is the main website colour. Changing this colour will also change all the schemes provided to match this colour. It is the colour around which all other colours should be arranged.

The colours 1-3 are the Website colours. They are ordered from dark to light, that is colour 1 is the darkest colour, colour 2 is a medium colour and colour 3 is the lightest colour. Depending on the design the three colours could have the same level of brightness, though.

The colours 4-6 are the content area colours. They are also ordered from dark to light, that is colour 4 is the darkest colour, colour 5 is a medium colour and colour 6 is the lightest colour.

Colour 7 is always white and 8 is always black.

Additional highlight colours can be assigned to the colours 9-13.

Especially when creating new templates these settings should be maintained, as this ensures that the Colour schemes supplied with ShopFactory can be always be applied to a template without breaking it.

Changing colours

To change a colour simply click on the Colour button under Website colours and select a new colour. To get a precise colour enter the appropriate HTML colour value.

All areas which have this colour assigned to them will now automatically change to the new colour.

You can also change all colours at once by selecting a different colour scheme from the available selections.

Alternatively you can scroll to the bottom of the schemes and open a previously saved colour scheme.

Note that colours may sometimes be covered by images, so not all areas will always change as expected.

Creating your own colour scheme

To create your own colour scheme simply change the colours 1-6. Keep in mind that always go lighter from left to right (See Initial colour settings).

Add more colours

If you need more colours, add them to colours 9-13 as highlight colours. To add a colour click on the Colour button under Website colours and chose the colour from the colour selector or enter a HTML colour value.

New colours should be highlight colours only, as the main website and content area colours must be assigned to colours 1-6, and the numbers 7 and 8 are taken up by white and black.

Remapping colours

When creating a template you may sometimes want to change where a colour is placed. To do this simply click on the area in the template which contains the colour. A menu will open up. Select the applicable Edit colour option in the menu, and pick a new colour from the Colour menu coming up.

This does not actually change the colour itself – it only assigns a different colour to this area, it remaps it. If you want a completely new colour to be assigned to this area, you must first add more colours.

Text colours

To make sure that text is always readable, most text items are set to Autotext colour. You can change this for specific text by clicking on the text and assigning a colour to it.

If Autotext is enabled for text, then ShopFactory will switch between the dark and light text colours, depending on the background colour of the text area.

To change the dark and light text colours, click on the Text colour button below the Website colours and colour schemes.

The colour mapping file mapping.xml

You should NEVER touch this file unless you want to do some very sophisticated remapping which you can not achieve via the CSS style sheets or Customize Design view. This is extremely unlikely, as you can easily remap colours with the ShopFactory Designer.

This file defines the initial global colours and fonts used in a shop. It is part of the website theme template folder.

See also Global colour mapping.

Colours in the website.css file

At the top of the website.css file the global colours for the website are defined. These are based on the settings in the mapping.xml file.

The colours are then used in the different classes assigned to the website elements used.

The best way to change colour settings is via the ShopFactory Designer. Make the changes as required, then save the adjusted website template as a new website theme.

You should NEVER touch these settings unless you want to make some very sophisticated changes which you can not achieve via Customize Design view. This is extremely unlikely, as you can easily remap and change colours with the ShopFactory Designer.

See also Global colour mapping.

Design Images

Design images are images embedded on the website and on pages to define the look of the website. You can easily replace them in Designer.

ShopFactory uses the size of the div which contains the design image to make sure any new image selected is either smaller or the same size. Design images therefore have a fixed image size – when selected with ShopFactory, they will have to cropped or resized by the user to fit the appropriate area.

Design images are assigned as background image of their container.

Transparent images

In some cases ShopFactory converts **png** design images to flash elements.

This helps overcome the problem that transparent png images have a large file size, which would negatively affect the speed of the website.

What are design elements

Design elements defines a given areas which may contain artwork color borders or text-decoration properties, which can be controlled from inside the software.

Each design element must always have the following tag in front of it, to allow the software to interact with it:

sf:object="LayoutObject" class="LayoutObject"

Function	**ID**
Images	DesignImage
Colors	DesignColor
Borders	DesignBorder
Text decoration	DesignText
Padding	DesignPadding
Spacing	DesignSpacing

Each design element must have an ID tag. It should represent the actual function of the tag

However these names are suggestion – the ID should most closely resemble the function of the design element, as the name will be shown to the user in the software for customization. The first letter of every word should be capitalized, and the name should have no spaces in it. Note that you always must write the name in the same way as your code will otherwise not work.

If you have multiple design objects of the same type, you can add numbers to them, such as DesignImage01, DesignImage02 and so on.

Of course if you want to use the same object such as a DesignColor in a number of locations, you must always use the same name.

Layout objects can be included inside XHTML tags such as <table>, <td>, , .

Please note that the image settings such as name of image and image properties for the DesignImage are set in website.css. The same of course would apply to all other design elements.

DesignImage01 as background image for the DIV	
DesignImage01 as background image for the SPAN	
DesignImage01 as background image for the TABLE	<table cellpadding="0" cellspacing="0" border="0" sf:object="LayoutObject" class="LayoutObject" id="DesignImage01"><tr><td></td></tr></table>
DesignImage01 as background image for the CELL	<td sf:object="LayoutObject" class="LayoutObject" id="DesignImage01"></td>

Website Templates

The website template defines the look of the website using different website lay-outs combined with design images and colours. See How SHOPFACTORY Website themes and templates work.

Different home page design

Use the ***if welcome page*** function to create different designs on the front or home page compared to the rest of the website.

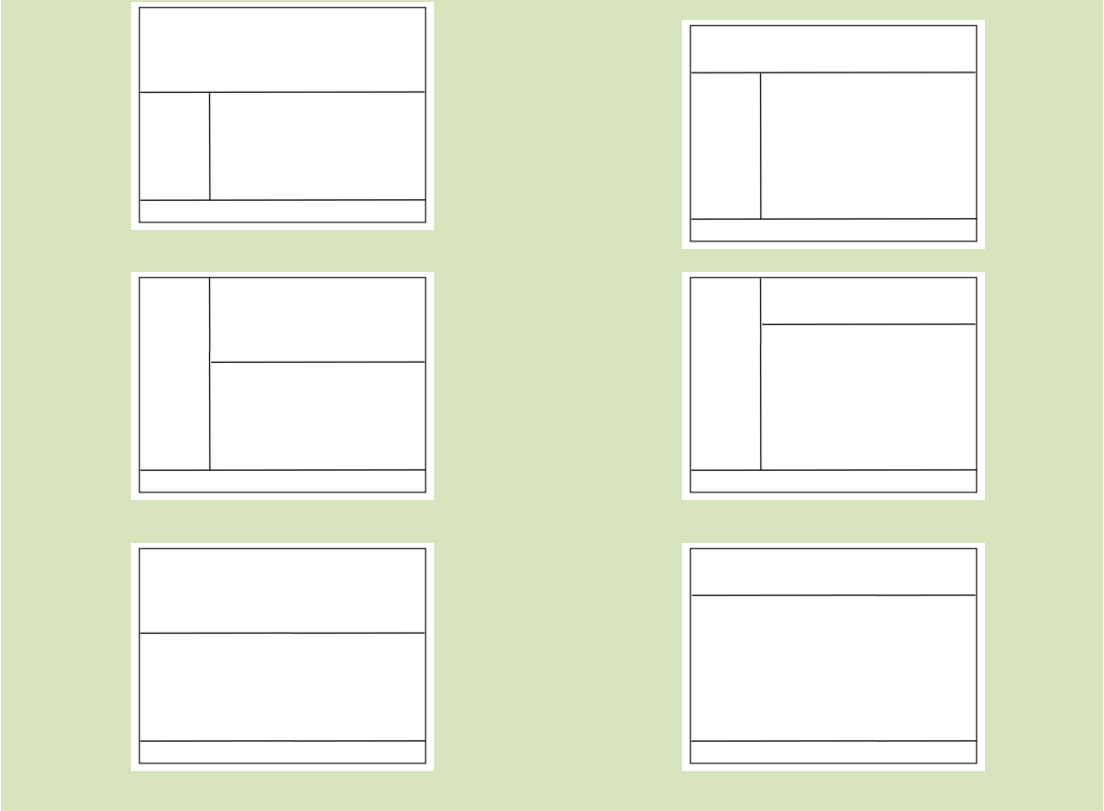
In most cases this means that the header area on these templates will be larger und uses a different design image when compared to the other websites.

Elements in the Header such as title, company logo, log-in, languages and so on may also have to be repositioned using the function.

In these cases the Slogan can be included only on the first page and not on the other pages, where the header field is smaller.

Some examples where this could be used are as follows.

Homepage	Regular page



You should be able to easily adjust these sizes via the CSS files and the template. However you must use the *'if welcome page'* function to resize and reposition the items.

Content area width

The width of the content area depends on the design of the template. Depending on the width ShopFactory automatically sets image sizes as well as the width of the Linkbox when switching between templates.

To do this only the display HTML values are changed – not the actual image files.

The width ID of S, M or L is defined in the Website Theme Template with the SF tag:
<sf:macro object="**SetContentStyleSize**" size="**L**" />

Setting Content area width

These are the width assigned to the different templates. When creating a template you must ensure to associate the correct Content Width Indicator into the **SetContentStyleSize** value. Your template must of course also assign this much space to the content area.

Width	Minimum	Maximum
S	550	649
M	650	979
L	980	Unlimited

How the content area width is calculated

Most page styles require you to place multiple elements next to each other – or at least to allow for them to be placed next to each other.

When looking at a typical page template, you may have next to each other the **page introduction**, the **page image** and the **link box**.

To ensure that the page style will not break when customers add content, the template must define maximum width values for these elements so they all fit into the available area.

Recommended width calculation

Here are the typical recommended widths we have preset in ShopFactory in the page templates for the different Content area width sizes.

These following values relate to the width of the Content div in the website template.

Recommended Size	Text	Picture	Linkbox	Total
L	250	350	200	**800**
M	160	300	170	**630**
S	160	200	130	**490**

Maximum width calculation

Here are the widths we have preset in ShopFactory in the page templates for the different Content area width sizes. Make sure when creating a new theme to assign the appropriate Size in the Website Theme to pages and product styles will fit correctly.

Maximum Size	Text	Picture	Linkbox	Total
L	250	540	200	**980**
M	160	400	170	**730**
S	160	260	130	**550**

Width calculations for different page lay-outs

Different page lay-outs which for example do not have all three elements next to each other can be created – however this does not affect the content width available in the website theme.

It only allows you to assign different recommended or maximum sizes to the page template.

Obviously an image can have larger values assigned to it for example, if the Linkbox is below the image rather than next to it.

Index 1 and Index 2 Navigation

Index 1 and Index 2 can have two different design templates assigned to them. They can also both be Horizontal or vertical, depending on the template design. The following rules must be obeyed when creating index styles.

Index layout

Every index consists of an index container with a beginning and an end item, as well as of the actual page links.

Each page link is separated into three areas:

Left | Centre | Right >

The Page title and the link image are placed in the centre, if supported. The left and right areas are for design elements such as colours or images.

Every area has a mouse over status, which allows showing different colours or images on mouse over.

The images and colours can easily be adjusted in ShopFactory with the Customize Design Function.

Main Indexes

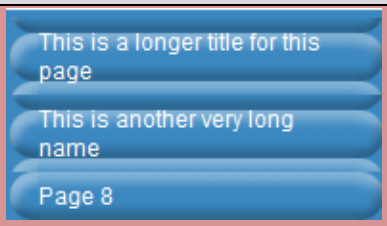
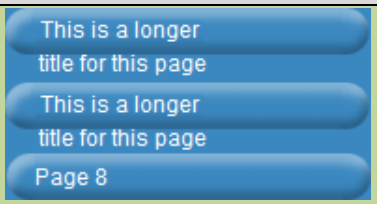
We never know how much text a user enters into a title field. Index styles must deal with names across multiple lines without falling apart.

Horizontal Indexes

Make sure the text area can expand in width to accommodate longer text. If the index is too long, a sideways scrolling function will appear to prevent the Website design from breaking. A user who doesn't like that can then change the names of the pages or reduce the number of links.

Vertical Indexes

Make sure the artwork allows for multiple lines of text. Where this is not possible, align the artwork and the text in such a way, that the design breaks clearly so that the user can see that his text is too long for the index. Make sure you test your design with long page titles.

Wrong	Right
 Here it looks as if the design is broken – it is not clear, why.	 Here you can easily see that you have entered too much text and it does not fit into the button

Editing in ShopFactory

A user must have access to all design functions to change the elements of the index, including the colours, images and borders.

Borders

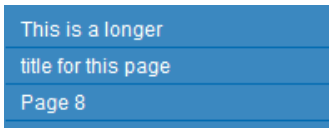
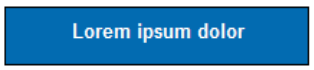
Borders must be added where they logically fit or where a customer with a different taste to yours could expect one to be (without totally destroying the design).

A button would not have a border. A text item might have one in the form of an underline or a separation line. An image link could have one around the whole index link object. The border must be editable via ShopFactory.

ShopFactory users do not have to be allowed to add extra border elements – they can only enable or disable the border elements provided.

Examples:

Below borders are used as separators and to highlight for the active page

	 Note green line under Home link	Here the border is around the whole item. 

