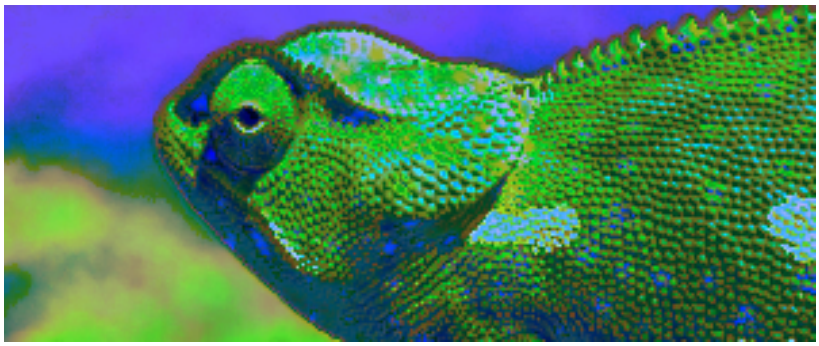


Blue Chameleon CMS
(Annex)

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

Blue Chameleon Content Management System full documentation.

May 8, 2012

Contents

List of Figures

List of Tables

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	`_Text;City;1;`	City :
2	Text area	`_Comment;Comments;2;`	Comments : WYSIWYG Text Editor
3	Image	`_Pic;Photo;3;`	Photo : <No image selected> Choose image No image Edit image map links
4	Link	`_Link;Link;4;`	Link : <No link selected> Choose link No link
5	Script (view)	`_Skr;Script;5;`	Script : ---------- paramètres:
6	Shared varfile	`_Varf;File;6;`	File : not selected Change varfile
7	Table	see 6.1.1	Table : internal format (;) Tabs Spaces
8	Script (edit)	Deprecated	Deprecated
9	Select	see 6.1.2	Country : Canada
10	Download	`_Dwnl;Download;10;`	Download : No download has been selected. Change download No download

Table 6.2: Leaf frame variables (continued)

11	Anchor	`_Ank;Anchor;11;`	Anchor :
12	Menu	`_Menu;Menu;12;`	Menu : No menu selected Change menu No menu
13	`SQL` select	see 6.1.3	Festival : Mittelalter und Goth Fest
14	Package object	`PO1;Main Artist;14;`	Main Artist : **Josh ``Lizard`` Jones** Choose Object No Object
15	Attribute	`POStyle2;Display style 2;15;`	Display style 2 : **Small showcase left** Choose Attribute 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 :

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`	`«INCLUDE design.phs»` Enables the Preview and Done! buttons to appear when a headline created from this model is edited. It is therefore a **mandatory** script.
`frame.phs`	`«INCLUDE frame.phs;_ParentId=«_FSId»;_FrameIndex={1,2,...}»` 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.
`metatags.phs`	`«INCLUDE metatags.phs»` Placed in the `<HEAD>...</HEAD>` section of the headline model's contents, inserts into the same section of the rendered `.htm` page the metatags that were filled on headline's **Manage** page.
`ListItems.phs`	`«INCLUDE ListItems.phs;_ListFrameItems="_ItemCount"»` In the case of a List model, puts into the variable `_ItemCount` (that has to be initialized before call) the number of list items that the list currently holds. It is then used to call `frame.phs` this very number of times.
`form.phs`	`«INCLUDE form.phs»` Will enable the headlines generated from this model to use forms.

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	«INCLUDE image.phs;pImage="_Pict";Html="border='0'..."» Loads the image as referred to by the variable _Pict ; the Html="..." string can be used to give any HTML additional parameter. As a result, the following tag will be generated : <IMG WIDTH=.. HEIGHT=.. ALT="..." SRC="..." *additional parameters*> «INCLUDE image.phs;pImage="_Pict";_URLOnly=1;_Return="_Url"»» This calls will store in variable _Url the address of the image.
href.phs	«INCLUDE href.phs;_Link="_Link";*Parameters*» Inserts a link as referred to by variable _Link, in different manners as specified by *Parameters* : •StartTag : generates the tag <A HREF="..."> if any link has been selected by the user ; •EndTag : generates the closing tag </A> if necessary ; •NoText : the generated <A HREF="..."> tag is trimmed so that it does not contain any redundant blanks or lines ; •Target : will add the specified target to the <A HREF="..." TARGET="Target"> tag ; •JustUrl : (as a unique parameter) returns only the address of the link ; •_Return=_ReturnVAR : makes href.phs not give out HTML tags, but specific information as stored in a _ReturnVAR variable that must be initialized before call. The various _ReturnVARs specify the information : ◇ _ReturnAddress contains the URL of the link ◇ _ReturnType contains the type of the link. Possible values are article, headline, URL, download, anchor and book ◇ _ReturnStyle contains the name of the file containing the link style ◇ _ReturnStyleId contains the Id of the link style. •AddDomain : forces the coding of the domain specified for the publisher in the URL ; •_OPSLink : 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.
scriptfile.phs	«INCLUDE scriptfile.phs» Enables the leaf frame to load scripts. If the published page is meant to be used dynamically, the commands «VAR NEW _Dynamic» «_Dynamic="yes"» must be placed before the call.
varphile.phs	«INCLUDE varfile.phs;_Action={1,2};_File="_MyFile"» Loads (_Action=1) or unloads (_Action=2) the variables defined in the varfile as referred to by _MyFile. Between these two commands, the variables defined by the selected varfile can be used as if they were variables defined for the current leaf.

