AJUI_Breadcrumb

User Manual

Version Control	Date	Comment (Change)	Author
1.0	04.11.2019	First version	Gary Criblez
1.0.1	08.11.2019	Section Corner Radius	Gary Criblez
1.1	09.11.2019	Using the placeholder "#" with the PicturePath property in a section	Gary Criblez
1.2	17.01.2020	Added model "groupedButtons"	Gary Criblez

1 Introduction

1.1 What is AJUI_Breadcrumb ?

AJUI_Breadcrumb is a component developed in the 4D language. Its use is intended for 4D developers. It allows you to dynamically generate and display Breadcrumbs in the context of a form.

What is a Breadcrumb?

Breadcrumb refers to a navigation element mainly used on the web. It is used to indicate to the user at which hierarchical level the user is positioned.

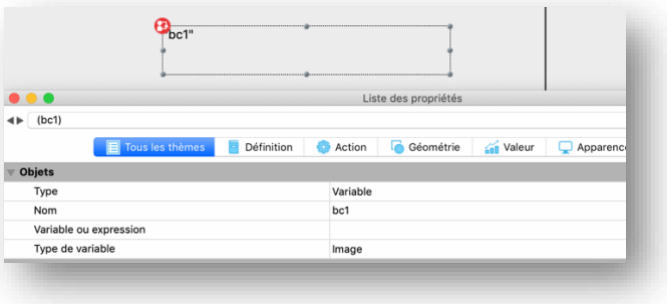
As part of a business application created with 4D, Breadcrump will allow you to visualize all the steps of a process, to know the current step and to return to one of the previous steps of a processing or entry process.

To do this, the Breadcrumb is composed of sections that will allow you to navigate between different pages of a form or display specific subforms according to the hierarchy of the application's business logic.

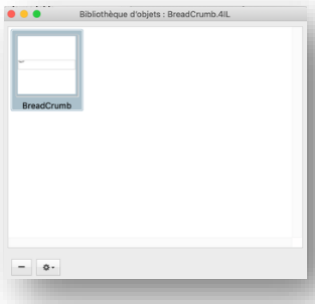
The component **AJUI_Breadcrumb** is defined using an object variable representing an instance of AJUI_BreadCrumb. This variable contains a set of properties and formulas (member functions) to define the structure and content of the Breadcrumb and to generate it graphically in a 4D form.

1.2 Composition of a Breadcrumb

Basically, the Breadcrumb generated by AJUI_BreadCrumb is an image type form object variable that you can create yourself in the context of a form.



Nevertheless, we provide you with a predefined template of this form object variable containing all the appropriate properties for creating a BreadCrumb. This is provided in an object library file "BreadCrumb.4dlibrary".



1.3 Breadcrumb structure and model

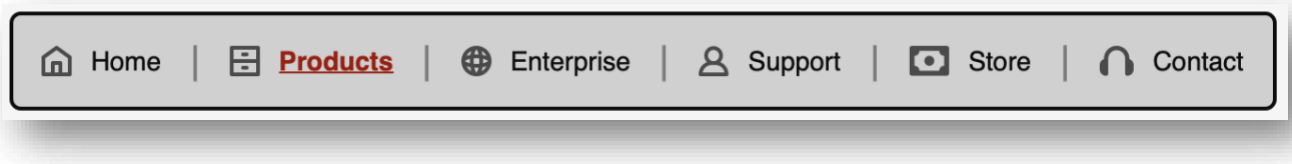
A Breadcrumb generated by the component is composed of a "*container*" that can be sized manually or dynamically. The container will contain sections. You can define the number of sections you want.

A section is associated with a "template" and can contain text elements, an image and a callback. It also has an identifier (id) that makes it unique and allows it to be identified when interacting with the Breadcrumb.

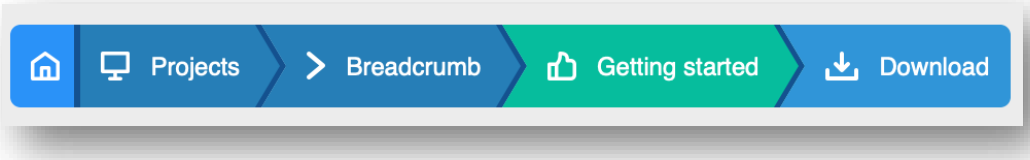
AJUI_Breadcrumb offers two main models of Breadcrumb.

- The **simple** model
- The **arrow** model
- The **groupedButtons** model

The **simple** model is a sequence of *sections*, in a single *container*, separated by a text-based *divider* (separator), i. e. a string of any character (e. g. ">", "/", "|", etc.)



The **arrow** model is different. Each section has its own *container* whose front and rear ends can be represented by an arrow shape or a vertical line. A *divider* (separator) is used to define the size of the line of each arrow.



The groupedButtons template is a bit particular because it is not exactly a new model. It is based on the arrow template using predefined properties to keep a design specific to that model.

When you use this model, the sections can be either "standard" or "current". The "current" type designates the selected section and there can only be one at a time. In other words, the model works like radio buttons.

