



# Blue Chameleon CMS (Annex)

*For links outside this document,  
download the relevant chapter or the*

*Blue Chameleon Content Management System full documentation.*

May 8, 2012



# Contents

<b>1</b>	<b>Configuring and administration</b>	<b>13</b>
1.1	System interface . . . . .	13
1.2	Users and usergroups . . . . .	14
1.2.1	Users . . . . .	14
1.2.1.1	User groups that a user belongs to . . . . .	15
1.2.1.2	User management through time . . . . .	16
1.2.2	User groups . . . . .	16
1.2.3	User groups access rights . . . . .	16
1.3	Blue Chameleon Content Management System user rights . . . . .	17
1.3.1	Rights on headlines . . . . .	17
1.3.2	Rights on Articles . . . . .	18
1.3.3	Rights on page objects . . . . .	18
1.3.4	Rights on Images . . . . .	18
1.3.5	Rights on Search . . . . .	18
1.3.6	Rights on URLs . . . . .	18
1.3.7	Rights on columns . . . . .	19
1.3.8	Rights on publishing . . . . .	19
1.3.9	Rights on Data Base Publishing . . . . .	20
1.3.10	Rights on Editions . . . . .	20
1.3.11	Rights on Background Frames . . . . .	20
1.3.12	Rights on Downloads . . . . .	20
1.3.13	Rights on Site management . . . . .	21
1.3.14	Rights on Article Models . . . . .	21
1.3.15	Rights on Headline Models . . . . .	21
1.3.16	Rights on Notice boards . . . . .	21
1.3.17	Rights on System log . . . . .	22
1.3.18	Rights on Scripts . . . . .	22
1.3.19	Administrator rights . . . . .	22
1.3.20	Rights on Hermetic groups . . . . .	22
1.3.21	Information level . . . . .	23
1.3.22	Rights on Books . . . . .	23
1.3.23	Rights on Book Models . . . . .	23
1.3.24	Java option menu . . . . .	23
1.3.25	Connection speed . . . . .	23
1.3.26	Rights on User management . . . . .	24
1.3.27	Rights on Menus, Menu Templates . . . . .	24

1.3.28	Rights on Packages . . . . .	24
1.3.29	Rights on Forms . . . . .	24
1.4	Columns . . . . .	25
1.4.1	Publisher and objects . . . . .	25
1.4.2	Creating columns . . . . .	25
1.4.3	Managing columns . . . . .	26
1.4.3.1	Changing columns . . . . .	26
1.4.3.2	Updating columns . . . . .	26
1.4.3.3	Backups . . . . .	27
1.4.4	Blue Chameleon directory structure . . . . .	28
1.4.4.1	Virtual columns . . . . .	28
1.5	Publisher settings . . . . .	28
1.5.1	File types . . . . .	29
1.6	FTP profiles . . . . .	30
<b>2</b>	<b>Composing content</b>	<b>33</b>
2.1	Principles . . . . .	33
2.2	Headlines models . . . . .	33
2.2.1	Creating a new headline model . . . . .	34
2.2.2	Updating a headline model . . . . .	35
2.2.3	Contents of a headline model . . . . .	36
2.2.3.1	Enriching a headline model with frames . . . . .	37
2.2.3.1.1	Frame templates used by a headline model . . . . .	39
2.2.3.1.2	Updating frame number . . . . .	40
2.2.3.2	Meta Tags . . . . .	40
2.3	Frame templates . . . . .	41
2.3.1	Creating and updating frame templates . . . . .	41
2.3.2	"List" frames . . . . .	42
2.3.3	"Leaf" frames . . . . .	43
2.3.4	Frame style . . . . .	44
2.4	Headlines : creation and editing . . . . .	45
2.4.1	Editing a headline : a basic example . . . . .	45
2.4.2	Editing a leaf frame . . . . .	46
2.4.3	Editing a list frame . . . . .	47
2.4.3.1	Example for a leaf frame as used by a list . . . . .	48
2.5	Layouts . . . . .	50
2.6	Articles and article models . . . . .	50
2.7	Managing pages . . . . .	50
2.7.1	Publishing pages . . . . .	50
2.7.1.1	Multiple publish . . . . .	51
2.7.1.2	Scheduled publishing . . . . .	52
2.7.2	Properties of headlines . . . . .	53
2.7.3	Other management options for pages . . . . .	54