Link image

The max and recommended size of the link image must be defined.

Page Templates

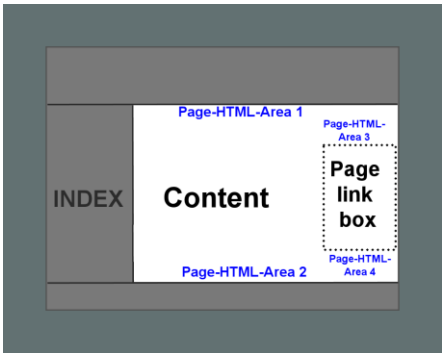
The Page template is called in by the Website theme to define the look of the content area. You can switch between different looks to change the look of the website.

Page HTML areas

You can add HTML code into multiple positions on each page. This HTML code will only be shown on the specific page you add it to.

HTML code is added to a page with the **Add HTML code** function in the ShopFactory Page properties dialog.

To add HTML code to ALL pages at the same time, review Website HTML areas



Product Loop Templates

The product loop contains all the products assigned to a page or product category (another name for page). The product loop does not have to show all product details – it just has to provide enough information to make the customer follow a link to the actual Product page, the page which lists “More Details” about the products.

In ShopFactory to get to the More Details Page you select a product and then click on the More Details button.

This means a product loop could contain as little as a product title or a product image, which are linked to the More Details Page or as much as all the product details.

The More Details Page is only generated, if the product loop does not display some product elements. This could be a “List of Features”, a “Longer Description” but also the actual Buy Now Button to add product to the shopping cart.

The default product loop templates which we have assigned to many website themes contain the most used elements, to prevent the creation of More Details Pages – mainly to reduce the time it takes to publish a shop.

However this is not a requirement.

Products templates (More details page)

If a product element is not included in the product loop, then ShopFactory automatically creates a “More details page”, which then shows all product elements. This is based on the Products template selected.

Object fragments

Object fragments are bits of code ShopFactory calls in from the Object fragments folder. By creating a new code fragment and linking a template to it, the new design will be called into the template.

An example for this is the Mini-Cart – shopping cart embedded on all pages. While the mini cart can be selected via the Designer menu, most object fragments can not be selected by the user and are linked directly to the template.

Existing Object fragments

Login

Defines the log-in field design.

Search

Defines the search field design.

Minicart

Defines the look of the Mini-Cart embedded on all pages. Can be selected via the Designer menu.

Switch Currency

Defines the switching the currency function

Switch Language (SwitchLang)

Defines the switch language function

Special Page Templates

Special Page templates work just like normal templates.

In fact in ShopFactory they are often simply a copy of the same page style, placed into the Special Page template folder for each special page type.

However you can customize the design of each of these pages to better cater for the special function the page performs.

His is just done like editing any other page, with the only difference being that the page is located in the appropriate special page folder.

Flash Design Elements in templates

ShopFactory supports the use of Flash elements in Templates.

However where Flash elements contain images and colours, it must be possible to change them via Customize Design. To do this, images and colours must be defined via a small XML file, which can be edited by ShopFactory.

This chapter show how this file is created.

To look for example files in existing templates, simply pick a website template with flash elements and look at the template.

The XML file must have the same name as the flash element. The file is imported by the Flash element via vars.

Example importing XML file into Flash element

In this example the Design image is actually the flash file **AGH-980-header-mirror.swf**. It imports its values for colours and images to be used by calling in the **AGH-980-header-mirror.xml** file.

```
<sfm_DesignSticker name="DesignImage1" src="../media/AGH-980-header-mirror.swf" width="980" height="265" flashvars="xmlfile=../media/AGH-980-header-mirror.xml">
```

The name of the Flash file and the XML file must match.

XML file for Flash elements

XML value	Explanation
Picture Identifier (PICTURE_1)	The Picture Identifier MUST have the name PICTURE_1, whereby X represents the number of the picture. To add multiple pictures to a flash element increase the number by 1 for every additional picture: PICTURE_1, PICTURE _2 and so on.
Src (../media/name_of_picture.jpg)	This defines where the picture is located. As all template images must be in the Media folder, all you have to do is replace the file name with the correct file name. Do **NOT** remove the path **../media/** !
fixedheight / fixedwidth	This defines the size of the picture used. If a ShopFactory user wants to replace the predefined images ShopFactory uses these values to automatically resize the new image to fit the flash element
transparency	Set the transparency of the image to achieve extra design effects.

Colour properties

XML value	Explanation

Colour Identifier	The Colour Identifier MUST have the name COLOR_1, whereby X represents the number of the colour. To add multiple colours to a flash element increase the number by 1 for every additional colour: COLOR_1, COLOR_2 and so on.
Clr	Enter the colour value in the correct HTML format, such as *c50000* or *3874a8*
map	This maps the colour to the appropriate GC colour in ShopFactory. If the GC colour is changed in Customize Design, the colour will also change in the Flash element to make sure it matches the colour selections by the user. See also Mapping colours

Sample XML File for using Flash files in ShopFactory

This file tells the flash file to import various images and to set a number colours and transparencies.

```
<XML>
	<ROOT>
		<PICTURE_1 src="../media/designimage1.jpg" fixedheight="130" fixedwidth="980"
transparency="0"/>
		<PICTURE_2 src="../media/designimage3.swf" fixedheight="260" fixedwidth="350"
transparency="0"/>
		<COLOR_1 clr="3B88C0" map="C2"/>
		<COLOR_2 clr="026BB1" map="C1"/>
		<COLOR_3 clr="FFFFFF" map="C7"/>
	</ROOT>
</XML>
```

Build.ini files

This file gives ShopFactory instructions on what to do with the files in a template folder.

It defines if ShopFactory copies files from this template folder to a new project, or if a file should be converted before being placed into the project folder.

Example Page Build.ini file

In the following Build.ini file ShopFactory is given the instructions to **convert** the website.html file before placing it into the new folder.

Converting means it will add content added by the user to the page or follow other instructions contained within the template.

ShopFactory is also told to simply copy the files **stylesheet.css** and **add_to_cart.png** to the new project, and to specifically place the **add_to_cart.png** image into the **media** folder of the project (otherwise it wouldn't be found ☺ by the shop.)

```
[Page]
Convert=%websitetemplate%\parseLang\website.html,%contents%\%lang%\%pageloc%
Copy=stylesheet.css,%styles%\pd_%stylename%.css
Copy=add_to_cart.png,%media%\add_to_cart.png

; Version tag, please don't remove
; $Revision: 2574 $ $HeadURL: svn://3d3-p432/ShopFactory/trunk/bin/Templates/Products/PDAK_1/build.ini
$
```

You could also add another template to the page folder which uses the ShopFactory SF-tags, and have ShopFactory convert it by adding it to the build.ini file.

This could be used for example to create an RSS Template for an RSS feed for the page.

Alias.ini files

Alias.ini files are advanced files which give ShopFactory instructions and allow various settings of ShopFactory to be overridden on a template level. They should only be edited with a full understanding of how they work.

Switching Website Themes

Automatic image resizing when switching website themes

When you switch a website theme from large to small, images and the link box width used on the page should automatically be adjusted by ShopFactory, to make sure that everything on the page still fits into the smaller content area provided by the smaller template.

This is based on the presets on the page and product loop templates for the different available content areas – see Page Pre-sets

To do this ShopFactory does not actually physically resize images – only the HTML values to display the images are changed.

The SF Smart-tags

In most cases you can create a new template design be editing an existing template and by simply moving elements on the pages around as well as by adjusting the style sheets.

This is made easier by the use of SF Smart tags, XML based tags and attributes which make up the template language used by the software to convert a website project into published website.

These tags and attributes allow the software to determine where HTML elements are positioned on the page and any style or decoration those and surrounding elements might have.

These tags also define the behaviour of ShopFactory as explained in ShopFactory Template Presets.

One of the abilities the tags have is to make sure that HTML code is only added to a page, if a certain condition is met – for example the HTML code to display an image will only be added, is an image is added to ShopFactory.

By moving the Smart tag elements, adjusting the HTML and the style sheets, you can create almost any design you want to.

SF Namespace

XML based tags and attributes make up the template language used by the software to convert a website project into published website. These tags and attributes allow the software to determine where HTML elements are positioned on the page and any style or decoration those and surrounding elements might have.

Through these tags and attributes designers can create their own unique themes for use with the software, providing endless possibilities of customization for both the designers and the software users.

SF Elements

SF elements consist of a collection of functional tags and attributes that can be inserted into an (X)HTML document. Each element has an "object" attribute that holds the object/array path or condition statement depending on what is being referenced.

Below you can see a simple example of the SF conventions within an HTML document.

```
<html>
	<body>
		<sf:if object="SiteTitle">
			<h1><sf:value object="SiteTitle" /></h1>
		</sf:if>
	</body>
</html>
```

sf:object

sf:object is used as an attributes of HTML tags to define areas of importance which can be controlled from inside the software.

Note:
This attribute is sometimes accompanied by an HTML id attribute that must contain a predefined value for the software to work correctly.

Below is an example of sf:object being used to identify the LayoutMaster object in a HTML document.

```
<html>
	<body sf:object="LayoutMaster"></body>
</html>
```

sf:name

sf:name is used similarly to sf:object with the exception that it also sets the id attribute of the HTML element it is added to. This is needed for the software to uniquely identify some elements that are repeated throughout a website such as Paragraphs or Products.

Note: **The id attribute is set automatically by the software for any HTML tag with an sf:name attribute, setting the id attribute on these tags will cause an undesired result.**

An Example of sf:name being used to identify the PageTitle object.

```
<div sf:name="PageTitle">
	<sf:value="PageTitle" />
</div>
```

sf:value

sf:value is used to display an objects text content.

An example of sf:value displaying the PageTitle content.

```
<sf:value="PageTitle" />
```

sf:if

sf:if allows us to include information in the document only if it meets a certain criteria.

sf:if statements include a set of simple matching operands

	Example	Description
>	Object > 0	Is greater than
<	Object < 10	Is less than
=	Object = 5	Equals
&	Object & Object < 10	Separator, match both adjacent conditions
|	Object = 1 | Object = 5	Separator, match either adjacent conditions
!=	Object != 1	Not equal
!	!Object	Non value or doesn't exist

Also note that sub expressions separated by & and | operands can also be enclosed in brackets to enforce in what order the expression is interpreted in. Also in the comparisons above, an empty value is interpreted as being equal to 0.

An example of sf:if being used to check and display the site title

```
<sf:if object="SiteTitle">
	The title of this site is <sf:value object="SiteTitle" />
</sf:if>
```

sf:else

sf:else is used with sf:if statements as an alternative if the criteria isn't met. It essentially reverses the if so for example if the if says "Value = 5" the else is true if "Value != 5" e.g. the direct opposite.

An Example of sf:else with the site title being checked.

```
 <sf:if object="SiteTitle">
	The title of this site is <sf:value object="SiteTitle" />
</sf:if>
<sf:else>
	No site title here.
</sf:else>
```

sf:repeat

sf:repeat is used to loop through and display a series of objects of the same type such as Products or Paragraphs.

An example of looping through the ProductLoop array, and displaying some product details about each product.

```
<sf:repeat object="ProductLoop">
	<div>
		<sf:object object="ProductTitle" />
		<sf:object object="ProductDescription" />
	</div>
</sf:repeat>
```

sf:inject

sf:inject can only be used within sf:repeat. It allows you to output code at certain intervals within the loop.

sf:inject statement's object attribute will accept a number or one of the following interval instructions

Name	Description
AllButLast	All iterations except last
FirstOnPage	The first of each split page
First	Only first iteration
Last	Only last iteration
Even	Every even iteration
Odd	Every odd iteration
A number greater than 0	Every time through the loop if the current interval is evenly divisible by this value, the inject will be true.

An example of sf:inject being used to give every product on an even iteration a different class

```
<sf:repeat object="ProductLoop">
	<sf:inject object="Odd">
	<div class="ProductOddClass">
	</sf:inject>
	<sf:inject object="Even">
	<div class="ProductEvenClass">
	</sf:inject>
		<sf:value object="ProductTitle" />
		<sf:value object="ProductDescription" />
	</div>
</sf:repeat>
```

sf:repeatc

sf:repeatc is used to loop through and display a series of objects of the same type such as Products or Paragraphs, but with a condition.

An example of looping through the ParentLoop array (the page hierachy), and displaying some product details about each page.

```
<sf:repeatc object="ParentLoop" condition="(isvisible[.ID]=1)&(isdeleted[.ID]=0)">
	<div>
		<sf:value object="PageTitle" />
		<sf:value object="PageDescription" />
	</div>
</sf:repeatc>
```

sf:injectc

sf:injectc can only be used within sf:repeatc. It allows you to output code at certain intervals within the loop.

sf:injectc statement's object attribute will accept a number or one of the following interval instructions

Name	Description
AllButFirst	All iterations except first
FirstOnPage	The first of each split page
First	Only first iteration
Even	Every even iteration
Odd	Every odd iteration
A number greater than 0	Every time through the loop if the current interval is evenly divisible by this value, the inject will be true.