*Small precision on the identifiers (id) of sections. In the **simple** model, the id allows the mouse to interact with the text and image of the section. For the **arrow** model, the id applies to the entire graphic area of the container in the section.*

1.3.1 Section - types and states

There are five types of sections:

- **Standard**
- **First**
- **Current**
- **Previous**
- **Next**

For each section type are associated four states:

- **default**
  This is the status of a section when a user does not interact with it and when it is not disabled.
- **hover**
  This state occurs when the "On Mouse Enter" or "On Mouse Move" event (found section id) is triggered. In concrete terms, this corresponds to hovering over the section with the mouse. It ends with the event "On Mouse Leave" or "On Mouse Move" (id of different section or none).
- **active**
  This state intervenes on the "On Clicked" event and ends with the "On Mouse Up. Note that it has priority over the "hover" state. In concrete terms, the user clicks on a section and the status stays until it is released.
- **disable**
  This is the state of a section if the "disable" property is true. When the section is "disable", only this state is taken into account to the exclusion of all others (hover and active).

A section can take on a different appearance and display different content depending on its type and status.

All states except "default" state inherit all the properties of the latter (default). Each property can be assigned exceptions.

1.3.2 Events and callbacks

The Breadcrumb requires that several events be activated on the image form object in order to be able to manage the different states. Here is the list of these:

- On Load
- On Clicked
- On Double Clicked (optional)
- On Mouse Enter
- On Mouse Leave
- On Mouse Up
- On Mouse Move

For both events (On Clicked, On Double Clicked) it is possible to associate callbacks to them using the member functions.

Warning: All methods used as callbacks must be shared (method properties) with the component so it can call it. To prevent an error from occurring in such case, the component will check if the method is well shared and if it is not, it will share it itself when executing a callback. In addition, the component will propose to create the callback method if it does not exist when using **Setters** for callbacks and also before executing a callback.

1.3.3 Exception and constant management

The component uses an exception system to distinguish which value of the state-related properties it should use when generating the Breadcrumb.

It should be noted that when creating an instance, the component assigns default values to the "**standard**" type properties and for the "**default**" state. So, when you are going to use member functions related to states (see subchapter 3.9: Typical section), you will have to pass as the first parameter, a constant that will designate the type and state for which you want to receive or pass a value (Getter/Setter).

Example :

```
  // Setter
$bc.SectionBgColor(AJUI_bc_standard_default;$color)
  //Getter
$colorSection:=$bc.SectionBgColor(AJUI_bc_standard_default)
```

How the component handles exceptions:

To explain this, we will take a concrete example. I have a "first" section and we want to define its "hover" state.

The component will therefore first check if there are any exceptions in the "*myInstance.breadCrumb.sectionTypes.first.hover*" object. If he finds any, he uses these values. If it does not find exceptions for all or some of the properties, it will control the exceptions for "*myInstance.breadCrumb.sectionTypes. first.default*". After that, it will check "*myInstance.breadCrumb.sectionTypes. standard.hover*" and finally "*myInstance.breadCrumb.sectionTypes.standard.default*" if there are any properties left to define.

It will not go any further, because the "standard-default" values are not exceptions, but the basic values.

	First	First	First	Standard	Standard
	hover (final values)	**hover**	**default**	**hover**	**default**
bgColor	#CECECE		#CECECE		#D4E3FE
borderColor	Red	Red			Black
borderSize	2			2	0
fontColor	White	White	Blue		#0148AA
fontName	Arial				Arial
fontSize	12				12
fontStyle	bold	bold			None
pictureOpacity	80	80			100
replacingColor	none				none

List of constants :

- *AJUI_bc_standard_default*
- *AJUI_bc_standard_hover*
- *AJUI_bc_standard_active*
- *AJUI_bc_standard_disable*

- *AJUI_bc_current_default*
- *AJUI_bc_current_hover*
- *AJUI_bc_current_active*
- *AJUI_bc_current_disable*

- *AJUI_bc_first_first_default*
- *AJUI_bc_first_hover*
- *AJUI_bc_first_active*
- *AJUI_bc_first_disable*