<b>3</b>	<b>Object management</b>	<b>57</b>
3.1	Images . . . . .	57
3.1.1	Uploading an image . . . . .	57
3.1.2	Image folders . . . . .	59
3.1.3	Managing images . . . . .	59
3.1.4	Integrating images . . . . .	59
3.2	Links . . . . .	60
3.2.1	Creating a URL link . . . . .	61
3.2.2	Managing URL links . . . . .	62
3.2.3	Integrating links . . . . .	62
3.3	Scripts . . . . .	64
3.3.1	Creating a script - Management . . . . .	64
3.3.2	Integrating a frame script . . . . .	65
3.3.3	Integrating a link style script . . . . .	66
3.4	Downloads . . . . .	67
3.4.1	Creating a download . . . . .	67
3.4.2	Managing downloads . . . . .	68
3.4.3	Integrating downloads . . . . .	68
3.5	Menus . . . . .	69
3.5.1	Menu scripts . . . . .	69
3.5.2	Menu models . . . . .	71
3.5.3	Creation of a menu . . . . .	71
3.5.3.1	Filling of a menu . . . . .	72
3.5.3.2	Menu result and integration . . . . .	74
3.5.4	More complex menus . . . . .	75
3.5.4.1	A menu script that handles children menu items . . . . .	75
3.5.4.2	A menu with children menu items . . . . .	75
3.6	Forms . . . . .	78
3.6.1	Form models . . . . .	78
3.6.1.1	Form model management - parameters . . . . .	79
3.6.2	Defining forms . . . . .	79
3.6.2.1	Updating a form . . . . .	80
3.6.3	Composing a form . . . . .	80
3.6.3.1	Editing a form page . . . . .	81
3.6.3.2	Form page element types . . . . .	82
3.6.3.3	Editing a form page element . . . . .	83
3.6.3.4	After form is submitted . . . . .	84
3.6.3.5	A form result . . . . .	84
3.6.3.6	A form with several pages . . . . .	84
3.6.4	Integrating a form . . . . .	85
3.7	Variable files . . . . .	86
3.7.1	Example of varfile contents . . . . .	86
3.7.2	Varfiles as variables . . . . .	87

<b>4</b>	<b>Other elements</b>	<b>89</b>
4.1	Site management . . . . .	89
4.1.1	Uploading files . . . . .	89
4.1.2	Creating directories . . . . .	90
4.1.3	Managing files and directories . . . . .	91
4.1.4	Executing SQL . . . . .	91
4.1.5	Export and Import . . . . .	93
	4.1.5.1 Generating export files - importing them . . . . .	93
	4.1.5.2 Export file contents . . . . .	94
4.2	Model defaults . . . . .	96
4.2.1	Model defaults and headlines . . . . .	96
4.3	Books . . . . .	97
4.3.1	Book models . . . . .	97
4.3.2	Creating books . . . . .	97
4.3.3	Modifying a book . . . . .	99
4.4	Notice boards . . . . .	100
4.4.1	Creating and managing notice boards . . . . .	100
4.4.2	Association with a column . . . . .	100
4.4.3	Notice board entries . . . . .	101
4.5	Editions . . . . .	102
4.5.1	General edition management . . . . .	103
4.5.2	Publishing editions . . . . .	104
4.6	Options . . . . .	104
4.6.1	Version . . . . .	104
4.6.2	System Logs . . . . .	105
4.6.3	Expiration report . . . . .	106
4.7	Searching for content . . . . .	106
4.8	Management of objects . . . . .	108
<b>5</b>	<b>Packages</b>	<b>109</b>
5.1	Mandatory scripts . . . . .	109
5.1.1	PackageInstall.phs . . . . .	109
5.1.2	PackageGroupModify.phs, PackageGroupModify1.phs . . . . .	110
	5.1.2.1 Result : package group rights . . . . .	110
5.1.3	PackageMain.phs . . . . .	113
	5.1.3.1 Result : package main page . . . . .	114
5.1.4	PackageUninstall.phs . . . . .	116
5.2	Package general management . . . . .	117
5.2.1	Installing a package . . . . .	117
5.2.2	Updating a package . . . . .	117
5.2.3	Uninstalling a package . . . . .	118
5.3	Using object types and objects to compose headlines . . . . .	118
5.3.1	Enabling a leaf frame to handle packages and an object . . . . .	119
5.3.2	PackageObjTypeList.phs . . . . .	120
5.3.3	PackageObjectList.phs . . . . .	122
5.3.4	PackageObjectName.phs . . . . .	122