An example of sf:injectc being used to give every product on an even iteration a different class

```
<sf:repeatc object="ProductLoop" condition=".Translated=1">
	<sf:injectc object="Odd">
	<div class="ProductOddClass">
	</sf:injectc>
	<sf:injectc object="Even">
	<div class="ProductEvenClass">
	</sf:injectc>
		<sf:value object="ProductTitle" />
		<sf:value object="ProductDescription" />
	</div>
</sf:repeatc>
```

sf:macro

sf:macro refers to predefined snippets of template code that have been created to minimise the developers time and effort when reproducing certain functionality in their website themes.

An example of using sf:macro to call the EnablePage macro.

```
<sf:macro object="EnablePage" />
```

sf_ or sf:value

The <sf_> tag is a special tag that allows placing values inside an HTML tag's attributes.

An example of using <sf_WelcomeURL> being used inside a href attribute.

```
<a href="<sf_WelcomeURL>" />Link to the Home Page</a>
```

The sf:value tag is a synonym for the <sf_> but cannot be used in a tag.

An example of using sf:value:

```
<a href="<sf_WelcomeURL>" /><sf:value object="PageTitle" /></a>
```

Global colour mapping

This is for your information only. You should never have to change the existing colour mapping outside of ShopFactory.

'Global Colour mapping' groups multiple elements into 'GC' CCS classes, for the purpose of setting colours throughout the website. There are 49 GC (GC1 – GC49) classes which are individually allocated with a text colour, background colour and border colour. Each colour is represented with an option of up to 16 C (C1 – C16) colours. The 'C' colours may be edited via the 'Customize design' mode within ShopFactory.

Example:
The colour (#f9f9ee) associated with C8 is mapped to the background of GC12 Product Description, Product Detailed Description and GC23 Page Image Caption.

GC classes may also contain several other settings which include Auto (Text Colours), No Settings (Border colours) or Transparent (Background colours).

By modifying the values of a GC class, multiple elements which are grouped to the modified GC throughout the site, will be modified automatically.

It is recommended for developers to modify colour mapping through 'Customize Design.' Changing colours and mapping generally affects two or more files, which need to stay synchronised at all times.

Manually modified mapping, may lead to unexpected results. It is recommended when changing mapping manually, the developer should be familiar with XHTML, XML and CSS.

Two files for general manual remapping:

```
My Documents/Name and version of software/Name of new theme/Runtime/contents/styles/website.css
```

```
My Documents/Name and version of software/Name of new theme/Templates/Website/Name of theme/mapping.xml
```

The mapping.xml and website.css files also contain other information, such as scroll bar colours, fonts, special text colours and default link colours.

Optional files for manual colour changes:

```
My Documents/Name and version of software/Name of new theme/Templates/Website/Name of theme/media/*.xml
```

Flash XML files do not require the # in front of the hexadecimal colours when modifying the CLR value. If colours are not correctly specified, the affected object will appear to be the colour black.

Default colour mapping list

Default Colour Mapping list below is based off the theme AAA_1:

Class	Element ID
GC1	WebsiteContent
GC2	Index1
GC3	Index2
GC4	ResellerFormTable, Product, ProductWeight, ProductWeightUnit, BreadCrumbs
GC5	ProductTableHeader, ViewbasketHeader
GC6	BreadCrumbs
GC7	ChangeCurrency
GC8	PageLinkBox, pageimagecaption
GC9	ProductHighlight, ProductCrossPromotion, ProductFeatures
GC10	ProductOptions
GC11	ProductIntroduction, ProductDeliveryAdvice
GC12	ProductDescription, ProductDetailedDescription, ListColor1
GC13	AddToBasketDialog
GC14	ViewbasketRow1, ListColor2
GC15	ViewbasketRow2
GC16	ViewbasketExtras
GC17	PageTitle
GC18	PageIntroduction
GC19	PageDescription
GC20	ProductPriceIntro, ProductCurrencySymbol, ProductPrice, ProductBasePrice,ProductPriceOriginal, ProductCurrencySymbol, ProductPrice, ProductBasePrice, ProductPriceOriginal
GC21	ShopDiscountMessage, ProductDiscountMessage
GC22	Content
GC23	PageImageCaption, MoreDetails
GC24	ViewbasketHeader, ProductMoreImagesIcon
GC25	Add to basket dialog button, FavoritesButton
GC26	TextInput in Basket page, Checkboxes (Add to basket dialog)
GC27	NextPreviousLink
GC28	Index1 TD

GC29	Index1 mouseover
GC30	index1 sub levels
GC31	index1 sub levels mouse over
GC32	SideBar_L, Index2 td
GC33	Index2 td mouseover
GC34	index2 sub levels
GC35	index2 sub levels mouse over
GC36	Mini Cart
GC37	ProductTitle
GC38	Website
GC39	WebsiteHeader
GC40	Sitefooter
GC41	Body
GC42	WebsiteSlogan
GC43	SelectBar
GC45	Container1
GC46	Reserved for website theme
GC47	Reserved for website theme
GC48	Reserved for website theme
GC49	Reserved for website theme
GC50	PageLinkBoxContainer1
GC51	SideBar_R
GC52	Product discount special text
GC53	Reserved for page style
GC54	Reserved for page style
GC55	Reserved for page style
GC56	Product calculated price discount special text
GC57	Reserved for product style
GC58	Reserved for product style
GC59	Reserved for product style
GC60	Spare not used
GC61	Spare not used

Website Html Components

Enable page

This macro initializes and displays meta and language information in the appropriate places in the HTML document

Note: This Macro must be placed before the opening HTML tag.

```
<sf:macro object="EnablePage" />
```

Page head title

This macro displays the page relative title.

Note:
This macro must be placed inside the title node within the head node.

```
<sf:macro object="DisplayPageTitle" />
```

Layout master

This object defines the containing layout element for the HTML document. This object is usually placed in the body tag

<body sf:object="LayoutMaster" />

Site title

This object defines the containing layout element for the HTML document. This object is usually placed in the body tag

```
<sf:if object="SiteTitle>
	<sf:value object="SiteTitle" />
</sf:if>
```

sf:if object="SiteTitle"

Evaluates true if the site title exists

sf:value object="SiteTitle"

Displays the SiteTitle content

Company image

```
<sf:if object="CompanyImage">
	<div sf:object="CompanyImage" id="CompanyImage">
```

```
                <a href="<sf_WelcomeUrl>" title="<sf_CompanyImageScreentip>">
                        <sf:macro object="CompanyImage" recwidth="" recheight="" maxwidth=""
maxheight="" />
                </a>
        </div>
</sf:if>
```

Note: The div element can be substituted with any valid HTML element.

```
sf:if object="CompanyImage"
```

Evaluates true if the company image exists

```
sf:object="CompanyImage"
```

This object identifies the enveloping element of the company image

Note: The element must also include the following attribute to work correctly

```
id="CompanyImage"
```

```
<sf_WelcomeUrl>
```

Outputs location of home page

```
<sf_CompanyImageScreentip>
```

Outputs company image screen tip

sf:macro object="CompanyImage"

Creates and displays the company image.

CompanyImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

Site slogan

This object defines the containing layout element for the HTML document. This object is usually placed in the body tag

```
<sf:if object="SiteSlogan">
	<sf:value object="SiteSlogan" />
</sf:if>
```

sf:if object="SiteSlogan"

Evaluates true if the site slogan exists

sf:value object="SiteSlogan"

Displays the SiteSlogan content

Search

```
<sf:if object="SearchEnabled">
	<div sf:object="search" id="search">
		<sf:macro object="LoadSearch" design="" />
	</div>
</sf:if>
```

Note: The div element can be substituted with any valid HTML element.

sf:if object="SearchEnabled"

Evaluates true if the site wide search is enabled

sf:object="search"

This object identifies the enveloping element of the search code

Note: The element must include the following attribute to work correctly

```
id="search"
```

sf:macro object="LoadSearch"

Loads the search code from the ObjectFragment Template

LoadSearch Attributes

Name	Description
design	Name of the ObjectFragment template to use.

Switch language

```
<sf:if object="MultipleLanguages">
	<div sf:object="SwitchLang" id="SwitchLang">
		<sf:macro object="LoadSwitchLang" design="" />
	</div>
</sf:if>
```

sf:if object="MultipleLanguages"

Evaluates true if the site wide search is enabled

sf:object="SwitchLang"

This object identifies the enveloping element of the SwitchLang code

Note: The element must include the following attribute to work correctly

```
id="SwitchLang"
```

sf:macro object="SwitchLang"

Loads the search code from the ObjectFragment Template

LoadSearch Attributes

Name	Description
Design	Name of the ObjectFragment template to use.

Mini cart

```
<sf:if object="ShopEnabled">
	<div sf:object="MiniCart" id="MiniCart">
		<sf:macro object="LoadMiniCart" design="" />
	</div>
</sf:if>
```

sf:if object="ShopEnabled"

Evaluates true if the site is a shop

sf:object="MiniCart"

This object identifies the enveloping element of the MiniCart code

Note: The element must include the following attribute to work correctly

```
id="MiniCart"
```

sf:macro object="LoadMiniCart"

Loads the search code from the ObjectFragment Template

LoadMiniCart Attributes

Name	Description
design	Name of the ObjectFragment template to use.

Index 1

```
<div sf:object="Index1" id="Index1">
	<sf:macro object="LoadIndex1" design="" orientation="" scroll="" popupdirection="" showhomelink=""
showlinkimagesublevels="" showlinkimagetoplevel=" " maxheight="" />
</div>
```

sf:object=Index1

This object identifies the enveloping element of the Index1 code

Note: The element must include the following attribute to work correctly

```
id="Index1"
```

sf:macro object="LoadIndex1"

Loads the search code from the Index Template

LoadIndex1 Attributes

Name	Description
design	Name of the preferred Index style to use.
orientation	Values: Horizontal or Vertical. Designates the orientation of the top level of the Index.
scroll	A number specifying the width (for Horizontal orientation) or height (for Vertical orientation) if the Index. If the Index width or height exceeds the number, scrollers will automatically be inserted.
popupdirection	Values: up, down, left or right. This attribute is for dropdown menu Index styles. The dropdown menus will popup in the specified direction.
showhomelink	Values: true or false. Automatically add an item in the Index that links to the Home page.
showlinkimagesublevels	Values: true or false. Show the link image, if any, of pages in sublevels of the Index.
showlinkimagetoplevel	Values: true or false. Show the link image, if any, of pages in the top level of the Index.
maxwidth	The link image can not be wider than this
maxheight	The link image can not be higher than this

Switch currency

```
<sf:if object="MultipleCurrencies">
	<div sf:object="SwitchCurrency" id="SwitchCurrency">
		<sf:macro object="LoadSwitchCurrency" design="" />
	</div>
</sf:if>
```

sf:if object="MultipleCurrencies"

Evaluates true if the multiple currencies are supported in shop

sf:object="SwitchCurrency"

This object identifies the enveloping element of the SwitchCurrency code

Note: The element must include the following attribute to work correctly

```
id="SwitchCurrency"
```

sf:macro object="LoadSwitchCurrency"

Loads the search code from the ObjectFragment Template

LoadSwitchCurrency Attributes

Name	Description
Design	Name of the Index template to use.

Login

```
<sf:if object="DisplayLoginForm">
	<div sf:object="Login" id="Login">
		<sf:macro object="LoadLogin" design="" />
	</div>
</sf:if>
```

sf:if object="DisplayLoginForm"

Evaluates true if the shop has member/reseller login

sf:object="Login"

This object identifies the enveloping element of the Login code

Note: The element must include the following attribute to work correctly

```
id="Login"
```

sf:macro object="LoadLogin"

Loads the login code from the ObjectFragment Template

LoadLogin Attributes

Name	Description
design	Name of the ObjectFragment template to use.