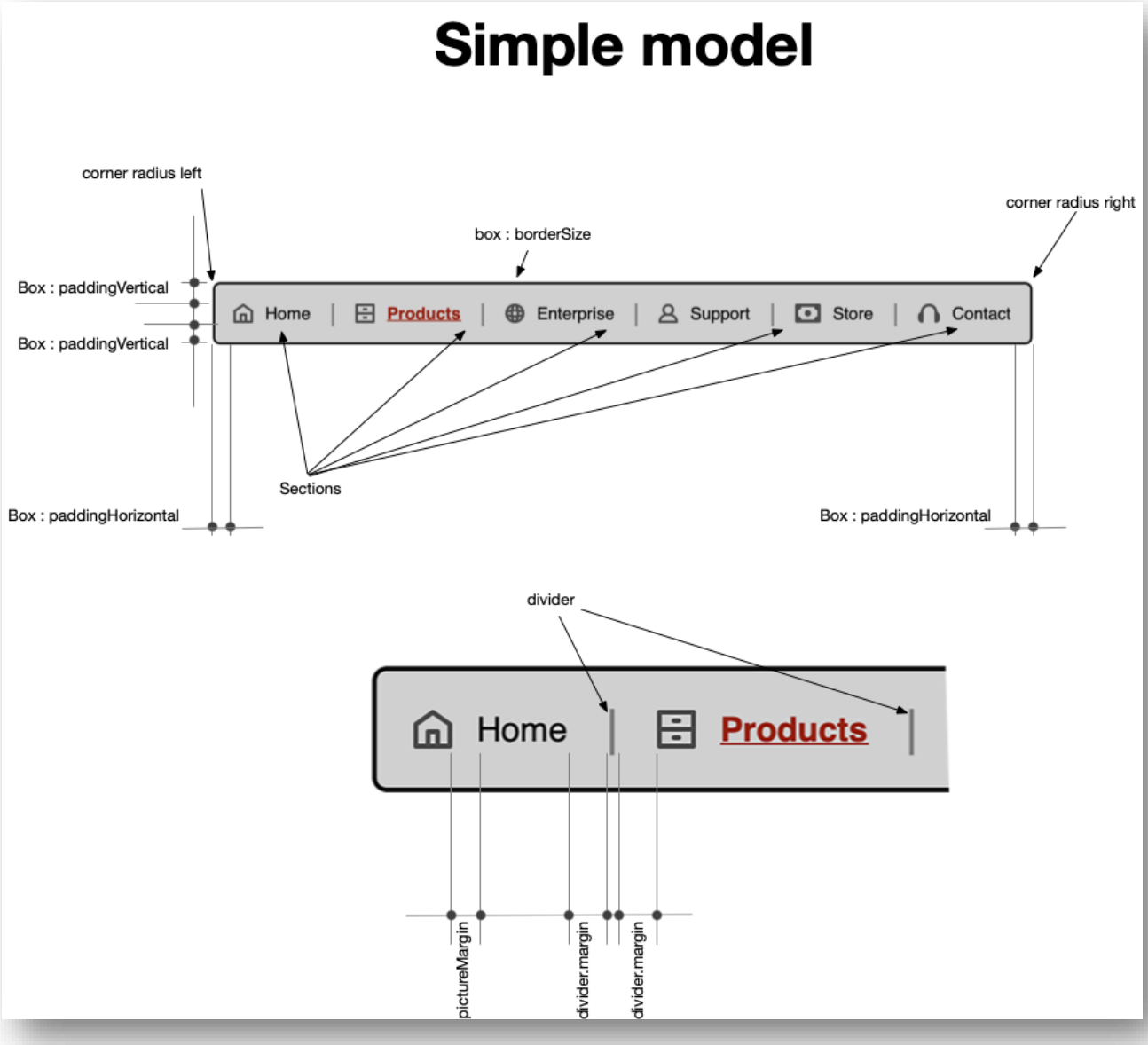
- *AJUI_bc_previous_default*
- *AJUI_bc_previous_hover*
- *AJUI_bc_previous_active*
- *AJUI_bc_previous_disable*

1.4 Technical overview

In this section, we provide you with two diagrams describing how the models are structured in terms of the graphic construction of the container and sections.