5.3.5	PackageObjectView.phs . . . . .	124
5.4	Using attributes . . . . .	125
5.4.1	Enabling a frame to handle attributes . . . . .	125
5.4.2	Scripts for attribute . . . . .	126
5.5	Featuring multiple objects on the same headline . . . . .	127
5.5.1	A leaf frame that handles multiple objects . . . . .	128
5.5.2	Packages : defining more of them . . . . .	130
5.5.3	Object and attribute scripts for this case (guidelines) . . . . .	130
5.5.3.1	The PackageObjectView.phs script for this example . . . . .	132
<b>6</b>	<b>Annex</b> . . . . .	<b>135</b>
6.1	Leaf frame variables . . . . .	135
6.1.1	"Table" variables . . . . .	137
6.1.2	"Select" variables . . . . .	137
6.1.3	"SQL select" variables . . . . .	138
6.2	System script glossary . . . . .	138
6.2.1	Headline model scripts . . . . .	138
6.2.2	Leaf frame model scripts . . . . .	139
6.3	Icon glossary . . . . .	141
6.4	Form page elements : detailed . . . . .	143
6.4.1	"Active" elements . . . . .	143
6.4.1.1	Checkboxes . . . . .	143
6.4.1.2	Radiobuttons . . . . .	144
6.4.1.3	Listboxes . . . . .	144
6.4.1.4	Input field and text area . . . . .	145
6.4.1.5	Form label - 'with check box' . . . . .	146
6.4.2	"Passive" elements . . . . .	146
6.4.2.1	Hidden field . . . . .	146
6.4.2.2	Form image . . . . .	146
6.4.2.3	Form label - 'normal' . . . . .	146
6.4.2.4	Form separator . . . . .	147





# List of Figures

1.1	The Publisher's main page. . . . .	13
1.2	Defining a new user. . . . .	15
1.3	Groups a user belongs to. . . . .	15
1.4	Defining a user group. . . . .	16
1.5	The most important access rights as defined for a user group. . . . .	17
1.6	Creating a new column. . . . .	26
1.7	Updating a column. . . . .	27
1.8	The properties of your Publisher. . . . .	29
1.9	Defining a FTP profile. . . . .	31
2.1	Creating a headline model. . . . .	34
2.2	Here, a headline model can be updated and its children frame templates be chosen. . . . .	36
2.3	Thanks to this, the pages created on this model will have meta-tags called Author, LastUpdate and a Page Title. . . . .	41
2.4	Creating a new headline. . . . .	45
2.5	First edit ; as both frames use only one template ("Basic text frame" and "Side Menu List"), so clicking on the icon shows them immediately. . . . .	46
2.6	Filling in with information the leaf's various variables. . . . .	47
2.7	A preview of the page. . . . .	48
2.8	Populating a list frame. . . . .	49
2.9	Publishing several headlines in one single time. . . . .	51
2.10	Configuring and viewing data related to headline "Home Page". . . . .	53
2.11	Managing headlines in a file system way. . . . .	55
3.1	Uploading an image. . . . .	58
3.2	Creating a folder for images. . . . .	59
3.3	From here, all images and folders can be globally managed. . . . .	60
3.4	Integrating an uploaded image. . . . .	61
3.5	Creating a link. . . . .	62
3.6	Putting a previously-defined URL link. . . . .	63
3.7	Creating a script to be used in a leaf frame. . . . .	65
3.8	Uploading a file that will be used for downloading purposes. . . . .	67
3.9	Integrating a download, with a 'download' variable. . . . .	69
3.10	Creating a menu model. . . . .	71
3.11	Creating a menu "Basic top menu" based upon model 'Top Menu'. . . . .	72
3.12	Creating a form model. . . . .	78