Index 2

```
<div sf:object="Index2" id="Index2">
<sf:macro object="LoadIndex2" design="" orientation="" scroll="" popupdirection="" showhomelink=""
showlinkimagesublevels="" showlinkimagetoplevel="" />
</div>
```

sf:if object="Index2"

This object identifies the enveloping element of the index code\

Note: The element must include the following attribute to work correctly

```
id="Index2"
```

sf:macro object="LoadIndex2"

Loads the code from the Index Template

LoadIndex2 Attributes

Name	Description
design	Name of the preferred Index style to use.
orientation	Values: Horizontal or Vertical. Designates the orientation of the top level of the Index.
scroll	A number specifying the width (for Horizontal orientation) or height (for Vertical orientation) if the Index. If the Index width or height exceeds the number, scrollers will automatically be inserted.
popupdirection	Values: up, down, left or right. This attribute is for dropdown menu Index styles. The dropdown menus will popup in the specified direction.
showhomelink	Values: true or false. Automatically add an item in the Index that links to the Home page.
showlinkimagesublevels	Values: true or false. Show the link image, if any, of pages in sublevels of the Index.
showlinkimagetoplevel	Values: true or false. Show the link image, if any, of pages in the top level of the Index.

maxwidth	The link image can not be wider than this
maxheight	The link image can not be higher than this

Content

```
<div sf:object="Content" id="Content">
	<sf:macro object="LoadContent" welcomepagedesign="" welcomeproductdesign="" pagedesign="" productdesign="" detailedproductdesign="" />
</div>
```

sf:object="Content"

This object identifies the enveloping element of the Content code

Note: The element must include the following attribute to work correctly

```
id="Content"
```

sf:macro object="LoadContent"

Loads the content code from the Pages Template

LoadContent Attributes

Name	Description
welcomepagedesign	
welcomeproductdesign	
pagedesign	
productdesign	
detailedproductdesign	

Application logo

sf: macro object="AppLogo"

Displays a small "Powered By" software logo button.

```
<sf:macro object="AppLogo" />
```

Page HTML components

Set banner-link image sizes

```
<sf:macro object="SetBannerLinkImageSizes" maxwidth="" maxheight="" />
```

sf: macro object="SetBannerLinkImageSizes"

Sets the desired width and height properties for the software to create the Banner images

Set page-link image sizes

```
<sf:macro object="SetPageLinkBoxImageSizes" fixwidth="" fixheight="" recwidth="" recheight="" maxwidth="" maxheight="" />
```

sf: macro object="SetPageLinkBoxImageSizes"

Sets the desired width and height properties for the software to create the PageLinkBox images

Breadcrumbs

```
<sf:if object="NotHomePage">
	<div sf:name="Breadcrumbs">
		<sf:repeat object="BreadcrumbsContent">
			<a href="<sf_BreadcrumbsPagelocation>" title="<sf_BreadcrumbsPagetitle>">
				<sf:value object="BreadcrumbsPagetitle" />
			</a>
		</sf:repeat>
	</div>
</sf:if>
```

sf:if object="NotHomePage"

Evaluates true if the current Page is not the Home Page

sf:name="Breadcrumbs"

This name identifies the enveloping element of the breadcrumbs loop

sf:repeat object="BreadcrumbsContent"

Loops through breadCrumbs for current page

sf:value object="BreadcrumbsPagetitle"

Displays the current breadcrumbs page title

<sf_BreadcrumbsPagelocation>

Outputs current breadcrumbs url for use in HTML tag attribute

<sf_BreadcrumbsPagetitle>

Outputs the current breadcrumbs page title for use in HTML tag attribute

Multiple pages

```
<sf:if object="MultiplePages">
</sf:if>
```

sf:if object="MultiplePages"

Evaluates true if this Page's Products/Paragraphs has been automatically separated into multiple pages.

First page

```
<sf:if object="IsFirstPage">
</sf:if>
```

sf:if object="IsFirstPage"

Evaluates true if this page is the first of multiple pages.

Last page

```
<sf:if object="IsLastPage">
</sf:if>
```

sf:if object="IsLastPage"

Evaluates true if this page is the last of multiple pages.

Html code top

```
<!-- Start HTMLCode top -->
<sf:if object="HasHtmlCodeTop">
<div sf:name="Htmlcode">
        <sf:macro object="Start_HTMLCodeTop_loop" />
```

```
                <div sf:object="HtmlcodeHtml" id="HtmlcodeHtml-<sf_.ID>"><sf:value
object="HtmlcodeHtml">Page Top HTML Code</sf:value></div>
        <sf:macro object="End_HTMLCodeTop_loop" />
</div>
</sf:if>
<!-- End HTMLCode top -->
```

sf:if object="HasHtmlCodeTop"

Evaluates true if user defined HTML code exists for the top

sf:name="Htmlcode"

This name identifies the enveloping element of the HTML code loop

sf:repeat object="HtmlCode"

Loops through all user defined HTML code

sf:if object="IsHtmlCodeTop"

Evaluates true if the current HTML is intended for the top

sf:name="HtmlcodeHTML"

This name identifies the enveloping element of the current HTML code

sf:value object="HtmlcodeHTML"

Displays the current HTML

Html code bottom

```
<!-- Start HTMLCode bottom -->
        <sf:if object="HasHtmlCodeBottom">
        <div sf:name="Htmlcode">
                <sf:macro object="Start_HTMLCodeBottom_loop" />
                        <div sf:object="HtmlcodeHtml" id="HtmlcodeHtml-<sf_.ID>"><sf:value
object="HtmlcodeHtml">Page Bottom HTML Code</sf:value></div>
                <sf:macro object="End_HTMLCodeBottom_loop" />
        </div>
        </sf:if>
        <!-- End HTMLCode bottom -->
```

sf:if object="HasHtmlCodeBottom"

Evaluates true if user defined HTML code exists for the bottom

sf:name="Htmlcode"

This name identifies the enveloping element of the HTML code loop

sf:repeat object="HtmlCode"

Loops through all user defined HTML code

sf:if object="IsHtmlCodeBottom"

Evaluates true if the current HTML is intended for the bottom

sf:name="HtmlcodeHTML"

This name identifies the enveloping element of the current HTML code

sf:value object="HtmlcodeHTML"

Displays the current HTML

Html code snippet area2

```
<!-- Start Page code snippet area2 -->
	<sf:if object="HasPageCodeSnippetArea2">
	<div sf:name="PageCodeSnippetArea2" class="PageCodeSnippetArea2">
		<sf:macro object="Start_PageCodeSnippetArea2_loop" /><sf:set
object="__RightStripHasContents=true" />
			<div sf:name="PageCodeSnippetArea2Content"
class="PageCodeSnippetArea2Content">
				<sf:macro object="Start_PageCodeSnippetArea2Content" />
					<sf:macro object="PageCodeSnippetArea2Content">Page code
snippet area2</sf:macro>
				<sf:macro object="End_PageCodeSnippetArea2Content" />
			</div>
		<sf:macro object="End_PageCodeSnippetArea2_loop" />
	</div>
	</sf:if>
	<!-- End Page code snippet area2 -->
```

sf:if object="HasHtmlCodeBottom"

Evaluates true if user defined HTML code exists for the bottom

sf:name="Htmlcode"

This name identifies the enveloping element of the HTML code loop

sf:repeat object="HtmlCode"

Loops through all user defined HTML code

sf:if object="IsHtmlCodeBottom"

Evaluates true if the current HTML is intended for the bottom

sf:name="HtmlcodeHTML"

This name identifies the enveloping element of the current HTML code

sf:value object="HtmlcodeHTML"

Displays the current HTML

Website HTML code snippet link box bottom

```
<!-- Start WebsiteLinkBoxBottom -->
                                    <sf:if object="HasWebsiteLinkBoxBottom">
                                    <div sf:name="WebsiteLinkBoxBottom" class="WebsiteLinkBoxBottom">
                                            <sf:macro object="Start_WebsiteLinkBoxBottom_loop" /><sf:set
object="__RightStripHasContents=true" />
                                                    <div sf:name="WebsiteLinkBoxBottomContent"
class="WebsiteLinkBoxBottomContent">
                                                            <sf:macro
object="Start_WebsiteLinkBoxBottomContent" />
                                                                    <sf:macro
object="WebsiteLinkBoxBottomContent">Website code snippet at bottom</sf:macro>
                                                            <sf:macro
object="End_WebsiteLinkBoxBottomContent" />
                                                    </div>
                                            <sf:macro object="End_WebsiteLinkBoxBottom_loop" />
                                    </div>
                                    </sf:if>
                                    <!-- End WebsiteLinkBoxBottom -->
```

Website HTML code snippet link box bottom

```
<!-- Start WebsiteLinkBoxTop -->
                                    <sf:if object="HasWebsiteLinkBoxTop">
                                    <div sf:name="WebsiteLinkBoxTop" class="WebsiteLinkBoxTop">
                                            <sf:macro object="Start_WebsiteLinkBoxTop_loop" /><sf:set
object="__RightStripHasContents=true" />
                                                    <div sf:name="WebsiteLinkBoxTopContent"
class="WebsiteLinkBoxTopContent">
                                                            <sf:macro
object="Start_WebsiteLinkBoxTopContent" />
                                                                    <sf:macro
object="WebsiteLinkBoxTopContent">Website code snippet at top</sf:macro>
                                                            <sf:macro
object="End_WebsiteLinkBoxTopContent" />
                                                    </div>
                                            <sf:macro object="End_WebsiteLinkBoxTop_loop" />
                                    </div>
                                    </sf:if>
```

```
<!-- End WebsiteLinkBoxTop -->
```

WebSite HTML code top

```
<!-- Start WebSite HTML code top -->
	<sf:if object="HasWebSiteHtmlCodeTop">
	<sf:macro object="Start_WebSiteHTMLCodeTop_loop" />
	<div sf:object="WebSiteHtmlCodeTop" id="WebSiteHtmlCodeTop-<sf_.ID>">
		<sf:macro object="Start_WebSiteHTMLCodeTopContent" />
			<sf:macro object="WebSiteHTMLCodeTopContent">Website Top HTML
Code</sf:macro>
		<sf:macro object="End_WebSiteHTMLCodeTopContent" />
	</div>
	<sf:macro object="End_WebSiteHTMLCodeTop_loop" />
	</sf:if>
	<!-- End WebSite HTML code top -->
```

WebSite HTML code bottom

```
<!-- Start WebSite HTML code bottom -->
<sf:if object="HasWebSiteHtmlCodeBottom">
<sf:macro object="Start_WebSiteHTMLCodeBottom_loop" />
<div sf:object="WebSiteHtmlCodeBottom" id="WebSiteHtmlCodeBottom-<sf_.ID>">
	<sf:macro object="Start_WebSiteHTMLCodeBottomContent" />
		<sf:macro object="WebSiteHTMLCodeBottomContent">Website Bottom HTML
Code</sf:macro>
	<sf:macro object="End_WebSiteHTMLCodeBottomContent" />
</div>
<sf:macro object="End_WebSiteHTMLCodeBottom_loop" />
</sf:if>
<!-- End WebSite HTML code bottom -->
```

Page HTML code snippet area 1

```
<!-- Start PageCodeSnippetArea1 -->
<sf:if object="HasPageCodeSnippetArea1">
	<div sf:name="PageCodeSnippetArea1" class="PageCodeSnippetArea1">
		<sf:macro object="Start_PageCodeSnippetArea1_loop" />
		<div sf:name="PageCodeSnippetArea1Content" class="PageCodeSnippetArea1Content">
			<sf:macro object="Start_PageCodeSnippetArea1Content" />
			<sf:macro object="PageCodeSnippetArea1Content">Page code snippet
area1</sf:macro>
			<sf:macro object="End_PageCodeSnippetArea1Content" />
		</div>
		<sf:macro object="End_PageCodeSnippetArea1_loop" />
	</div>
</sf:if>
<!-- End PageCodeSnippetArea1 -->
```

Index code snippet top

```
<!-- Start IndexCodeSnippetTop -->
<sf:if object="HasIndexCodeSnippetTop">
	<div sf:name="IndexCodeSnippetTop" class="IndexCodeSnippetTop">
		<sf:macro object="Start_IndexCodeSnippetTop_loop" />
		<div sf:name="IndexCodeSnippetTopContent" class="IndexCodeSnippetTopContent">
			<sf:macro object="Start_IndexCodeSnippetTopContent" />
			<sf:macro object="IndexCodeSnippetTopContent">Website code snippet at
top</sf:macro>
			<sf:macro object="End_IndexCodeSnippetTopContent" />
		</div>
		<sf:macro object="End_IndexCodeSnippetTop_loop" />
	</div>
</sf:if>
<!-- End IndexCodeSnippetTop -->
```

Index code snippet bottom

```
	<!-- Start IndexCodeSnippetBottom -->
<sf:if object="HasIndexCodeSnippetBottom">
<div sf:name="IndexCodeSnippetTop"
class="IndexCodeSnippetTop">
<sf:macro object="Start_IndexCodeSnippetBottom_loop"
/>
<div sf:name="IndexCodeSnippetBottomContent"
class="IndexCodeSnippetBottomContent">
<sf:macro
object="Start_IndexCodeSnippetBottomContent" />
<sf:macro
object="IndexCodeSnippetBottomContent">Website code snippet at bottom</sf:macro>
<sf:macro
object="End_IndexCodeSnippetBottomContent" />
</div>
<sf:macro object="End_IndexCodeSnippetBottom_loop"
/>
</div>
</sf:if>
<!-- End IndexCodeSnippetBottom -->
```

WebSite footnote

```
<!-- Start WebsiteFootnote -->
<div sf:name="WebSiteFootnote" class="WebSiteFootnote">
	<sf:value object="WebSiteFootnote">WebsiteFootnote</sf:value>
</div>
<!-- End WebsiteFootnote -->
```

Banner top

```
<sf:if object="HasBannerTop">
	<div sf:name="Banner">
		<sf:repeat object="BannerContent">
			<sf:if object="IsBannerTop">
				<a href="<sf_BannerLocation>" target="<sf_BannerTarget>">
					<div sf:name="BannerImage">
						<sf:macro object="BannerImage"  maxwidth=""
maxheight="" border="0" />
					</div>
					<div sf:name="BannerCaption">
						<sf:value object="BannerCaption" />
					</div>
				</a>
			</sf:if>
		</sf:repeat>
	</div>
</sf:if>
```

sf:if object="HasBannerTop"

Evaluates true if user defined banner links exists for the top

sf:name="Banner"

This name identifies the enveloping element of the banner links loop

sf:repeat object="BannerContent"

Loops through all banner links

sf:if object="IsBannerTop"

Evaluates true if the current banner link is intended for the top

sf:name="BannerImage"

This name identifies the enveloping element of the current banner image

sf:macro object="BannerImage"

Creates and displays the current banner image

BannerImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template

recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

sf:name="BannerCaption"

This name identifies the enveloping element of the current banner caption

sf:value object="BannerCaption"

Displays the banner caption content

Page link box

```
<sf:if object="PageLinkBox">
	<div sf:name="PageLinkBox">
		<sf:repeat object="PageLinkBoxLoop">
			<a target="<sf_PageLinkBoxTarget>" href="<sf_PageLinkBoxHRef>">
				<div sf:name="PageLinkBoxTitle">
					<sf:value object="PageLinkBoxTitle" />
				</div>
				<div sf:name="PageLinkBoxImage">
					<sf:macro object="PageLinkBoxImage" recwidth="" recheight=""
maxwidth="" maxheight="" border="" />
				</div>
				 <div sf:name="PageLinkBoxDescription">
					<sf:value="PageLinkBoxDescription" />
				</div>
			</a>
		  <sf:macro object="PageLinkBox_MoreDetails" />
		</sf:repeat>
	</div>
</sf:if>
```

sf:name="PageLinkBox"

Evaluates true if the current page contains page links

sf:name="PageLinkBox"

This name identifies the enveloping element of the page link loop

sf:repeat object="PageLinkBoxLoop"

Loops through page links

sf:name="PageLinkBoxTitle"

This name identifies the enveloping element of the current page link title

sf:value object="PageLinkBoxTitle"

Displays the page link title content

sf:name="PageLinkBoxImage"

This name identifies the enveloping element of the current page link title

sf:macro object="PageLinkBoxImage"

Creates and displays the current page link image

PageLinkBoxImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

sf:name="PageLinkBoxDescription"

This name identifies the enveloping element of the current page link description

sf:value object="PageLinkBoxDescription"

Displays the page link description

sf:macro object="PageLinkBox_MoreDetails"

This macro outputs an HTML anchor pointing to a linked Products more details page if it has one.