1.4.1 Simple Breadcrumb

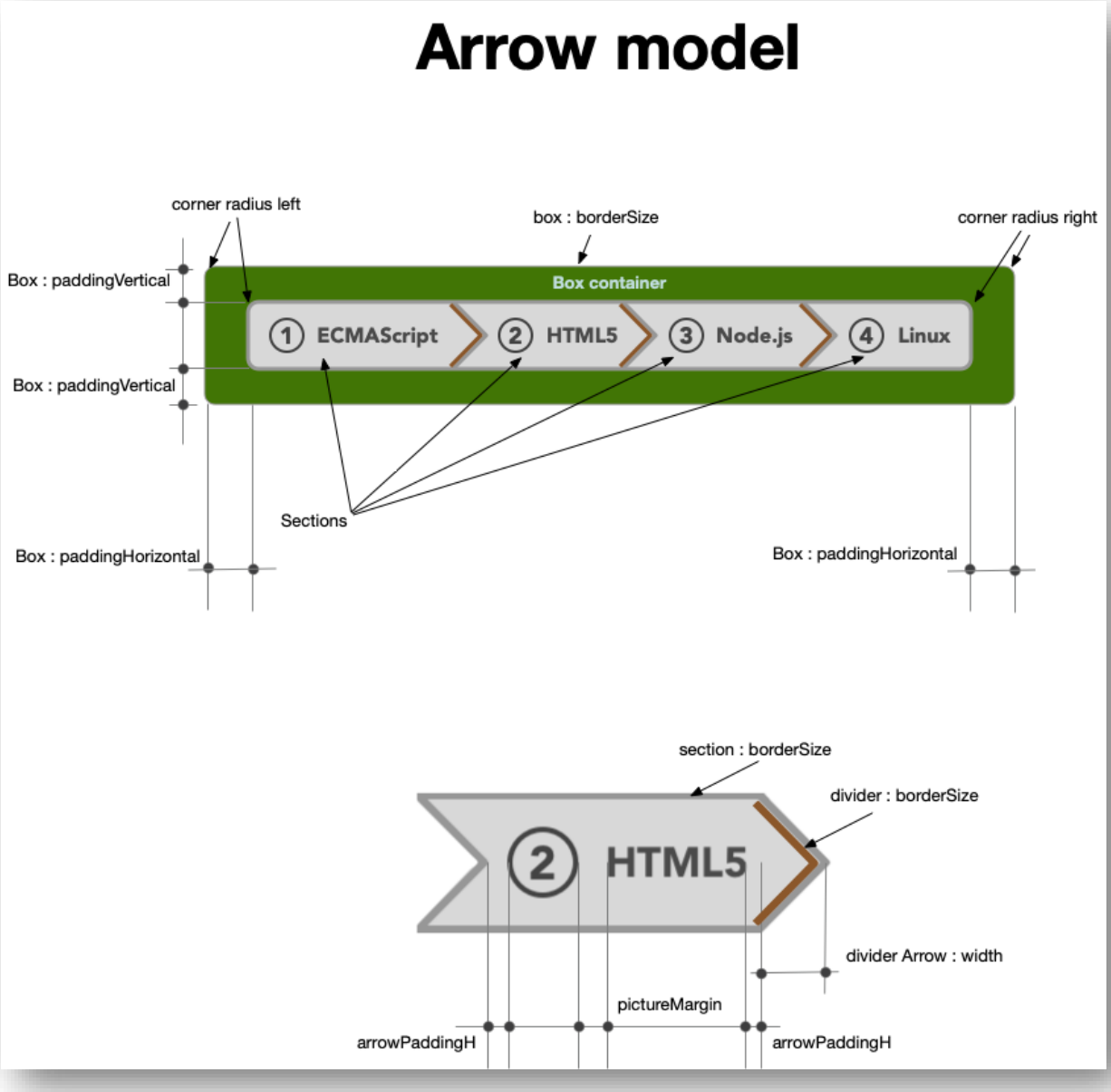
The "simple" model is a sequence of sections containing text and/or an image separated by a divider and a margin that applies to each side of the divider.



1.4.2 Breadcrumb arrow

The "arrow" model is a sequence of sections whose right part is finished with an arrow and whose size can be defined. The rear part of the section reproduces the shape of the previous section. It is possible to define a divider (line) that will be placed inside the front part of the section and will reproduce the shape of the arrow. It is possible to separate the sections via a margin.

You can define a border size for the sections that will be placed inside them. The first and last sections are assigned the defined values of the corner radius of the container, applying a proportional factor equal to the difference in height between the sections and the container, in order to maintain and guarantee a perfect parallelism between the sections and the container.The first and/or last section can be defined with or without an arrow.



1.4.3 Breadcrumb groupedButtons

The "groupedButtons" model has the same structure as the "arrow" model. It differs from the latter in its behavior when clicking on a section. Thus when a click is made, it changes its state to "current" and sets the other sections to the "standard" state. It thus behaves like a radio button. In addition, several properties are predefined in order to keep a logic in the design of this model (for example, the sections do not end with an arrow).

List of properties and their predefined values :

- dynamicSize (box) : True
- bgColor (box) : value of the color of the border of the box
- margin (divider) : value of the size of the border of the box
- padding horizontal (box) : 0
- padding vertical (box) : 0
- width (arrow) : 0%
- borderSize (arrow) : 0
- lastSectionWithArrow (arrow) : False)
- FirstSectionWithArrow (arrow) : False
- borderSize (section) : 0

For more information on these, see the chapter on the list of properties.

In addition, all the sections are set to "standard" type except the "current" type which corresponds to the section for which the identifier is stored in the instance (id of the last selected section).

All the values you could have set for these properties will be overridden by the predefined ones.

1.5 Life cycle of a Breadcrumb

1.5.1 Phase 1: Initialization and configuration

The first step is to define the properties of the Breadcrumb that will be displayed. To do this, the component offers a "New AJUI_BreadCrumb" method generating a base object representing a default definition. Then the user is free to use the member functions attached to the object to customize its properties as he wishes and add the number of sections he wants.

1.5.2 Phase 2: Calculations and generation of the breadcrumb

The second phase that is the most significant for the component is the creation of the Breadcrumb based on its definition. During this phase, he will be in charge of retrieving the object's properties in order to perform a series of calculations and checks in order to generate/assign the various elements that will make up the Breadcrumb. It is also in this phase that exceptions related to the report and section type are processed. If a callback associated with an event (On Clicked, On Double clicked) is assigned to a section and the event is detected, then it will be executed at the end of the generation.