3.13	Adding parameters to a form model. . . . .	79
3.14	Creating a form based on the example model. . . . .	79
4.1	Uploading a file to be used in headline contents. . . . .	90
4.2	Creating a new directory, directly under the «#SQLWWWHOME#» root. . . . .	91
4.3	Editing a text-based file. . . . .	92
4.4	Outputting the contents of a table. . . . .	93
4.5	Exporting the current column. . . . .	94
4.6	An example of export file contents. . . . .	95
4.7	Creating a model default for a headline model. . . . .	96
4.8	Creating a book model. . . . .	98
4.9	Creating a book. . . . .	98
4.10	From here, book elements can be managed. . . . .	100
4.11	Creating a general notice board. . . . .	101
4.12	Manually adding two entries to this notice board. . . . .	102
4.13	Checking one's own activity for last week. . . . .	106
4.14	Performing a search on objects which publishing name begin by 'Josh'. . . . .	107
4.15	From here, any object can be managed. . . . .	108
5.1	Modifying this package's rights for a particular user group. . . . .	110
5.2	The main page for package Festivals. . . . .	114
5.3	The first install of the package "Festivals". . . . .	117
5.4	The process of selecting an object type and an object for a leaf frame that is package-enabled. . . . .	119
5.5	A selection of object types. . . . .	121
5.6	Finally choosing the object. . . . .	123
5.7	A page design that calls for multiple objects to be featured. . . . .	128
5.8	The 3-object frame after all relevant choices of objects and attributes have been made. . . . .	132
6.1	Composing a checkbox selection for a form. . . . .	144

# List of Tables

5.1	Scripts, functions and variables as used for attribute setting . . . . .	126
6.1	Leaf frame variables . . . . .	136
6.2	Leaf frame variables (continued) . . . . .	137
6.3	System scripts for headline, list and container models . . . . .	139
6.4	System scripts for leaves . . . . .	140
6.5	System scripts for leaves (continued) . . . . .	141
6.6	Blue Chameleon Content Management icons . . . . .	142
6.7	Blue Chameleon Content Management icons (continued) . . . . .	143



# Chapter 6

## Annex

### 6.1 Leaf frame variables

Variables of a leaf frame must be given in the following manner,

*Name;Label;Type; [Data]*

where :

- *Name* is the name of the variable, that will be used in the leaf frame model's contents;
- *Label* is the label under which, in the context of leaf frame editing, the variable will be prompted ;
- *Type* is an identifier describing the type of variable (see Tables 6.1, 6.2 below) ;
- *Data* is a string of parameters : it is only given for 'select'- or 'SQL select'- type variables (6.1.2, 6.1.3) and empty for all of the others.

Table 6.1: Leaf frame variables

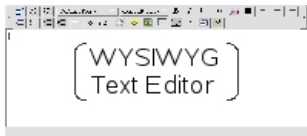
Type id.	Type	Example	Leaf frame edit
1	Text edit	<code>_Text;City;1;</code>	City : <input type="text"/>
2	Text area	<code>_Comment;Comments;2;</code>	Comments : 
3	Image	<code>_Pic;Photo;3;</code>	Photo : <No image selected> <input type="button" value="Choose image"/> <input type="button" value="No image"/> <input type="button" value="Edit image map links"/>
4	Link	<code>_Link;Link;4;</code>	Link : <No link selected> <input type="button" value="Choose link"/> <input type="button" value="No link"/>
5	Script (view)	<code>_Skr;Script;5;</code>	Script : <input type="text" value="-----"/> <input type="button" value="v"/> paramètres: <input type="text"/>
6	Shared varfile	<code>_Varf;File;6;</code>	File : not selected <input type="button" value="Change varfile"/>
7	Table	see 6.1.1	Table : <input checked="" type="radio"/> internal format (;) <input type="radio"/> Tabs <input type="radio"/> Spaces <input type="text"/>
8	Script (edit)	Deprecated	Deprecated
9	Select	see 6.1.2	Country : <input type="button" value="Canada"/> <input type="button" value="v"/>
10	Download	<code>_Dwnl;Download;10;</code>	Download : No download has been selected. <input type="button" value="Change download"/> <input type="button" value="No download"/>

Table 6.2: Leaf frame variables (continued)

11	Anchor	<code>_Ank;Anchor;11;</code>	Anchor : <input type="text"/>
12	Menu	<code>_Menu;Menu;12;</code>	Menu : No menu selected <input type="button" value="Change menu"/> <input type="button" value="No menu"/>
13	SQL select	see 6.1.3	Festival : <input type="text" value="Mittelalter und Goth Fest"/> ▾
14	Package object	<code>P01;Main Artist;14;</code>	Main Artist : <b>Josh "Lizard" Jones</b> <input type="button" value="Choose Object"/> <input type="button" value="No Object"/>
15	Attribute	<code>P0Style2;Display style 2;15;</code>	Display style 2 : <b>Small showcase left</b> <input type="button" value="Choose Attribute"/> <input type="button" value="No Attribute"/>



*These variables are then practically integrated in leaf frames thanks to system scripts as detailed at Tables 6.4 and 6.5.*

### 6.1.1 "Table" variables

### 6.1.2 "Select" variables

They ultimately serve as to provide, as featured on Table 6.1, a drop-down menu filled with elements.