Page image

```
<sf:if="PageImage">
	<div sf:name="PageImage">
		<sf:macro object="PageImage" border="" recwidth="" recheight="" maxwidth=""
maxheight="" />
	</div>
	<div sf:name="PageImageCaption">
		<sf:value object="PageImageCaption" />
	</div>
</sf:if>
```

sf:if="PageImage"

Evaluates true if the current page image exists

sf:macro object="PageLinkBoxImage"

Creates and displays the page image

PageLinkBoxImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

sf:name="PageImageCaption"

This name identifies the enveloping element of the current page link description

sf:value object="PageImageCaption"

Displays the page image caption content

Page title

```
<sf:if="ShowPageTitle">
	<div sf:name="PageTitle">
		<sf:value object="PageTitle" />
	</div>
</sf:if>
```

sf:if="ShowPageTitle"

Evaluates true if the page has a title

sf:name="PageTitle"

This name identifies the enveloping element of the current page link title

sf:value object="PageTitle"

Displays the page title content

Shop discount message

```
<sf:if="ShowShopDiscountMessage">
	<div sf:name="ShopDiscountMessage">
		<sf:value object="ShopDiscountMessage" />
	</div>
</sf:if>
```

sf:if="ShowShopDiscountMessage"

Evaluates true if there is a shop disount message applicable to the current page

sf:name="ShopDiscountMessage"

This name identifies the enveloping element of the shop discount message

sf:value object="ShopDiscountMessage"

Displays the shop discount message content

Page introduction

```
<div sf:name="PageIntroduction">
	<sf:value object="PageIntroduction" />
</div>
```

sf:name="PageIntroduction"

This name identifies the enveloping element of the page introduction

sf:value object="PageIntroduction"

Displays the page introduction content

Page description

```
<div sf:name="PageDescription">
	<sf:value object="PageDescription" />
</div>
```

sf:name="PageDescription"

This name identifies the enveloping element of the page description

sf:value object="PageDescription"

Displays the page description content

Navigation sub-levels

```
<sf:macro object="SubPageNavigationSubLevels_Top" />
<sf:macro object="SubPageNavigationSubLevels_Bottom" />
```

sf:macro object="SubPageNavigationSubLevels_Top"

Designates the top position for the subpage navigation component.

sf:macro object="SubPageNavigationSubLevels_Bottom"

Designates the bottom position for the subpage navigation component.

Product loop

```
<sf:macro object="LoadProductLoop" />
```

sf:macro object="LoadProductLoop"

Loads ProductLoop template

Product footer

```
<div sf:name="PageFooter">
	<div sf:name="PageFootnote">
		<sf:value object="PageFootnote" />
	</div>
</div>
```

sf:name="PageFooter"

This name identifies the enveloping element of the page footer

sf:name="PageFootNote"

This name identifies the enveloping element of the page foot note

sf:value object="PageFootNote"

Displays the page footnote content

Banner bottom

```
<sf:if object="HasBannerBottom">
	<div sf:name="Banner">
		<sf:repeat object="BannerContent">
			<sf:if object="IsBannerBottom">
				<a href="<sf_BannerLocation>" target="<sf_BannerTarget>">
					<div sf:name="BannerImage">
						<sf:macro object="BannerImage"  maxwidth=""
maxheight="" border="0" />
					</div>
```

```
                                        <div sf:name="BannerCaption">
                                                <sf:value object="BannerCaption" />
                                        </div>
                                </a>
                        </sf:if>
                </sf:repeat>
        </div>
</sf:if>
```

sf:if object="HasBannerBottom"

Evaluates true if user defined banner links exists for the bottom

sf:name="Banner"

This name identifies the enveloping element of the banner links loop

sf:repeat object="BannerContent"

Loops through all banner links

sf:if object="IsBannerBottom"

Evaluates true if the current banner link is intended for the bottom

sf:name="BannerImage"

This name identifies the enveloping element of the current banner image

sf:macro object="BannerImage"

Creates and displays the current banner image

BannerImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

sf:name="BannerCaption"

This name identifies the enveloping element of the current banner caption

sf:value object="BannerCaption"

Displays the banner caption content

Mulitple pages index top

```
<sf:macro object="MultiplePagesIndexTop" />
```

sf:macro object="MultiplePagesIndexTop"

Outputs a div as the place holder for the multiple pages index before product/paragraph list

Mulitple pages index bottom

```
<sf:macro object="MultiplePagesIndexBottom" />
```

sf:macro object="MultiplePagesIndexBottom"

Outputs a div as the place holder for the multiple pages index after product/paragraph list

Product Loop HTML Components

Set cross promotion image sizes

```
<sf:macro object="SetProductCrossPromotionImageSizes" recwidth="" recheight="" maxwidth="" maxheight="" />
```

sf: macro object="SetProductCrossPromotionImageSizes"

Sets the desired width and height properties for the software to create the Banner images

SetProductCrossPromotionImageSizes Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

Set product image sizes

```
<sf:macro object="SetProductImageSizes" recwidth="" recheight="" maxwidth="" maxheight="" />
```

sf: macro object="SetProductImageSizes"

Sets the desired width and height properties for the software to create the Banner images

SetProductImageSizes Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

Set page-link image sizes

```
<sf:macro object="SetPageLinkBoxImageSizes" fixwidth="" fixheight="" recwidth="" recheight="" maxwidth="" maxheight="" />
```

sf: macro object="SetPageLinkBoxImageSizes"

Sets the desired width and height properties for the software to create the PageLinkBox images

Product loop

```
<form name="productForm">
	<sf:repeat object="Productloop">
		<sf:if object="IsTranslated">
			<div sf:name="Product">
				<!-- Product/Paragraph Information must go here -->
			</div>
		</sf:if>
	</sf:repeat>
</form>
```

<form name="productForm">

This HTML tag is needed to allow users to add products to a cart if the site is a Shop.

Note: This tag must envelope the product loop to work correctly.

sf:repeat object="Productloop"

Loops through products/paragraphs for the current page

sf:if object="IsTranslated"

Evaluates true is the product/paragraph is available for the current site language.

sf:name="Product"

This name identifies the enveloping element of the product/paragraph

Product heading

```
<div sf:object="ProductTableHeader" id="ProductTableHeader-<sf_ID>">
	<!—Product/Paragraph information like title, price, etc can go here -->
</div>
```

sf:object="ProductTableHeader"

This object identifies the enveloping element of the product heading.

Note: The element must include the following attribute to work correctly

```
id="ProductTableHeader-<sf_ID>"
```

Product bookmark

```
<sf:macro object="ProductBookmark" />
```

sf:macro object="ProductBookmark"

This macro outputs a hidden anchor for linking to this product in cross promotions etc.

Product title

```
<div sf:name="ProductTitle">
	<sf:value object="ProductTitle" />
</div>
```

sf:name="ProductTitle"

This name identifies the enveloping element of the product title.

sf:value object="ProductTitle"

Displays the product title content.

Product price

Simple method

The simple method allows for the placement of the price using one tag, with the shopping cart taking care of adding the original price and calculated price elements.

```
<sf:if object="ProductPrice">
	<sf:value object="ProductPriceIntro" />
	<div sf:name="ProductPrice">
		<sf:macro object="jsProductPrice" />
	</div>
</sf:if>
```

sf:if object="ProductPrice"

Evaluates true if the current product has a price.

sf:value object="ProductPriceIntro"

Displays the current products price introduction, i.e. "From" or "Only".

sf:name="ProductPrice"

This name identifies the enveloping element of the product price. Where a product discount applies, the original price and calculated price elements will be automatically placed in this element.

sf:macro object="jsProductPrice"

This macro displays product price.

Method for more placement control

This method allows for greater control over the placement of each element that constitutes a product. Each element can be placed anywhere within the product container element.

```
<div sf:name="ProductPriceIntro" class="ProductPriceIntro"></div>
<div sf:name="ProductPrice" class=" ProductPrice"></div>
<div sf:name="ProductPriceOriginal" class="ProductPriceOriginal"></div>
<div sf:name="ProductPriceCalculated" class="ProductPriceCalculated"></div>
<div sf:name="ProductIncTaxes" class="ProductIncTaxes"></div>
<sf:macro object="End_ProductPrice" />
```

sf:name="ProductPriceIntro"

Displays the current product price introduction, i.e. "From" or "Only".

sf:name="ProductPrice"

This name identifies the element for the product price.

sf:name="ProductPriceOriginal"

This name identifies the element for the original product price where a product discount is present. Where a product discount is present, the ProductPrice element will not be used.

sf:name="ProductPriceCalculated"

This name identifies the element for the calculated product price where a product discount is present. Where a product discount is present, the ProductPrice element will not be used.

sf:name="ProductIncTaxes"

This name identifies the element for tax texts associated with the product.

Cart quantity and icons

Simple method

The simple method allows for the placement of the quantity text box, Add To Basket icon and Favorites icon using one tag.

```
<sf:if object="ProductPrice">
	<div id="ProductIcons-<sf_ProductId>">
		<sf:macro object="QntyAndIcons" />
	</div>
</sf:if>
```

sf:if object="ProductPrice"

Evaulates true if the current product has a price (we can only purchase a product with a price).

id="ProductIcons-<sf_ProductId>"

This HTML attribute must be apart of the enveloping element for the cart quantity and icons code.

sf:macro object="QntyAndIcons"

This macro displays the quantity text box, Favorites and Add To Basket icons.

Method for more placement control

This method allows for greater control over the placement of each element. Each element can be placed anywhere within the product container element.

```
<sf:macro object="QntyAndIcons_QntyField" />
<sf:macro object="QntyAndIcons_AddToBasket" />
<sf:macro object="QntyAndIcons_Favorites" />
<sf:macro object="QntyAndIcons_AddToBasket_Text" />
<sf:macro object="QntyAndIcons_Favorites_Text" />
```

sf:macro object="QntyAndIcons_QntyField"

This macro displays the quantity text box. Only one is allowed per product.

sf:macro object="QntyAndIcons_AddToBasket"

This macro displays the Add To Basket button as an icon image.

sf:macro object="QntyAndIcons_Favorites"

This macro displays the Favorites button as an icon image.

sf:macro object="QntyAndIcons_AddToBasket_Text"

This macro displays the Add To Basket button as a text link.

sf:macro object="QntyAndIcons_Favorites_Text"

This macro displays the Favorites button as a text link.

Base price

```
<sf:if object="ShowBasePrices&HasBasePrice">
	<div sf:object="ProductBasePrice" id="ProductBasePrice-<sf_ID>">
		<sf:macro object="jsBaseProductPrice" />
	</div>
</sf:if>
```

sf:if object="ShowBasePrices&HasBasePrice"

Evaulates true if the current product has a base.

sf:object="ProductBasePrice"

This object identifies the enveloping element of the product base price.

Note: The element must also include the following attribute to work correctly

```
id="ProductBasePrice-<sf_ID>"
```

sf:macro object="jsBaseProductPrice"

This macro displays the product base price.

