AJUI Banner

User Manual

Version Control	Date	Comment (Change)	Author
1.0	11.04.2019	First version	Gary Criblez
1.1	02.05.2019	CornerRibbon and modification HDI	Maurice Inzirillo
1.5	17.06.2019	Spinner animation	Gary Criblez
1.5.1	11.06.2019	Subform Compatibility	Gabriel Inzirillo
1.6	10.09.2019	CornerRibbon – specific positioning	Gabriel Inzirillo

1 Introduction

1.1 What is AJUI

AJUI_Banner is based on an original idea proposed during a training session given by 4D expert Olivier Deschanels from 4D SAS, that we have implemented as a 4D component.

AJUI Banner is a 4D component developed with 4D 17 R3. Its use is intended for 4D developers. It allows you to generate banners and tags in the context of a form as an overlay picture.

The purpose of this component is to enable 4D developers to enhance their interface in a simple and fast way by automating the generation of visual elements that clearly and quickly provide information to users..

The component allows to generate a banner object instance giving access to the different properties associated with it. These can be manipulated using formulas that are also integrated into this instance (we will come back to the chapter on properties and the chapter on getting started). The call of the image generation is also made from the object. Image creation is done using the SVG methods available in 4D.

The picture is then associated with a picture variable in the target form and it will finally be properly placed according to the type of banner selected.

1.2 Banners

Banners are differentiated into two formats or types called "cornerRibbon" banner or "window" banner.

A banner is necessarily associated with a picture form variable (see chapter Getting Started). This one will receive the banner. The positioning of the picture variable will also be managed by the component according to the type selected.

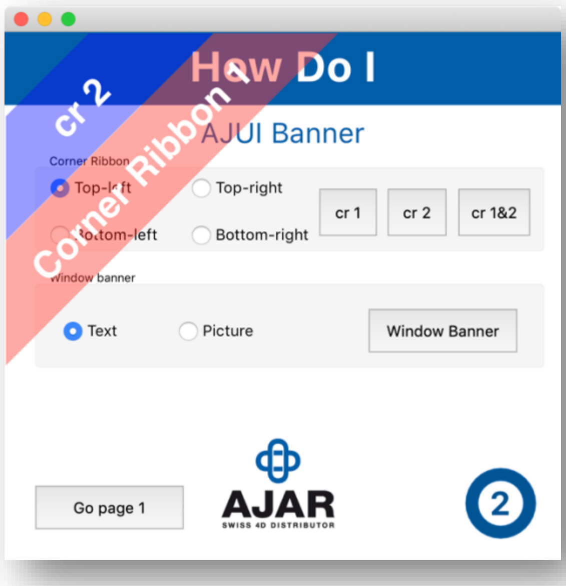
The display of the picture variable is the responsibility of the developer.

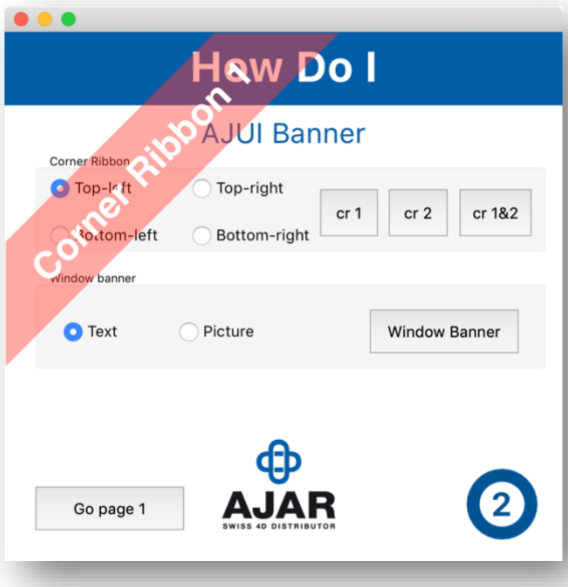
1.2.1 Type « cornerRibbon »

The corner Ribbon banner that can be placed at one of the four corners of a form.