The variable is set for instance as follows :

```
_Cntr;Country;9;USA/Canada/UK/...
```

Here, the [Data] string lists values, separated by slashes, that will be featured in the drop-down menu while editing the leaf frame. Upon previewing (and after, publication) of the headline, the variable will be set to the very value that was chosen, for instance :

## Headline models / Modify / a leaf frame mdl :

```
...
Born in : <_Cntr>
...
```

Update

[Frame edit] :

Country : Canada

USA  
Canada  
UK  
...

Preview

Born in : Canada

The [Data] string can be defined in a more elaborate manner with value=label couples, also separated by slashes, for instance as follows :

```
_Rezur;Reservation;9;-1=N.A./0=Non necessary/1=Mandatory
```

During editing of the frame, the drop-down menu will display the labels "N.A.", "Non necessary"... and, upon validation of the headline, the variable <\_Rezur> will be assessed the corresponding value (-1,0,...).

### 6.1.3 "SQL select" variables

Those variables work in a similar manner as the previous ones, i.e. a drop-down menu is provided, with elements corresponding to values. Except here, the couples value=label are generated by a SQL query, a SELECT, done so as **two columns** of a table are the result of the query :

```
_Price;Festival;13;SELECT DISTINCT(azName),mPrice FROM OPS_FESTIVALS
a,OPS_TICKET_PRICE b WHERE a.idFestival=b.idFestival
```

This query will generate two columns with the festival name and its corresponding price, and, upon validation of the headline, the variable <\_Price> will amount to the chosen festival.

## 6.2 System script glossary

### 6.2.1 Headline model scripts

During the composing of a headline model's contents (and also for list and container models), some system scripts are called. Table 6.3 sums them all.



*For 'Text edit', 'Text area', 'Select' and 'SQL select' variables, they are simply inserted via <\_Variable>, e.g. <\_Text>, <\_Comment>, <\_Cntr>, <\_Price>.*



Table 6.3: System scripts for headline, list and container models

Name	Call & Function
design.phs	<pre>«INCLUDE design.phs»</pre> <p>Enables the <b>Preview</b> and <b>Done!</b> buttons to appear when a headline created from this model is edited. It is therefore a <b>mandatory</b> script.</p>
frame.phs	<pre>«INCLUDE frame.phs;_ParentId=«_FSId»;_FrameIndex={1,2,...}»</pre> <p>For models that can have children (headlines, containers, lists), loads the frame number 1, 2,... This script must be called as many times as model has children frames, and those calls must be disposed, HTML-wise, in a manner that mirrors the desired layout of the page.</p>
metatags.phs	<pre>«INCLUDE metatags.phs»</pre> <p>Placed in the &lt;HEAD&gt;...&lt;/HEAD&gt; section of the headline model's contents, inserts into the same section of the rendered .htm page the meta-tags that were filled on headline's <a href="#">Manage</a> page.</p>
ListItems.phs	<pre>«INCLUDE ListItems.phs;_ListFrameItems="_ItemCount"»</pre> <p>In the case of a List model, puts into the variable <code>_ItemCount</code> (that has to be initialized before call) the number of list items that the list currently holds. It is then used to call <code>frame.phs</code> this very number of times.</p>
form.phs	<pre>«INCLUDE form.phs»</pre> <p>Will enable the headlines generated from this model to use forms.</p>

## 6.2.2 Leaf frame model scripts

Leaf frame models have also their system scripts, aimed at loading their variables, as detailed in Tables 6.4 and 6.5.