1.5.3 Phase 3: Show and hide

At the end of the generation, the image is associated with the image form object associated with it. Breadcrumb display management can be done using the "Show" and "Hide" member functions or using 4D methods to manage form objects.

2 Component methods

`New AJUI_BreadCrumb ( { template_param } ) -> breadcrump`

This method returns an object variable (breadcrump) that represents an instance of AJUI_BreadCrumb. It contains all the properties and their default values as well as the formulas (member functions) to manipulate them.

It is possible to pass an object (template_param) as a parameter to it in order to import an existing template (JSON file). Object Properties expected for the object (template_param) used as parameter:

- **templateName** : Represents the name of the JSON file to import (template). If the file is not found, the method will return a new instance of AJUI_Breadcrumb.
- **templatePath** (optional): You can specify a path to recover the file otherwise, the component will search in the default folder located in the host database resources (.../Resources/AJUI_BreadCrumb_Templates/).

```
C_OBJECT($template_obj;$bc)
$templateName:="MybreadcrumbTemplate.json"
$templatePath:=Get 4D folder(Current resources folder)
$templatePath:=$templatePath+"my_Templates_Folder"+Folder separator
$template_obj:=New object()
$template_obj.templateName:=$templateName
$template_obj.templatePath:=$templatePath
$bc:=New AJUI_BreadCrumb ($template_obj)
```

`AJUI_BC_info -> version`

This method returns a character string representing the version number of the component.

Version is a string.

`AJUI_BC_BC_loadTemplates ( { folder_path} ) -> templates_col`

Utility method to retrieve in a collection (templates_col), all templates JSON files inside a folder. You can pass in parameter (folder_path) the path of the folder containing the templates. If this is not the case, the method will search in default path (.../Resources/AJUI_BreadCrumb_Templates/).

3 List of properties

A BreadCrumb has a set of properties that will define its representation and interactions with the host form.

In this chapter, we will review the different existing properties that can be accessed by a formula that acts as a **Setter** or **Getter**[1].

All formulas can be called up at the first level of the object.

For color properties, you can give them a color code as a CSS string (Example: "blue" or "#0000FF" or "rgb(0,0,255)")

```
$color:="blue:80"
$color:="#0000FF:80"  //or #FF
$color:="rgb(0,0,255):80"
```

By default the opacity factor applied to the colors is 100%. To change this value to 80%, simply add this value to the string defining the color by separating it with the colon character". (Example: "blue:80)

It is recommended not to manually change properties whose values are not defined by formulas, as they are managed internally by the component. Properties related to sections will be dealt with in the section management chapter.

3.1 Global

Global properties at the second level of the object: **MyBC.** breadCrumb.global

Name	Type	Description	By default	Formula
name	string	Name of the Breadcrumb and therefore of the image form object.	Empty string	**Name**
model	string	Model to use: simple, arrow, groupedButtons	arrow	**Model**
rtl	Boolean	The sections are generated from right to left	False	**RTL**
maxSectionToDraw[2]	longint	Maximum number of sections to be generated	-1	**MaxSectionToDraw**

Global - Formulas and their parameters

Formula name	Parameter(s)
Name (nom)	- Name of the form object (string)
Model (**modèle**)	- Model name (string)
RTL (activate)	- Section layout (Boolean)

[1] in the case that no parameters are passed
[2] maximum number of sections to be displayed. No limit corresponds to -1 which is the default value. It is possible to display fewer sections than those created.

MaxSectionToDraw (max)	- Maximum number of sections to display (longint)

3.2 Boxes

Properties related to the container on the second level of the object: **MyBC.** breadCrumb.box

Name	Type	Description	By default	Formula
bgColor	text	Background color. It is possible to define an opacity rate (%) (Ex: black:80 - black with an opacity of 80%)	#F0EEF1	**BgColor**
centered	Boolean	Allows you to center the Breadcrumb in relation to the position and size of the image object defined in the form editor. Applies only if dynamicSize is enabled.	False	**IsCentered**
dynamicSize	Boolean	Sizes the width of the Breadcrumb automatically according to the size of the sections.	True	**IsDynamicSize**
height	logint	Height of the Breadcrumb. If -1, it takes the value defined in the form editor as the height.	-1	**BoxHeight**
width	logint	Width of the Breadcrumb. If -1, it takes the width defined in the form editor as the width.	-1	**BoxWidth**

Box - Formulas and their parameters

Formula name	Parameter(s)
BgColor (color)	- Background color (string)
IsCentered (activate)	- Automatic centering (Boolean)
IsDynamicSize (activate)	- Automatic sizing (Boolean)
BoxHeight (height)	- Container height (longint)
BoxWidth (width)	- Container width (longint)

3.3 Box corner radius

Properties related to the corners of the container on the third level of the object: **MyBC.** breadCrumb.box.cornerRadius

Name	Type	Description	By default	Formula
left	logint	Value of the radius applied to the corners on the left side.	5	**CornerRadiusLeft**
right	logint	Value of the radius applied to the corners on the right side.	5	**CornerRadiusRight**