Product number

```
<sf:if object="ShowProductNumber">
	<div sf:name="ProductNumber">
		<sf:value object="ProductNumber" />
	</div>
</sf:if>
```

sf:if object="ShowProductNumber"

Evaulates true if the the product number exists.

sf:name="ProductNumber"

This name identifies the enveloping element of the product number

sf:value object="ProductNumber"

Displays the product number

Product weight

```
<sf:if object="ShowProductWeight">
	<sf:value object="LD_WEIGHT" />
	<div sf:name="ProductWeight">
		<sf:value object="ProductWeight" />
	</div>
	<div sf:name="ProductWeightUnit">
		<sf:value object="ProductWeightUnit" />
	</div>
</sf:if>
```

sf:if object="ShowProductWeight"

Evaulates true if the product weight exists.

sf:name="ProductWeight"

This name identifies the enveloping element of the product weight

sf:value object="ProductWeight"

Displays the product weight

sf:name="ProductWeightUnit"

This name identifies the enveloping element of the product weight unit

sf:value object="ProductWeightUnit"

Displays the product weight unit

Product stock

```
<sf:if object="ShowProductStock">
	<div sf:object="ProductStock" id="ProductStock-<sf_ID>"></div>
</sf:if>
```

sf:if object="ShowProductStock"

Evaluates true if product stock control is enabled.

sf:object="ProductStock"

This object identifies the enveloping element of the product stock

Note: The element must include the following attribute to work correctly

```
id="ProductStock-<sf_ID>"
```

Product image

```
<sf:if object="ProductImageSrc|ProductImageCaption|ProductThumbnailImageSrc|ProductMoreImages">
	<sf:macro object="ProductImageGroup" recwidth="" recheight="" maxwidth="" maxheight="" />
</sf:if>
```

sf:if object="ProductImageSrc|ProductImageCaption|ProductThumbnailImageSrc|ProductMoreImages"

Evaluates true if product image exists.

sf:macro object="ProductImageGroup"

Displays the Product Image.

ProductImageGroup Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

Product options

```
<sf:if object="HasProductOptions">
	<div sf:name="ProductOptions">
		<sf:macro object="ProductOptions" />
	</div>
</sf:if>
```

sf:if object="HasProductOptions"

Evaluates true if current product has selectable options.

sf:name="ProductOptions"

This name identifies the element enveloping the product options code

sf:macro object="ProductOptions"

This macro displays the product options.

Product disount message

```
<sf:if object="ShowProductDiscount">
	<div sf:name="ProductDiscountMessage">
		<sf:value object="ProductDiscountMessage" />
	</div>
</sf:if>
```

sf:if object="ShowProductDiscount"

Evaluates true if current product has a discount.

sf:name="ProductDiscountMessage"

This name identifies the element enveloping the product discount message

sf:value object="ProductDiscountMessage"

Displays the product discount message

Product international catalog number

```
<sf:if object="ProductInternationalCatalogNumber">
	<sf:value object="LD_EAN" />
	<div sf:name="ProductInternationalCatalogNumber">
		<sf:value object="ProductInternationalCatalogNumber" />
	</div>
</sf:if>
```

sf:if object="ProductInternationalCatalogNumber"

Evaluates true if current product has an EAN.

sf:name="ProductInternationalCatalogNumber"

This name identifies the element enveloping the product EAN.

sf:value object="ProductInternationalCatalogNumber"

Displays the product EAN.

Product introduction

```
<div sf:name="ProductIntroduction">
	<sf:value object="ProductIntroduction" />
</div>
```

sf:name="ProductIntroduction"

This name identifies the element enveloping the product introduction

sf:value object="ProductIntroduction"

Displays the product introduction content

Product description

```
<div sf:name="Product Description ">
	<sf:value object="Product Description " />
</div>
```

sf:name="ProductDescription"

This name identifies the element enveloping the product description

sf:value object="ProductDescription"

Displays the product description content

Product more details link

```
<sf:macro object="MoreDetails" />
```

sf:macro object="MoreDetails"

This macro outputs an anchor HTML element link to the product details page with the text LD_PRODUCT_CLICKHERE.

sf:macro object="MoreDetails_BuyNow"

This macro outputs an anchor HTML element link to the product details page with the text LD_BUY_NOW.

Multiple page index

```
<div sf:object="MultiplePageIndex">
	<sf:macro object="MultiplePageIndex" element="" elementclass="" class="" selectedclass="" />
</div>
<sf:macro object="PageBreak" />
```

sf:object="MultiplePageIndex"

This object identifies the enveloping element for the multiple pages code.

sf:macro sf:object="MultiplePageIndex"

This macro outputs a list of HTML anchors representing page numbers.

MultiplePageIndex Attributes

Name	Description
class	class of the outputted HTML anchor elements
selectedclass	class of the anchor representing the current page
element	optional enveloping element for each anchor e.g. div, td, li, etc
elementclass	class of enveloping element

sf:macro object="PageBreak"

This macro performs the actual page separation.

Products template components

Product detailed view

```
<form name="productForm">
	<div sf:name="Product">
		<!-- Product Information goes here -->
	</sf:repeat>
</form>
```

<form name="productForm">

This HTML tag is needed to allow users to add products to a cart if the site is a Shop.

Note: This tag must envelope the product div to work correctly.

sf:name="Product"

This name identifies the enveloping element of the product

Product heading

```
<div sf:object="ProductTableHeader" id="ProductTableHeader-<sf_ID>">
	<!—Product/Paragraph information like title, price, etc can go here -->
</div>
```

sf:object="ProductTableHeader"

This object identifies the enveloping element of the product heading.

Note: The element must include the following attribute to work correctly

```
id="ProductTableHeader-<sf_ID>"
```

Product bookmark

```
<sf:macro object="ProductBookmark" />
```

sf:macro object="ProductBookmark"

This macro outputs a hidden anchor for linking to this product in cross promotions etc.

Product title

```
<div sf:name="ProductTitle">
	<sf:value object="ProductTitle" />
</div>
```

sf:name="ProductTitle"

This name identifies the enveloping element of the product title.

sf:value object="ProductTitle"

Displays the product title content.

Product price

Simple method

The simple method allows for the placement of the price using one tag, with the shopping cart taking care of adding the original price and calculated price elements.

```
<sf:if object="ProductPrice">
	<sf:value object="ProductPriceIntro" />
	<div sf:name="ProductPrice">
		<sf:macro object="jsProductPrice" />
	</div>
</sf:if>
```

sf:if object="ProductPrice"

Evaluates true if the current product has a price.

sf:value object="ProductPriceIntro"

Displays the current products price introduction, i.e. "From" or "Only".

sf:name="ProductPrice"

This name identifies the enveloping element of the product price. Where a product discount applies, the original price and calculated price elements will be automatically placed in this element.

sf:macro object="jsProductPrice"

This macro displays product price.

Method for more placement control

This method allows for greater control over the placement of each element that constitutes a product. Each element can be placed anywhere within the product container element.

```
<div sf:name="ProductPriceIntro" class="ProductPriceIntro"></div>
<div sf:name="ProductPrice" class=" ProductPrice"></div>
<div sf:name="ProductPriceOriginal" class="ProductPriceOriginal"></div>
<div sf:name="ProductPriceCalculated" class="ProductPriceCalculated"></div>
<div sf:name="ProductIncTaxes" class="ProductIncTaxes"></div>
<sf:macro object="End_ProductPrice" />
```

sf:name="ProductPriceIntro"

Displays the current product price introduction, i.e. "From" or "Only".

sf:name="ProductPrice"

This name identifies the element for the product price.

sf:name="ProductPriceOriginal"

This name identifies the element for the original product price where a product discount is present. Where a product discount is present, the ProductPrice element will not be used.

sf:name="ProductPriceCalculated"

This name identifies the element for the calculated product price where a product discount is present. Where a product discount is present, the ProductPrice element will not be used.

sf:name="ProductIncTaxes"

This name identifies the element for tax texts associated with the product.

Cart quantity and icons

Simple method

The simple method allows for the placement of the quantity text box, Add To Basket icon and Favorites icon using one tag.

```
<sf:if object="ProductPrice">
	<div id="ProductIcons-<sf_ProductId>">
		<sf:macro object="QntyAndIcons" />
	</div>
</sf:if>
```

sf:if object="ProductPrice"

Evaulates true if the current product has a price (we can only purchase a product with a price).

id="ProductIcons-<sf_ProductId>"

This HTML attribute must be apart of the enveloping element for the cart quantity and icons code.

sf:macro object="QntyAndIcons"

This macro displays the quantity text box, Favorites and Add To Basket icons.

Method for more placement control

This method allows for greater control over the placement of each element. Each element can be placed anywhere within the product container element.

```
<sf:macro object="QntyAndIcons_QntyField" />
<sf:macro object="QntyAndIcons_AddToBasket" />
<sf:macro object="QntyAndIcons_Favorites" />
<sf:macro object="QntyAndIcons_AddToBasket_Text" />
<sf:macro object="QntyAndIcons_Favorites_Text" />
```

sf:macro object="QntyAndIcons_QntyField"

This macro displays the quantity text box. Only one is allowed per product.

sf:macro object="QntyAndIcons_AddToBasket"

This macro displays the Add To Basket button as an icon image.

sf:macro object="QntyAndIcons_Favorites"

This macro displays the Favorites button as an icon image.

sf:macro object="QntyAndIcons_AddToBasket_Text"

This macro displays the Add To Basket button as a text link.

sf:macro object="QntyAndIcons_Favorites_Text"

This macro displays the Favorites button as a text link.

Base Price

```
<sf:if object="ShowBasePrices&HasBasePrice">
	<div sf:object="ProductBasePrice" id="ProductBasePrice-<sf_ID>">
		<sf:macro object="jsBaseProductPrice" />
	</div>
</sf:if>
```

sf:if object="ShowBasePrices&HasBasePrice"

Evaulates true if the current product has a base.

sf:object="ProductBasePrice"

This object identifies the enveloping element of the product base price.

Note: The element must also include the following attribute to work correctly

```
id="ProductBasePrice-<sf_ID>"
```

sf:macro object="jsBaseProductPrice"

This macro displays the product base price.

Product number

```
<sf:if object="ShowProductNumber">
	<div sf:name="ProductNumber">
		<sf:value object="ProductNumber" />
	</div>
</sf:if>
```

sf:if object="ShowProductNumber"

Evaulates true if the the product number exists.

sf:name="ProductNumber"

This name identifies the enveloping element of the product number.

sf:value object="ProductNumber"

Displays the product number.

Product weight

```
<sf:if object="ShowProductWeight">
	<sf:value object="LD_WEIGHT" />
	<div sf:name="ProductWeight">
		<sf:value object="ProductWeight" />
	</div>
	<div sf:name="ProductWeightUnit">
		<sf:value object="ProductWeightUnit" />
	</div>
</sf:if>
```

sf:if object="ShowProductWeight"

Evaulates true if the product weight exists.

sf:name="ProductWeight"

This name identifies the enveloping element of the product weight.

sf:value object="ProductWeight"

Displays the product weight.

sf:name="ProductWeightUnit"

This name identifies the enveloping element of the product weight unit.

sf:value object="ProductWeightUnit"

Displays the product weight unit.

Product stock

```
<sf:if object="ShowProductStock">
	<div sf:object="ProductStock" id="ProductStock-<sf_ID>"></div>
</sf:if>
```

sf:if object="ShowProductStock"

Evaluates true if product stock control is enabled.

sf:object="ProductStock"

This object identifies the enveloping element of the product stock.

Note: The element must include the following attribute to work correctly

```
id="ProductStock-<sf_ID>"
```

Product image

```
<sf:if object=" ShowProductImage">
	<sf:macro object="ProductImageGroup" recwidth="" recheight="" maxwidth="" maxheight="" />
</sf:if>
```

sf:if object="ProductImageSrc|ProductImageCaption|ProductThumbnailImageSrc|ProductMoreImages"

Evaluates true if product image exists.

sf:macro object="ProductImageGroup"

Displays the Product Image.

ProductImageGroup Attributes

Name	Description

fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

Product options

```
<sf:if object="HasProductOptions">
	<div sf:name="ProductOptions">
		<sf:macro object="ProductOptions" />
	</div>
</sf:if>
```

sf:if object="HasProductOptions"

Evaluates true if current product has selectable options.

sf:name="ProductOptions"

This name identifies the element enveloping the product options code.

sf:macro object="ProductOptions"

This macro displays the product options.

Product disount message

```
<sf:if object="ShowProductDiscount">
	<div sf:name="ProductDiscountMessage">
		<sf:value object="ProductDiscountMessage" />
	</div>
</sf:if>
```

sf:if object="ShowProductDiscount"

Evaluates true if current product has a discount.

sf:name="ProductDiscountMessage"

This name identifies the element enveloping the product discount message.

sf:value object="ProductDiscountMessage"

Displays the product discount message.

Product international catalog number

```
<sf:if object="ProductInternationalCatalogNumber">
	<sf:value object="LD_EAN" />
	<div sf:name="ProductInternationalCatalogNumber">
		<sf:value object="ProductInternationalCatalogNumber" />
	</div>
</sf:if>
```

sf:if object="ProductInternationalCatalogNumber"

Evaluates true if current product has an EAN.

sf:name="ProductInternationalCatalogNumber"

This name identifies the element enveloping the product EAN.

sf:value object="ProductInternationalCatalogNumber"

Displays the product EAN.

Product introduction

```
<div sf:name="ProductIntroduction">
	<sf:value object="ProductIntroduction" />
</div>
```

sf:name="ProductIntroduction"

This name identifies the element enveloping the product introduction.

sf:value object="ProductIntroduction"

Displays the product introduction content.

Product description

```
<div sf:name="Product Description ">
	<sf:value object="Product Description " />
</div>
```

sf:name="ProductDescription"

This name identifies the element enveloping the product description.

sf:value object="ProductDescription"

Displays the product description content.