Table 6.4: System scripts for leaves

image.phs	<pre>«INCLUDE image.phs;pImage="_Pict";Html="border='0'..."»</pre> <p>Loads the image as referred to by the variable <code>_Pict</code>; the <code>Html="..."</code> string can be used to give any HTML additional parameter. As a result, the following tag will be generated :</p> <pre>&lt;IMG WIDTH=.. HEIGHT=.. ALT="..." SRC="..." <i>additional parameters</i>&gt;</pre> <pre>«INCLUDE image.phs;pImage="_Pict";_URLOnly=1;_Return="_Url"»»</pre> <p>This calls will store in variable <code>_Url</code> the address of the image.</p>
href.phs	<pre>«INCLUDE href.phs;_Link="_Link";Parameters»</pre> <p>Inserts a link as referred to by variable <code>_Link</code>, in different manners as specified by <i>Parameters</i> :</p> <ul style="list-style-type: none"> <li>● <b>StartTag</b> : generates the tag <code>&lt;A HREF="..."&gt;</code> if any link has been selected by the user ;</li> <li>● <b>EndTag</b> : generates the closing tag <code>&lt;/A&gt;</code> if necessary ;</li> <li>● <b>NoText</b> : the generated <code>&lt;A HREF="..."&gt;</code> tag is trimmed so that it does not contain any redundant blanks or lines ;</li> <li>● <b>Target</b> : will add the specified target to the <code>&lt;A HREF="..." TARGET="Target"&gt;</code> tag ;</li> <li>● <b>JustUrl</b> : (as a unique parameter) returns only the address of the link ;</li> <li>● <b>_Return=_ReturnVAR</b> : makes <code>href.phs</code> not give out HTML tags, but specific information as stored in a <code>_ReturnVAR</code> variable that must be initialized before call. The various <code>_ReturnVARs</code> specify the information : <ul style="list-style-type: none"> <li>◇ <code>_ReturnAddress</code> contains the URL of the link</li> <li>◇ <code>_ReturnType</code> contains the type of the link. Possible values are article, headline, URL, download, anchor and book</li> <li>◇ <code>_ReturnStyle</code> contains the name of the file containing the link style</li> <li>◇ <code>_ReturnStyleId</code> contains the Id of the link style.</li> </ul> </li> <li>● <b>AddDomain</b> : forces the coding of the domain specified for the publisher in the URL ;</li> <li>● <b>_OPSLink</b> : forces the link to be made to the OPS name of the article/headline/page object even if the user has indicated an publication name.</li> </ul>
scriptfile.phs	<pre>«INCLUDE scriptfile.phs»</pre> <p>Enables the leaf frame to load scripts. If the published page is meant to be used dynamically, the commands <code>«VAR NEW _Dynamic»</code> <code>«_Dynamic="yes"»</code> must be placed before the call.</p>
varphile.phs	<pre>«INCLUDE varfile.phs;_Action={1,2};_File="_MyFile"»</pre> <p>Loads (<code>_Action=1</code>) or unloads (<code>_Action=2</code>) the variables defined in the <code>varfile</code> as referred to by <code>_MyFile</code>. Between these two commands, the variables defined by the selected <code>varfile</code> can be used as if they were variables defined for the current leaf.</p>

Table 6.5: System scripts for leaves (continued)

<code>table.phs</code>	«INCLUDE <code>table.phs</code> ;_Table="_Tbl";_Align="Align"» Inserts a table as referred to by variable <code>_Tbl</code> with the <code>Align</code> parameter being any of the alignments used for TD tags (left, right...)
<code>download.phs</code>	«INCLUDE <code>download.phs</code> ;_Download="_Dwnl"» Generates the <code>&lt;A HREF="..."&gt;</code> , <code>&lt;/A&gt;</code> tags around the download description as relevant to the <code>_Dwnl</code> variable.
<code>anchor.phs</code>	«INCLUDE <code>anchor.phs</code> ;_Anchor="_Ank"» Generates the <code>&lt;A NAME="..."&gt;</code> tag for the anchor as referred to by variable <code>_Ank</code> .
<code>showmenu.phs</code>	«INCLUDE <code>showmenu.phs</code> ;_Menu=_Menu» Loads the menu as referred to by the variable <code>_Menu</code> .
<code>PackageObjectView.phs</code>	see 5.3.5, 5.4

## 6.3 Icon glossary

Tables 6.6 and 6.7 sum up what is the role icons as used during the editing of pages and in other contexts.