Box corner radius - Formulas and their parameters

Formula name	Parameter(s)
CornerRadiusLeft (radius)	- Value of the radius on the left side of the container (longint)
CornerRadiusRight (radius)	- Value of the radius on the right side of the container (longint)

3.4 Box border

Properties related to the container border on the third level of the object: **MyBC.** breadCrumb.box.border

Name	Type	Description	By default	Formula
color	text	Color of the container border. It is possible to define an opacity rate (%) (Ex: black:80 - black with an opacity of 80%)	black	**BorderColor**
size	logint	Size of the container border.	2	**BorderSize**

Box border - Formulas and their parameters

Formula name	Parameter(s)
BorderColor (color)	- Border color (string)
BorderSize (size)	- Border size (longint)

3.5 Box padding

Properties related to the paddings of the container on the third level of the object: **MyBC.** breadCrumb.box.padding

Name	Type	Description	By default	Formula
horizontal	logint	Horizontal padding.	0	**PaddingH**
vertical	logint	Vertical padding.	0	**PaddingV**

Box padding - Formulas and their parameters

Formula name	Parameter(s)
PaddingH (padding)	- Horizontal padding (longint)
PaddingV (padding)	- Vertical padding (longint)

3.6 Divider

Properties of the separator on the second level of the object: **MyBC.** breadCrumb. divider

Name	Type	Description	By default	Formula
margin	logint	Divider margin.	0	**DividerMargin**
text	text	Character(s) to be used as separator for the simple model.	>	**Divider**

Divider - Formulas and their parameters

Formula name	Parameter(s)
DividerMargin (margin)	- Divider margin (longint)
Divider (text)	- Separator (string)

3.7 Divider font

Separator properties for the simple model on the third level of the object: **MyBC.** breadCrumb. divider.font

Name	Type	Description	By default	Formula
color	text	Color of the separator. It is possible to define an opacity rate (%) (Ex: black:80 - black with an opacity of 80%)	blue	**DividerFontColor**
name	text	Name of the font to use. Several fonts can be defined by separating them with a comma without spaces. The component will search for the first available font on your current OS. Be careful, some fonts lower some characters and therefore these cannot be vertically centered at the Breadcrumb level.	Arial	**DividerFontName**
size	logint	Size of the separator.	24	**DividerFontSize**
style	text	Style to be applied to the separator. Available: bold, italic, underline, strikethrough. It is possible to use several styles separated by a comma without spaces.	none	**DividerFontStyle**

Divider font - Formulas and their parameters

Formula name	Parameter(s)
DividerFontColor (color)	- Font color (string)
DividerFontName (policy)	- Name of the font (string)
DividerFontSize (size)	- Font size (longint)
DividerFontStyle (style)	- Font style (string)

3.8 Divider arrow

Properties for the arrow model located at the third level of the object: **MyBC.** breadCrumb. divider.arrow

Name	Type	Description	By default	Formula
borderSize	logint	Arrow thickness.	4	**ArrowBorderSize**
color	text	Arrow color. It is possible to define an opacity rate (%) (Ex: black:80 - black with an opacity of 80%)	#0048AA	**ArrowColor**
firstSectionWithArrow	Boolean	If false, the width of the arrow is zero for the first section.	True	**IsFirstSectionWithArrow**
lastSectionWithArrow	Boolean	If false, the width of the arrow is zero for the last section.	False	**IsLastSectionWithArrow**
paddingH	logint	Padding apply to the content of the section in an arrow template	16	**ArrowPaddingH**
width	text	Arrow width.	50%	**ArrowWidth**

Divider arrow - Formulas and their parameters

Formula name	Parameter(s)
ArrowBorderSize (thickness)	- Arrow thickness (long)
ArrowColor (color)	- Arrow color (string)
IsFirstSectionWithArrow (activate)	- Arrow to first section (Boolean)
IsLastSectionWithArrow (activate)	- Arrow to the last section (Boolean)
ArrowPaddingH (padding)	- Section padding (longint)
ArrowWidth (width)	- Arrow width (longint)

3.9 Section types

Properties related to states and section types at the fifth level of the object: **MyBC.** breadCrumb. sectionTypes[Type][State]

Name	Type	Description	By default	Formula
bgColor	text	Background color. It is possible to define an opacity rate (%) (Ex: black:80 - black with an opacity of 80%)	#D4E3FE	**SectionBgColor**
borderColor	text	Color of the border.	Black	**SectionBorderColor**
borderSize	logint	Border size.	0	**SectionBorderSize**
fontColor	text	Font color. It is possible to define an opacity rate (%) (Ex: black:80 - black with an opacity of 80%)	#0148AA	**SectionFontColor**
fontName	text	Name of the font.	Arial	**SectionFontName**
fontSize	logint	Font size.	12	**SectionFontSize**
fontStyle	text	Police style. Available: bold, italic, underline, strikethrough. It is possible to use several styles separated by a comma without spaces.	None	**SectionFontStyle**
pictureOpacity	logint	Percentage of opacity of the images.	100	**PictureOpacity**
replacingColor	text	Color to replace the current color in an SVG image.	Empty chain	**ReplacingColor**