Product detailed description

```
<div sf:name="ProductDetailedDescription">
	<sf:value object="ProductDetailedDescription" />
</div>
```

sf:name="ProductDetailedDescription"

This name identifies the element enveloping the product detailed description.

sf:value object="ProductDetailedDescription"

Displays the product detailed description content.

Product highlights

```
<sf:if object="ProductHighlight">
	<div sf:name="ProductHighlight">
		<sf:value object="ProductHighlight" />
	</div>
</sf:if>
```

sf:if object="ProductHighlight"

Evaulates true id product has highlight content.

sf:name="ProductHighlight"

This name identifies the element enveloping the product highlights.

sf:value object="ProductHighlight"

Displays the product highlight content.

Product back button

```
<sf:macro object="BackButton" />
```

sf:macro object="BackButton"

This macro outputs an anchor HTML element link back to the product loop page.

Note: **The returned anchor has an id attribute of the value "BackLink"**

Product features

```
<div sf:name="ProductFeatures">
	<sf:repeat object="ProductFeaturesLoop">
		<sf:value object="ProductFeaturesTitle" />
		<sf:value object="ProductFeaturesDescription" />
	</sf:repeat>
```

```
</div>
```

sf:if object="ProductFeatures"

Evaulates true id product has features content.

sf:name="ProductFeatures"

This name identifies the element enveloping the product features.

sf:repeat object="ProductFeaturesLoop"

Loops through the product features.

sf:value object="ProductFeaturesTitle"

Displays the current features title content.

sf:value object="ProductFeaturesDescription"

Displays the current features Descrpition content.

Product cross promotions

```
<div sf:name="ProductCrossPromotion">
	<sf:repeat object="ProductCrossPromotionLoop">
		<sf:if object="PromotionHasContent">
			<a href="<sf_ProductCrossPromotionHRef>">
				<div sf:name="ProductCrossPromotionTitle">
					<sf:value object="ProductCrossPromotionTitle" />
				</div>
				<div sf:name="ProductCrossPromotionImage">
					<sf:macro object="ProductCrossPromotionImage" recwidth=""
recheight="" maxwidth="" maxheight="" />
				</div>
				<div sf:name="ProductCrossPromotionDescription">
					<sf:value object="ProductCrossPromotionDescription" />
				</div>
			</a>
		</sf:if>
	</sf:repeat>
</div>
```

sf:name="ProductCrossPromotion"

This name identifies the element enveloping the cross promotion links.

sf:repeat object="ProductCrossPromotionLoop"

Loops through cross promotions.

sf:if object="PromotionHasContent"

Evaluates true if the current cross promotion has content.

```
<sf_ProductCrossPromotionHRef>
```

Outputs current cross promotions url for use in HTML tag attribute.

sf:name="ProductCrossPromotionTitle"

This name identifies the element enveloping the cross promotion title.

sf:value object="ProductCrossPromotionTitle"

Displays the current cross promotion title

sf:name="ProductCrossPromotionImage"

This name identifies the element enveloping the cross promotion image.

sf:macro object="ProductCrossPromotionImage"

This macro displays the current cross promotion image.

ProductCrossPromotionImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

sf:name="ProductCrossPromotionDescription"

This name identifies the element enveloping the cross promotion description.

sf:value object="ProductCrossPromotionTitle"

Displays the current cross promotion description.

Index template HTML components

toplevel.html Components

Index1

```
<sf:if object="IsIndex1">
	<sf:if object="ShowIndex1HomeLink">
		<a href="<sf_HomeHref>" sf:object="idx1" id="idx1<HomeID>">
			<sf:value object="LD_HOME" />
		</a>
	</sf:if>
	<sf:repeat object="Index1Loop">
		<sf:if object="IsTranslated">
			<a href="<sf_NavigationHref>" sf:object="idx1" id="idx1<sf_NavigationID>">
				<sf:value object="NavigationTitle" />
				<sf:if object="ShowIndex1NavigationImage">
					<sf:macro object="NavigationImage" />
				</sf:if>
			</a>
		</sf:if>
	</sf:repeat>
</sf:if>
```

sf:if object="IsIndex1"

Evaluates true if called index is Index1.

sf:if object="ShowIndex1HomeLink"

Evaluates true if show home link is set for Index1.

sf:object="idx1"

This object identifies the anchor element enveloping the index content.

Note: The element must also include the following attribute to work correctly

```
id="idx1<sf_NavigationID>"
```

Note: In the case of the Home Link the following attribute must be included

```
id="idx1<HomeID>"
```

<sf_HomeHref>

Outputs the Home URL for use in a HTML tag attribute.

sf:value object="LD_HOME"

Displays The "Home" title.

sf:repeat object="Index1Loop"

Loops through the Index1 pages.

sf:if object="IsTranslated"

Evaluates true if the page is available in the current language.

<sf_NavigationHref>

Outputs the Current pages URL for use inside a HTML tag attribute.

sf:if object="IsTranslated"

Evaluates true if the page is available in the current language.

sf:value object="NavigationTitle"

Displays the current pages title content.

sf:macro object="NavigationImage"

This macro creates and ouputs the current pages link image.

sf:if object="ShowIndex1NavigationImage"

Evaluates true if Show Images has been set for Index1.

sf:macro object="Set_SubPageNavigationSubLevels_Position"

This macro specifies which position the subpage navigation component should appear in the page template. This macro only applies to the navigation styles that use subpage navigation – this includes all the styles beginning with VS.

```
<sf:macro object="Set_SubPageNavigationSubLevels_Position" position="top" />
<sf:macro object="Set_SubPageNavigationSubLevels_Position" position="bottom" />
```

Set_SubPageNavigationSubLevels_Position Attributes

Name	Description
position	Values: top or bottom. The subpage navigation component should be placed either in the top position or the bottom position.

Index 2

```
<sf:if object="IsIndex2">
	<sf:if object="ShowIndex1HomeLink">
		<a href="<sf_HomeHref>" sf:object="idx1" id="idx1<HomeID>"><sf_LD_HOME></a>
	</sf:if>
	<sf:repeat object="Index2Loop">
```

```
            <sf:if object="IsTranslated">
                  <a href="<sf_NavigationHref>" sf:object="idx1" id="idx1<sf_.ID>">
                        <sf:value object="NavigationTitle" />
                        <sf:macro object="NavigationImage" />
                  </a>
            </sf:if>
      </sf:repeat>
</sf:if>
```

sf:if object="IsIndex2"

Evaluates true if called index is Index 2.

sf:if object="ShowIndex2HomeLink"

Evaluates true if show home link is set for Index 2.

sf:object="idx2"

This object identifies the anchor element enveloping the index content.

Note: The element must also include the following attribute to work correctly.

```
id="idx2<sf_NavigationID>"
```

Note: In the case of the Home Link the following attribute must be included.

```
id="idx2<HomeID>"
```

<sf_HomeHref>

Outputs the Home URL for use in a HTML tag attribute.

sf:value object="LD_HOME"

Displays The "Home" title.

sf:repeat object="Index2Loop"

Loops through the Index2 pages.

sf:if object="IsTranslated"

Evaluates true if the page is available in the current language.

<sf_NavigationHref>

Outputs the Current pages URL for use inside a HTML tag attribute.

sf:if object="IsTranslated"

Evaluates true if the page is available in the current language.

sf:value object="NavigationTitle"

Displays the current pages title content.

sf:if object="ShowIndex2NavigationImage"

Evaluates true if Show Images has been set for Index 2.

sf:macro object="NavigationImage"

This macro creates and ouputs the current pages link image.

SubPageNavigationImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width
recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

sf:macro object="Set_SubPageNavigationSubLevels_Position"

This macro specifies which position the subpage navigation component should appear in the page template. This macro only applies to the navigation styles that use subpage navigation – this includes all the styles beginning with VS.

```
<sf:macro object="Set_SubPageNavigationSubLevels_Position" position="top" />
<sf:macro object="Set_SubPageNavigationSubLevels_Position" position="bottom" />
```

Set_SubPageNavigationSubLevels_Position Attributes

Name	Description
position	Values: top or bottom. The subpage navigation component should be placed either in the top position or the bottom position.

sublevels.html Components

Sub page navigation loop

```
<sf:repeat object="SubPageNavigationLoop">
	<sf:if object="IsTranslated">
		<a href="<sf_NavigationHref>" sf:object="idx1Sub" id="idx1Sub<sf_NavigationID>">
			<sf:value object="SubPageNavigationTitle" />
			<sf:value object="SubPageNavigationIntroduction" />
```

```
                        <sf:if object="ShowSubPageNavigationImage">
                                <sf:macro object="SubPageNavigationImage" />
                        </sf:if>
                </a>
        </sf:if>
</sf:repeat>
```

sf:object="idx1Sub"

This object identifies the anchor element enveloping the index content.

Note: The element must also include the following attribute to work correctly

```
id="idx1Sub<sf_NavigationID>"
```

sf:repeat object="SubPageNavigationLoop"

Loops through the sub page index for the current page.

sf:if object="IsTranslated"

Evaluates true if the page is available in the current language.

<sf_NavigationHref>

Outputs the Current pages URL for use inside a HTML tag attribute.

sf:if object="IsTranslated"

Evaluates true if the page is available in the current language.

sf:value object="SubPageNavigationTitle"

Displays the current pages title content.

sf:value object="SubPageNavigationIntroduction"

Displays the current pages introduction content.

sf:if object="ShowSubPageNavigationImage"

Evaluates true if Show Images has been set for Index1.

sf:macro object="SubPageNavigationImage"

This macro creates and ouputs the current pages link image.

SubPageNavigationImage Attributes

Name	Description
fixwidth	The image must have this width
fixheight	The image must have this width

recwidth	The image width is recommended by the template
recheight	The image height is recommended by the template
maxwidth	The image can not be wider than this
maxheight	The image can not be higher than this

Appendix 1

Colour mapping

C# Colours are organised within groups of elements called GC (Global Colours).

Example:
The colour (#f9f9ee) associated with C8 is mapped to the background of GC12 Product Description, Product Detailed Description and GC23 Page Image Caption.

See also:

Change theme Colours

Global Colour mapping

DOM Inspector

DOM inspectors make editing your website much easier by allowing you to inspect the website as displayed within the browser. This helps adjusting the template.

DOM stands for 'Document Object Model.' A DOM Inspector is a program or utility which explores the visual code structure of a website. They are very useful tools when locating rendering differences between Web browsers and performing alterations to websites.

DOM inspectors are available within most popular Web browsers as advanced options or are enhanced via extensions. Below is a list of several DOM inspectors:

FireBug for Mozilla Firefox
FireBug is the most commonly used DOM inspector, at the time of publication. It is a Mozilla Firefox extension which allows you to find which elements need alterations, edit values, and can display useful graphical diagrams in real time. This DOM inspector is useful for debugging and building sites for Gecko based browsers.
Links: Mozilla Firefox, FireBug, Gecko based browsers

Web Developer toolbar for Mozilla Firefox
The Web Developer toolbar is an extension for Mozilla Firefox. This extension adds a toolbar to Firefox which allows you to simply edit the some DOM elements by clicking the toolbar buttons. This is not a complete DOM Inspector, but proves to be a valuable tool when quickly referencing elements on a website.
Links: Mozilla Firefox, Web Developer

DOM Explorer for Microsoft Internet Explorer 6 – 7
DOM Explorer allows you to find which elements need alterations and edit in real time. This DOM Inspector has very limited functionality.
Links: Internet Explorer, DOM Explorer

Developer Tools for Microsoft Internet Explorer 8
Developer Tools allows you to you to find which elements need alterations, edit values, displays useful graphical diagrams in real time. It allows you to switch rendering modes and is similar to FireBug.

Links: Internet Explorer

Safari Web Inspector

At the time of publication, the Web Inspector may be visible if a nightly build of Safari is installed. The development name for Safari is WebKit. It is available within the 'Develop' toolbar. Note: If the Develop menu does not appear in the menu bar, open Safari preferences, click Advanced, and select "Show Develop menu in menu bar."

This DOM Inspector is useful for debugging and building sites for WebKit based browsers such as Safari, Chrome, Nokia, iPhone and to unknown extent Konqueror.
Links: WebKit Nightly, Webkit based browsers

Opera

Opera's Developer Tools allows you to find which elements need alterations, edit values, displays useful graphical diagrams in real time. This DOM inspector is useful for debugging and building sites for Opera browsers which are available on personal computers and mobile devices.
Links: Opera

Runtime directory

A runtime directory is a working folder containing all the website, current template and temporary files created by the software.

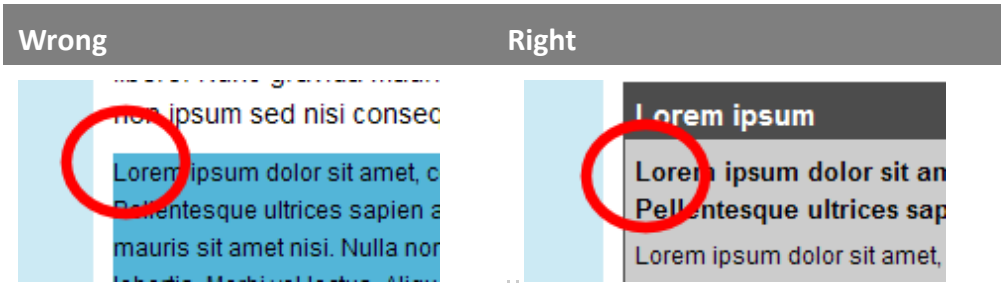
```
My Documents \ Name of software and Version number Websites \ Name of Website you are working on
```

Appendix 2: Testing Templates

To properly test templates you must use a shop which contains all elements for the website page and product styles.