Table 6.6: Blue Chameleon Content Management icons


















Icon	[Context] and Action
 (Layout)	<p><b>[Headline edit]</b></p> <ul style="list-style-type: none"> <li>•If a headline, list, container can use several types of frames (2.2.3.1.1), this icon leads to a page where the frame to use there is selected.</li> </ul>
 (Edit)	<p><b>[Headline edit]</b></p> <ul style="list-style-type: none"> <li>•Edits a leaf frame, i.e. leads to a screen where its variables (text areas,...) are filled.</li> </ul> <p><b>[Menu item (3.5.3.1)]</b></p> <ul style="list-style-type: none"> <li>•Edits the menu item's text and link.</li> </ul> <p><b>[Form model (3.6.1.1)]</b></p> <ul style="list-style-type: none"> <li>•Edits that parameter of a form model.</li> </ul> <p><b>[Modify Form Page (3.6.3)]</b></p> <ul style="list-style-type: none"> <li>•Edits a specific page of a form.</li> </ul> <p><b>[A page of a form (3.6.3.3)]</b></p> <ul style="list-style-type: none"> <li>•Edits a form element.</li> </ul>
 (Extd. edit)	<p><b>[Headline edit]</b></p> <ul style="list-style-type: none"> <li>•Changes the parameters of the scripts, as well as the scripts that are selected for frames that handle them (in extended mode only, appearance set in 2.3.4)</li> </ul> <p><b>[Menu item (3.5.3.1)]</b></p> <ul style="list-style-type: none"> <li>•Displays the output of the script related to this menu item.</li> </ul>
 (Style)	<p><b>[Headline edit]</b></p> <ul style="list-style-type: none"> <li>•Accesses the frame style settings (2.3.4)</li> </ul> <p><b>[A page of a form (3.6.3.3)]</b></p> <ul style="list-style-type: none"> <li>•Leads to a screen where the style of the form element (checkbox, listbox,...) is selected (3.6.3.2).</li> </ul>
 (Hide/Display)	<p><b>[Headline edit]</b></p> <ul style="list-style-type: none"> <li>•Hides/shows a frame (appearance set in 2.3.4)</li> </ul> <p><b>[Menu item (3.5.3.1)]</b></p> <ul style="list-style-type: none"> <li>•Hides/shows a menu item</li> </ul>
 (Add elt.)	<p><b>[Headline edit]</b></p> <ul style="list-style-type: none"> <li>•For list frames, adds a single list element downwards.</li> </ul>
 (Append)	<p><b>[Menu item (3.5.3.1)]</b></p> <ul style="list-style-type: none"> <li>•Appends a child element for a menu item (3.5.4)</li> </ul>

Table 6.7: Blue Chameleon Content Management icons (continued)

Icon	[Context] and Action
 ,  (Add elt up, down)	<b>[Various]</b> Respectively allow to add an element before (if not the first) or after (if not the last) current element. Occur for : <ul style="list-style-type: none"> <li>•list elements ;</li> <li>•form pages ;</li> <li>•form elements ;</li> <li>•menu/submenu elements.</li> </ul>
 ,  (Displace elt up, down)	<b>[Various]</b> Allow to place element before the previous one (if not on the top) or after the next one (if not on the bottom). Occur in the same places as  ,  except form pages.
 (Remove)	<b>[Various]</b> Deletes an element. Occurs in the same context as for  ,  .

## 6.4 Form page elements : detailed

The following shows how the various form page elements (3.6.3.3) such as checkboxes, listboxes,... are composed. Whatever the kind of element, this is always achieved via the  icon.

### 6.4.1 "Active" elements


The following deals with elements on which, on the final form, actions such as checking, filling, selecting... will be done.

#### 6.4.1.1 Checkboxes

Fig.6.1 is how the checkbox choice as seen in "Form preview" (3.6.3.5) has been composed.

There, apart from the label that will appear on the final form :

- the range of values are given in a `<val>=<label>` manner ; the chosen `<val>` is the value that will be assessed to the CGI variable as given by 'Name', here `element20` ; a default value (checked when form is accessed) can be chosen ;
- how the checkboxes will be disposed is chosen, either on a single line or one below another ;
- the visibility for the form element is selected, from 'hidden'/'visible'/'editable'.

[Form element]  :

<b>Name</b>	element20
<b>Type</b>	Checkbox
<b>Subtype</b>	
<b>Label</b>	<input type="text" value="Instrument played"/>
<b>Value</b>	<input type="text" value="0=NONE&lt;br/&gt;1=Guitar&lt;br/&gt;2=Bass&lt;br/&gt;3=Drums&lt;br/&gt;4=Kazoo&lt;br/&gt;5=Other"/>
<b>Default</b>	<input type="text" value="0"/>
<b>Disposition</b>	<input type="text" value="one choice per line"/>
<b>Visibility</b>	<input type="text" value="editable"/>

Syntax for 'Value' is the following :  
<val>=<label>

**val** : the value that will be returned by the CGI. It can be anything (integer, string) but cannot contain the equal sign (=).  
**label** : text displayed for the choice.  
Choices are separated by ENTER. Empty lines are ignored.