Sample Section - Formulas and their parameters

Formula name	Parameter(s)
SectionBgColor (constant; color)	- Type and state (constant) - Background color (string)
SectionBorderColor (constant; color)	- Type and state (constant) - Border color (string)
SectionBorderSize (constant; size)	- Type and state (constant) - Edge size (longint)
SectionFontColor (constant; color)	- Type and state (constant) - Font color (string)
SectionFontName (constant; font)	- Type and state (constant) - Name of the font (string)
SectionFontSize (constant; size)	- Type and state (constant) - Font size (longint)
SectionFontStyle (constant; style)	- Type and state (constant) - Font style (string)
PictureOpacity (constant; percentage)	- Type and state (constant) - Opacity percentage (longint)
ReplacingColor (constant; color)	- Type and state (constant) - Replacement color (string)

4 Section management

In AJUI_BreadCrumb, a **section** is an object containing a set of properties affecting only that one. All these section objects are grouped together in a collection at the instance level.

4.1 Description and use of a section

A section is created and manipulated using an identifier (id) that allows the component to recognize it when the mouse interacts with it. A position attribute is associated with it in order to define the order in which it will be positioned and displayed in the breadcrumb.

Each section is also associated with a type (standard, first, previous, next, current). The type will define which state properties (e. g. AJUI_bc_standard_hover) the component should apply for this section.

Each section has a textual content (label) and/or a specific image, a callback and an object containing the parameters to be passed to the callback. A callback can be assigned to a section on "On Clicked" and/or "On DoubleClicked" events

Finally, a section has a (disable) property to change it from active to inactive.

In the next two sub-chapters, we will present the member functions available to manage the sections and also the properties associated with each section.

4.2 Member functions

The management of the sections is done via CRUD member functions as well as some utility methods to simplify their management. Here is this list:

```
AddSection ( id { ; param })
```

Allows you to add a section. It waits in first parameter for the id of the target section and you can optionally pass in second parameter an object containing some or all of the properties of a section with the values you want. Unspecified properties will receive their default values.

```
UpdateSection ( id { ; param }) -> { param }
```

Allows you to update a section. It waits in the first parameter for the ID of the target section and in the second optional parameter for an object containing the properties to be modified with their new values (Setter). If you do not pass the second parameter, the function returns the section definition to you in an object (Getter).

```
RemoveSection ( id)
```

Allows you to delete a section. It waits for the ID of the target section as a parameter.

```
SetCurrentSection ( id)
```

Utility function to assign the "current" type to a section. It will also apply the "previous" type to previous sections and the "next" type to subsequent sections. This processing will not apply to "first" sections. This function waits for the id of the target section in parameter.

```
GetNextPosition -> nextPosition
```

Returns the next position value (largest value plus one) in length.

```
RemoveAllSections
```

Removes all sections from the collection.

4.3 List of section properties

As you have seen in the member functions, all the properties below can be modified by passing them into an object in the CRUD member functions. If this is not the case, the default values will be applied (except for the id which is mandatory when creating a section)

Name	Type	Description	By default
id	text	Section id. It does not have a default value, as it **must** be filled in when using the CRUD functions.	-
position	logint	Position of the section in the Breadcrumb. By default, the component searches for the highest value of the positions and will add one.	Largest position value +1
type	text	The section is related to one of the following types: standard, first, current, previous and next.	standard
label	text	Text content of the section.	section + position value
disable	Boolean	If True, the section is considered in the disable state.	False
picturePath	text	Folder path from the "resources" folder of the host database. You can also use "#" plus the name of your picture file as a path. This will only work with an imported AJUI Button template and the picture must be located in the same folder as the template.	Empty chain
picturePosition	text	Position of the image in relation to the text. Possible value: left, right.	left
pictureMargin	logint	Margin between text and image.	10
pictureHeight	text	Size of the image in height, the number must be followed by "px" if you want to define a size in pixels otherwise pass a value in % if you want a percentage in relation to the available height of the section. The width is automatically resized according to the height calculated based on the font size.	40%
colorToReplace	text	Current color of an SVG. Here is the value of the "fill" tag. Can only be used with an image whose extension is SVG (see the replacingColorSVG property).	Empty chain
onClickCB	text	Name of the callback method on the event "On Clicked".	Empty chain
onClickCBParams	object	Object containing the callback parameters on the "On Clicked" event	null
onDoubleClick	text	Name of the callback method on the event "On Double Clicked".	Empty chain
onDoubleClickCBParams	object	Object containing the callback parameters on the "On Double Clicked" event	null

5 Member functions not related to a property

List of member functions that are not attached to a property.

`Draw ()`

Generates the BreadCrumb by taking into account the current status for each section.

`Export (templateName { ; templateFolderPath })`

Allows you to create a template of your AJUI_Breadcrumb object in a JSON format file. The formula first requires the name to be associated with the template (file name). A second optional parameter allows you to specify the folder path, otherwise it will be exported to the default folder (.../Resources/AJUI_BreadCrumb_Templates/).