Table 6.5: System scripts for leaves (continued)

table.phs	«INCLUDE table.phs;_Table="_Tbl";_Align="*Align*"» Inserts a table as referred to by variable _Tbl with the Align parameter being any of the alignments used for TD tags (left, right...)
download.phs	«INCLUDE download.phs;_Download="_Dwnl"» Generates the <A HREF="..."> , </A> tags around the download description as relevant to the _Dwnl variable.
anchor.phs	«INCLUDE anchor.phs;_Anchor="_Ank"» Generates the <A NAME="..."> tag for the anchor as referred to by variable _Ank.
showmenu.phs	«INCLUDE showmenu.phs;_Menu=_Menu» Loads the menu as referred to by the variable _Menu.
PackageObjectView.phs	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)	**[Headline edit]** •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.
 (Edit)	**[Headline edit]** •Edits a leaf frame, i.e. leads to a screen where its variables (text areas,...) are filled. **[Menu item (3.5.3.1)]** •Edits the menu item's text and link. **[Form model (3.6.1.1)]** •Edits that paramater of a form model. **[*Modify Form Page* (3.6.3)]** •Edits a specific page of a form. **[A page of a form (3.6.3.3)]** •Edits a form element.
 (Extd. edit)	**[Headline edit]** •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) **[Menu item (3.5.3.1)]** •Displays the output of the script related to this menu item.
 (Style)	**[Headline edit]** •Accesses the frame style settings (2.3.4) **[A page of a form (3.6.3.3)]** •Leads to a screen where the style of the form element (checkbox, listbox,...) is selected (3.6.3.2).
 (Hide/Display)	**[Headline edit]** •Hides/shows a frame (appearance set in 2.3.4) **[Menu item (3.5.3.1)]** •Hides/shows a menu item
 (Add elt.)	**[Headline edit]** •For list frames, adds a single list element downwards.
 (Append)	**[Menu item (3.5.3.1)]** •Appends a child element for a menu item (3.5.4)

Table 6.7: Blue Chameleon Content Management icons (continued)

Icon	[Context] and Action
 ,  (Add elt up, down)	**[Various]** Respectively allow to add an element before (if not the first) or after (if not the last) current element. Occur for : •list elements ; •form pages ; •form elements ; •menu/submenu elements.
 ,  (Displace elt up, down)	**[Various]** 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)	**[Various]** 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'.