'Default' contains the value (val) of the element which should be used as default.

Figure 6.1: Composing a checkbox selection for a form.

#### 6.4.1.2 Radiobuttons

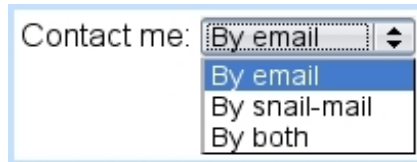
They are configured in the exact same way as for checkboxes, except that on the final form, of course one only item will be allowed to be selected.

#### 6.4.1.3 Listboxes

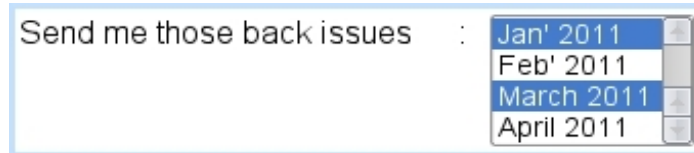
Listboxes, which provide three different styles, allows the same configuration method (<val>=<label>) and choices as for checkboxes, except there is no disposition choice.

The three styles are :

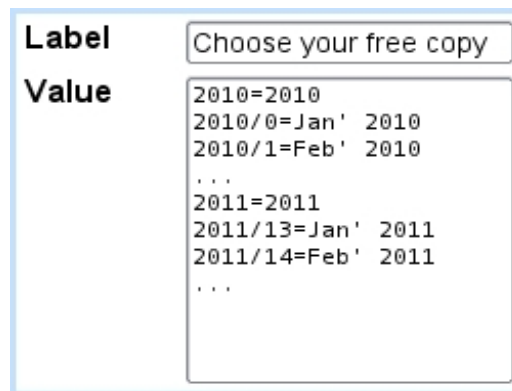
- 'single selection' : a standard drop-down menu, from which only one item can be chosen :



- 'multiple selection' : several items can be chosen, by clicking and holding **Ctrl** :



- 'chained selection' : two drop-down menus will be available, with the choices of the second (secondary list) depending on what was chosen for the first (main list). Main list elements are labelled `<valA>=<labelA>`, while secondary list elements `<valA>/<valB>=<labelB>` if this element B belongs to main list element A. For instance :



As a result, if for example '2011' is chosen in the first menu, only the corresponding choices for the secondary list (Jan' 2011, Feb' 2011,...) will be available :



#### 6.4.1.4 Input field and text area

The only configuring possibility for these two are :

- choosing whether input field will be 'normal' or 'password' (in that case hiding the typed content) ;
- choosing the *width x height* of the text areas, amongst various values (given in *number of characters x number of lines*).

#### 6.4.1.5 Form label - 'with check box'

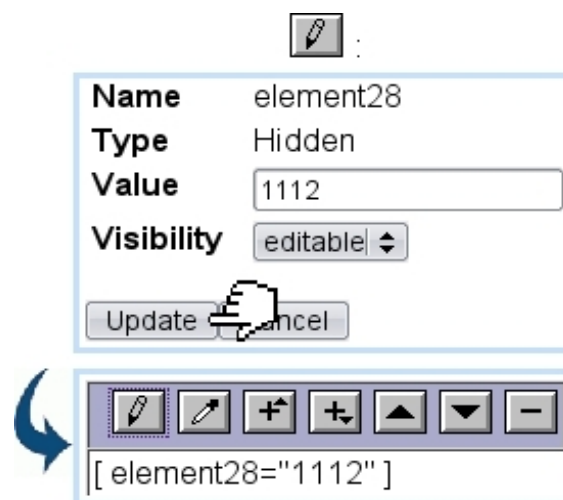
This choice for 'Label' allows to display a read-only text with a single checkbox before it.

### 6.4.2 "Passive" elements

These elements consist on graphical elements that will not be acted on the front-end form.

#### 6.4.2.1 Hidden field

A hidden field form object is simply assessed a value, which is then remembered during form edit :



#### 6.4.2.2 Form image

The Image :  *single image* choice for a form element allows to include an image into the form. Image selection is then similar to anywhere else in the Publisher :



#### 6.4.2.3 Form label - 'normal'

This choice for 'Label' type allows to display just a read-only text on the final version of the form.



#### **6.4.2.4 Form separator**

A form separator input is a centered horizontal line (corresponding to HTML's `<HR>`), which width is chosen amongst a range of percentages relative to page width.