`Hide ()`

Hide the Breadcrumb.

`ResetAllStates`

Allows you to delete all exceptions.

`ResetState ( state)`

Allows you to delete all the exceptions of a state. You must pass the constant corresponding to the state in parameter. (see chapter 1.3.3 about exceptions)

`RemovePropertyException ( property; state)`

Allows you to delete an exception related to a property. The property name is passed in the first parameter and the constant corresponding to the state in the second parameter. (see chapter 1.3.3 about exceptions)

`RemovePropertyExceptions ( property)`

Allows to delete all the exceptions related to a property for each of the states. The property name is expected in parameter. (see chapter 1.3.3 about exceptions)

`RemovePropertyTypeExceptions ( property ; typeSection )`

Allows you to delete all exceptions related to a property, but only for a defined type. The property name is expected in the first parameter and the type in the second parameter (available: standard | first | current | next | previous). (see chapter 1.3.3 about exceptions)

`Show ()`

Displays the Breadcrumb.

`RemoveCurrent ()`

When using a **groupedButtons** model, the instance keeps the id of the selected section (“current type”). This function allows to erase the id in memory.

Note that the **`Hide`** and **`Show`** member functions are actually wrappers of the 4D "OBJECT SET VISIBLE" method. You can use it directly if you wish.

Concerning the member functions allowing to remove one or more exceptions, it must be understood that they will not have repercussions for the properties linked to "standard-default", because these properties are the default values and are therefore not exceptions.

6 Getting Started

6.1 Installation

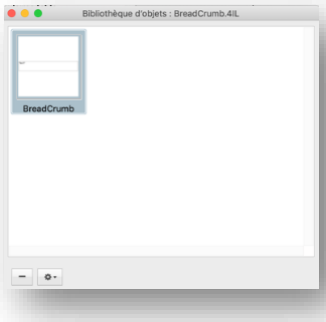
AJUI_BreadCrumb must be placed in the component folder of your application.

6.2 Principles of use

In this section, we describe the sequence of operations to be performed in order to generate a simple Breadcrumb in the context of the main form.

6.2.1 Prerequisites

AJUI_Breadcrumb needs an image object that will serve as a model. This object is provided with the "BreadCrumb.4IL" library that you can open from your application (/File/Open/Object Library...) and you just have to drag & drop the "BreadCrumb_object" object on your form.



You can duplicate this object to create as many Breadcrumb as necessary. Each Breadcrumb should have a specific name that you can define according to your imagination.

If you create the Breadcrumb by yourself in the form editor, do not forget to activate the events required to operate the Breadcrumb (see the chapter on events).

6.2.2 Basis for creating a breadcrumb

The first thing to do when creating a Breadcrumb is to open the method of your image form object and create an instance with the method "New AJUI_BreadCrumb". Then, associate the name of your form object to the instance with the "Name" member function. Then, add sections with the "AddSection" member function. Now, the Breadcrumb is created and will use the default values, all you have to do is adapt the parameters of it to your needs in terms of interface and operation.

```
Case of
: (Form event=On Load)

  Form.bc:=New AJUI_BreadCrumb ()
  Form.bc.Name("myBC")
  Form.bc.AddSection("sect_home";New object("label";"Home"))
  Form.bc.AddSection("sect_products";New object("label";"Products"))
  Form.bc.AddSection("sect_enterprise";New object("label";"Enterprise"))
  Form.bc.AddSection("sect_support";New object("label";"Support"))
  Form.bc.AddSection("Store";New object("label";"Store"))
  Form.bc.AddSection("sect_contact";New object("label";"Contact"))

End case

Form.bc.Draw()
```

To do this, simply call the member functions of the properties described in the corresponding chapters. Test, the different compositions and types. Also change the property values for different states using the desired constants. Finally, encapsulate everything in an "on load" event and launch the "Draw" function.

For information, the default values make it possible to create an "arrow" model.

We have reviewed the construction bases and for the next step, we strongly recommend that you use the "AJUI_BreadcrumbLab" application, which is provided to you with the source code and can be used as a demonstrator for the component (Lab and HDI)

AJUI_BreadcrumbLab provides you with an environment that allows you to better understand the graphical results of the different properties and to quickly create templates to simplify the definition of your Breadcrumb. It also shows clearly which properties can be used by each model. It is delivered with a number of predefined templates.

7 Conclusion

The purpose of this document was to introduce you to the theoretical principles of the component as well as the different methods, formulas and properties available to you to generate breadcrumbs.

As for the practical elements presented, they are intended to allow you to get off to a good start and to address some specific cases that could arise in the use of the component.

If you want help for the implementation of the component AJUI_BreadCrumb in your application. You want to modify or extend its functionalities for a specific use. You want to have the source code of the AJUI_BreadCrumb component in order to perpetuate its use in your application with future versions of 4D. Feel free to contact us to discuss it.