When you test a template you must

1. Verify that all elements are displayed
2. That all elements display correctly
3. That switching between different Website themes and back works without breaking the template
4. That changes made in Customize Design mode are accurately reflected in the website.
5. Text areas must have padding around them. If you enable a background colour for a text area or if you see a text area with a background colour enabled, there should be some space between the text and the border of the coloured area.



6. That the “More Details” Link for products only comes up, if there ARE more details displayed on that page. This is different for each product style.
7. That design images fit properly into the assigned areas and can be replaced with images correctly cropped by ShopFactory.

Website Template

To test Pages and Indexes see Test pages, Test Product Loops, Test Product pages (More details), Test Indexes

Are all website elements displayed?

All website elements must be displayed. When using a shop containing all these elements, verify that you can see them on the home page:

Test – Can you see these Website Element	Y / N
Website title	
Company logo	
Website Slogan	
Index 1	

Index 2	
Search function	
Member-Log-in	
Language Selector for multilingual shops	
Website HTML areas 1, 2, 3, 4, 5, 6	
Content area	
Website Footer	
Website Design images	

Company Logo

The company Logo can be added on the ShopFactory Central Page. It must have a recommended Size setting and a Maximum Size setting. You should see these when adding a company Logo in the lower left hand corner of the image file dialog.

Test – Company Logo	Y / N
Can you see the company Logo on the page	
Does the recommended logo size display properly without overlapping text elements	
Does the maximum image size display properly	
At Logo MAXIMUM size is the Shop **TITLE** still displayed properly	
At Logo MAXIMUM size is the Shop **SLOGAN** still displayed properly	
At Logo MAXIMUM size is the Banner image still displayed properly	
At Logo MAXIMUM size are the INDEXES still displayed properly	
At Logo MAXIMUM size are Search and Language fields still displayed properly	
At Logo MAXIMUM size are the INDEXES still displayed properly	
At Logo MAXIMUM size is the Mini-cart still displayed properly	
When adding a new logo, can you use recommended and maximum sizes	

Website Title

Test – Website Title	Y / N
Can you see and easily read this element	
IF this element is on a **plain colour** background, is TEXT set to Auto-Colour?	
Can you edit the Font size	
Can you edit the Font type	

Can you edit the Font colour	
Does changing the Bold setting change the element	
Does changing the Italics setting change the element	
Can you change the Background Colour	
Can you add a Background Image to the element	
Can you remove the Background image again	
If there is a border, can you change the border colour	
Can you make the border colour transparent	
If the Title text is too long – does the template deal with this without breaking	

Website Slogan

Test – Website Slogan	Y / N
Can you see and easily read this element	
Is the slogan supporting 80 characters?	
IF this element is on a **plain colour** background, is TEXT set to Auto-Colour?	
Can you edit the Font size	
Can you edit the Font type	
Can you edit the Font colour	
Does changing the Bold setting change the element	
Does changing the Italics setting change the element	
Can you change the Background Colour	
Can you add a Background Image to the element	
Can you remove the Background image again	
If there is a border, can you change the border colour	

Index 1 , Index 2

See Test Indexes

Can you see two indexes

Search function

Test – Search function	Y / N
Can you see and easily read this element	
Can your turn this object on and off via the Shop setting?	

Member-Log-in

Test – Member Log-in	Y / N
Can you see and easily read this element	
Can your turn this object on and off via the Services menu?	

Language Selector for multilingual shops

Test – language seelection	Y / N
Can you see and easily read this element if a 2nd language is enabled?	
Does this element disappear, if other languages are disabled?	

Website Footer

Test – Website Footer	Y / N
Can you see and easily read this element	
IF this element is on a **plain colour** background, is TEXT set to Auto-Colour?	
Can you edit the Font size of inserted text	
Can you edit the Font type of inserted text	
Can you edit the Font colour of inserted text	
Does changing the Bold setting change the element	
Does changing the Italics setting change the element	
Can you change the Background Colour	
Can you add a Background Image to the element	
Can you remove the Background image again	

Website HTML areas 1, 2, 3, 4, 5, 6

HTML Website areas are added via ShopFactory Central, Edit Website HTML.

Test – Website HTML Areas	Y / N
Is HTML area **1** included in the website in the expected location	
Is HTML area **2** included in the website in the expected location	
Is HTML area **3** included in the website in the expected location	
Is HTML area **4** included in the website in the expected location	
Is HTML area **5** included in the website in the expected location	
Is HTML area **6** included in the website in the expected location	

Some templates do not support Website HTML areas 3 and 4. They may either not be displayed or, if displayed, will be shown above Area 5.

Website Design images

Design images set the look of the Website design. A website template may include multiple Design images. It must be possible via Customize design to easily replace and restore the current design image for the whole website as well as for a specific page only.

Design images have fixed sizes. These sizes are part of the template and can not be changed in ShopFactory. ShopFactory should enforce this size when selecting a larger image than the original design image.

Test – Design Image	Y / N
Is the design image shown in the click menu when selecting it	
Can you remove the design image	
Can you restore the design image	
Can you replace the design image	
When replacing a design image is the new image forced to the same size as the old image	
When using a smaller image is it tiled as per image properties	
If you change the image for the page only do other page keep their look	
If you change the image for the page style, do all pages with the page style change	
If you change the image for the website, do all website pages change	

Website Colours

Try changing the website colours to ensure changing the colours will work. The quickest way to do this is to select a new colour scheme from the list of colour schemes provided on the left side in Customize Design.

Test – Website colours	Y / N
When selecting a new website colour scheme – does the site switch colours as expected?	

Content

See Test Test pages, Test Product Loops, Test Product pages (More details)

Test Indexes

Depending on the website theme different index styles are provided for Index 1 and index 2.

Test – Test Indexes	Y / N
Can you see both indexes?	
Do both indexes display correctly?	
If the index is horizontal: do the scroll buttons display if there are more links or longer page names provided than fit in the available area?	
If the index is vertical: If a page name is longer than supported by the width of the index, does it wrap properly without looking too bad? See Vertical Indexes	
If the index is vertical: - Does the scroll button or scroll bar kick in if there are more pages than fit in the provided area? (In fixed size templates the scroll button should kick in, in unlimited height templates the page should simply grow larger.	
For both indexes: Can you change background colour of the left index area?	
For both indexes: Can you change background colour of the right index area?	
For both indexes: Can you change background colour of the centre index area?	
For both indexes: Can you change background colour of the mouse over right index area?	
For both indexes: Can you change background colour of the mouse over left index area?	
For both indexes: Can you change background colour of the mouse over centre index area?	
For both indexes: Can you change or add a background image to the left index area?	
For both indexes: Can you change or add a background image to right index area?	
For both indexes: Can you change or add a background image to centre index area?	
For both indexes: Can you change or add a background image to the mouse over right index area?	

For both indexes: Can you change or add a background image to the mouse over left index area?	
For both indexes: Can you change or add a background image to the mouse over centre index area?	
For both indexes: Can you remove the Background images again?	
For both indexes: Can you change the index container background colour independently of the area which contains the index (i.e. left column or row containing the index)?	
For both indexes: If there are border elements – can you change their colour	
For both indexes: If there are image elements in the design – can you easily change the image elements	
For both indexes: If name of page linked is on a **plain colour** background, is TEXT set to Auto-Colour?	
For both indexes: Can you edit the Font size of inserted text	
For both indexes: Can you edit the Font type of inserted text	
For both indexes: Can you edit the Font colour of inserted text	
For both indexes: Does changing the Bold setting change the element	
For both indexes: Does changing the Italics setting change the element	
For both indexes: Can you click on the page link to go to the page linked?	
For both indexes: If you mouse over a link – can you see the mouse over effect?	

Test Pages

Do pages Auto-Split as set in the Settings –Settings for this Website – Miscellaneous

Do max and recommended sizes work for all images including Banners

Do design images have fixed sizes and enforce them

Test – Pages	**Y / N**
Details for Search engines (check source code Meta tags)	
Page HTML code area 1,2,3,4	
Banner link	
Breadcrumbs	
Change Currency	
Mini Shopping cart	

Page Title	
Page Image	
Page Introduction	
Page Description	
Page Footer	
Product Loop	
Linkbox	
Subpage navigation	
Auto-Split Page counter	
Footer	

Page Title

Test – Page Title	Y / N
Can you see and easily read this element	
IF this element is on a **plain colour** background, is TEXT set to Auto-Colour?	
Can you edit the Font size	
Can you edit the Font type	
Can you edit the Font colour	
Does changing the Bold setting change the element	
Does changing the Italics setting change the element	
Can you change the Background Colour	
Can you add a Background Image to the element	
Can you remove the Background image again	
If there is a border, can you change the border colour	
Can you make the border colour transparent	

Page Image

Test – Page Image	Y / N
Can you see this element	

Can you easily replace this element with the Page Properties Wizard	
Does the image have a maximum and recommended size defined	
Does the page still look correct if maximum image size is used AND the Page Link box is displayed (Link to other pages and products).	
Can you add a caption	
Does the Alt Tag display on mouse over	
If there is a border, can you change the border colour	
If there is a border, can you make the border colour transparent	

Page Introduction

Test – Page Title	Y / N
Can you see and easily read this element	
IF this element is on a **plain colour** background, is TEXT set to Auto-Colour?	
Can you edit the Font size	
Can you edit the Font type	
Can you edit the Font colour	
Does changing the Bold setting change the element	
Does changing the Italics setting change the element	
Can you change the Background Colour	
Can you add a Background Image to the element	
Can you remove the Background image again	
If there is a border, can you change the border colour	
Can you make the border colour transparent	

Page Description

Test – Page Title	Y / N
Can you see and easily read this element	
IF this element is on a **plain colour** background, is TEXT set to Auto-Colour?	
Can you edit the Font size	

Can you edit the Font type	
Can you edit the Font colour	
Does changing the Bold setting change the element	
Does changing the Italics setting change the element	
Can you change the Background Colour	
Can you add a Background Image to the element	
Can you remove the Background image again	
If there is a border, can you change the border colour	
Can you make the border colour transparent	

Page Footer

Test – Page Footer	Y / N
Can you see and easily read this element	
IF this element is on a **plain colour** background, is TEXT set to Auto-Colour?	
Can you edit the Font size	
Can you edit the Font type	
Can you edit the Font colour	
Does changing the Bold setting change the element	
Does changing the Italics setting change the element	
Can you change the Background Colour	
Can you add a Background Image to the element	
Can you remove the Background image again	
If there is a border, can you change the border colour	
Can you make the border colour transparent	

Test Product Loops

See product elements and make sure all the ones required by the template are displayed correctly. Then check if their design can be adjusted via customize design. More to come.

Test – Product Loops	Y / N
Can you see products on the page	
Are all the elements you see displayed properly?	
If there is a Buy Now Button – does the purchasing function work correctly?	
Is there a link to a more details page if the product has elements which are not shown in the product loop?	
For all text elements displayed, can you correctly change the Font, font size, colour?	
Are text elements set to auto-colour	
For all design images display – can you replace them with other images?	
Is there a proper image size defined?	
Does the Multimedia link image display properly in the bottom right or left of the product /paragraph image, regardless of the size of the product image?	

Test Product Pages (More details)

Can you see all product elements assigned to the product on the more details page?

Make sure you use a test shop and product which has all properties assigned to it.

Test – Product Pages (More Details)	Y / N
Product ID	
Product Price	
Discount Note	
Price	
Quantity Box	
Quantity Unit	
Currency Symbol	
Discount Price	
Special Discount Message	
Base Price	
Weight	
Catalog Number	
Product Image	

Product Image Screen Tip	
Product Image Caption	
Multimedia Link	
Add to Basket button	
Product Headline	
Product Description	
Options and Choices	
Cross Promotions	
Highlights	
Longer Description	
Features	
Slideshow	
Product Discounts	
Link to Shipping charges	
Tax message	
Stock Message	
Stock Level	
Manufacturer	
Manufacturer Code	
Product Code	
Distributor Code	
Price Code	
EAN-UCC	
Design images	

1. Verify for all text items that they are set to auto-text colour.

2. Verify for the different items that it is possible to change background colours and add background images and remove them again.

3. Verify that the multimedia and slide-show link works correctly.

4. Verify that the product can be purchased.

Testing switching themes

When you switch between Website themes of different sizes, images inserted via the page or product wizards will be resized, as will be the width of the Linkbox. See Page Pre-sets and Product presets.

If the Website was built in a large theme:

When switching from large to small themes and back again, the image sizes and Link box width should automatically adjust to make sure everything fits.

If the Website was built in a small theme:

When switching from a small theme to a large theme, images will not change size, as ShopFactory does NOT physically increase image sizes to maintain display quality. However the Linkbox width should increase.

If the Website was built in a medium theme:

The same principles apply as for small and large themes.