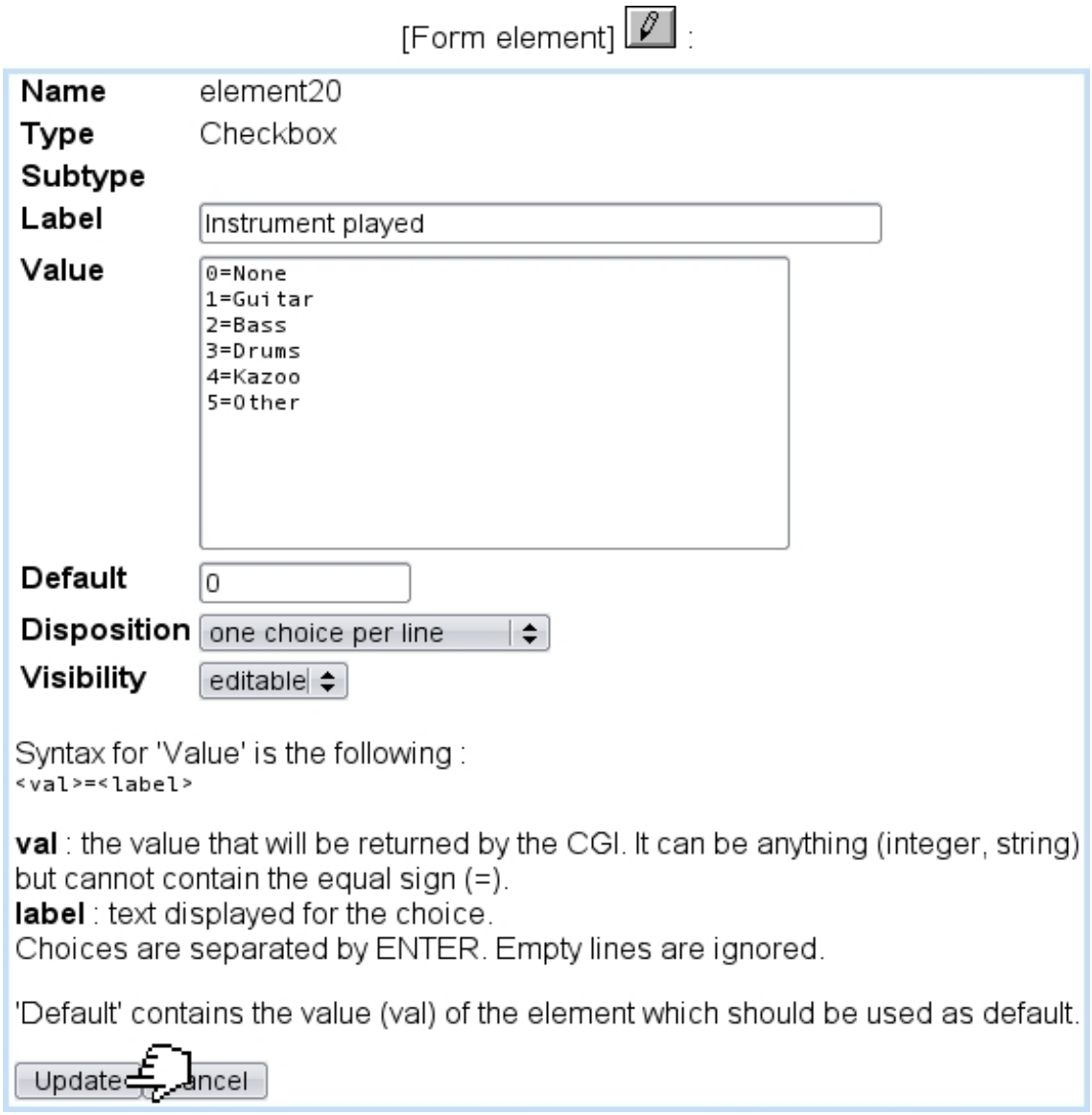


Figure 6.1: Composing a checkbox selection for a form.

6.4.1.2 Radiobuttons

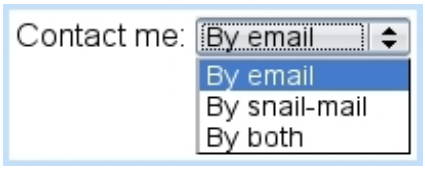
They are configured in the exact same way as for checkboxes, except that on the final form, of course one only item will be allowed to be selected.

6.4.1.3 Listboxes

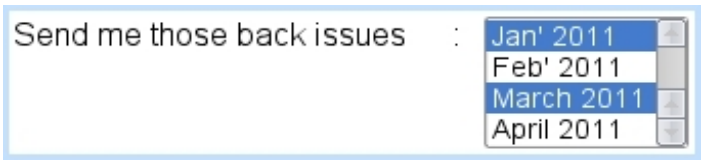
Listboxes, which provide three different styles, allows the same configuration method (`<val>=<label>`) and choices as for checkboxes, except there is no disposition choice.

The three styles are :

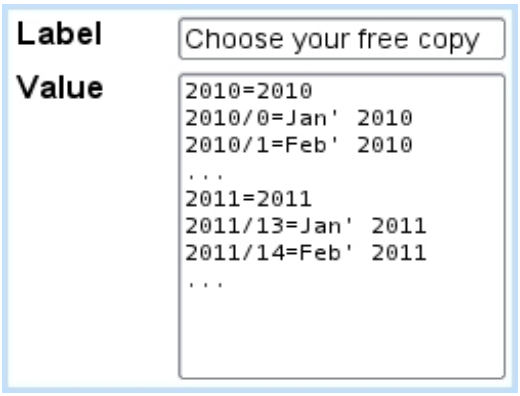
- 'single selection' : a standard drop-down menu, from which only one item can be chosen :



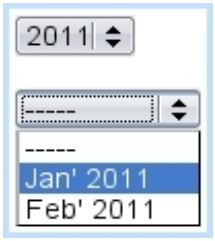
- 'multiple selection' : several items can be chosen, by clicking and holding `Ctrl` :



- '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'

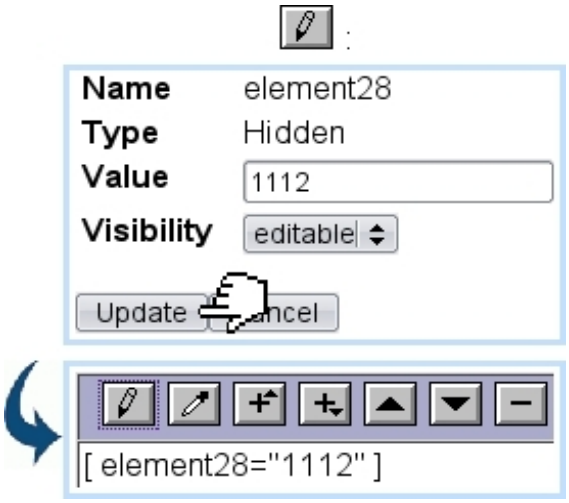
This choice for 'Label' allows to display a read-only text with a single checkbox before it.

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.