The corner ribbon shows a banner that will be arranged on a diagonal depending on its position.

Its main purpose is to provide clear and distinct information to the user.





Warning: since it is a picture, if you put the picture object that will hold it in the foreground and you have fields or variables below you will not be able to access them, even they are visible. Take care to size it properly in terms of its properties (size and choice of side) and to give it a location that does not hide the use of the form. Even if visually it represents a diagonally shaped banner, the image is square in shape.

How to use it

The use of the corner ribbon banner is useful when you want to give important information to the user about a dataset. For example, if the record loaded in the form is currently locked, you can inform via the banner of the status of the form.

On the same principle, one can imagine to indicate when loading an invoice, the fact that it is paid or that a customer is considered as a bad payer or that a priority level is displayed. Generally speaking, the idea is that the banner will attract the user's attention to mention important information.

1.2.2 Type « window »

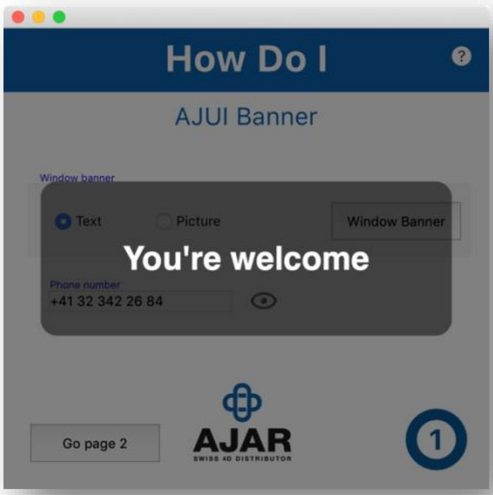
The "window" banner is a picture covering the entire associated form, so it must be placed in the foreground of the form page. It can contain either *text* or an *picture*.

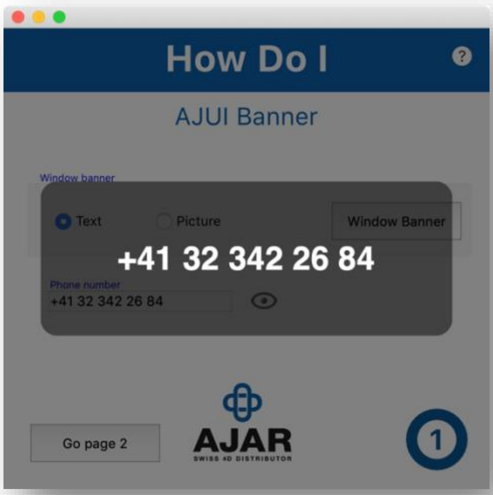
In the case of text, AJUI_Banner draws a first rectangular shape, used to cover the entire form, with which is associated a background color with an opacity rate. Then, it centers a second shape, the banner (central rectangle) and its textual content.

When using a picture, AJUI_Banner also applies a first global overlapping rectangle and then centers the associated picture.

The purpose of the window banner is mainly to report an ongoing processing to the user, or to display a visual help picture by overlapping the form.

How to use it

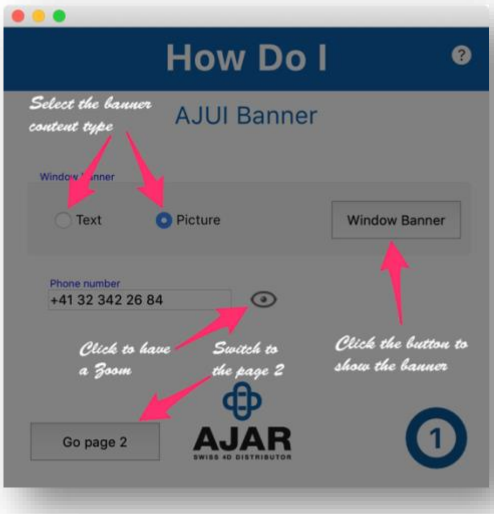


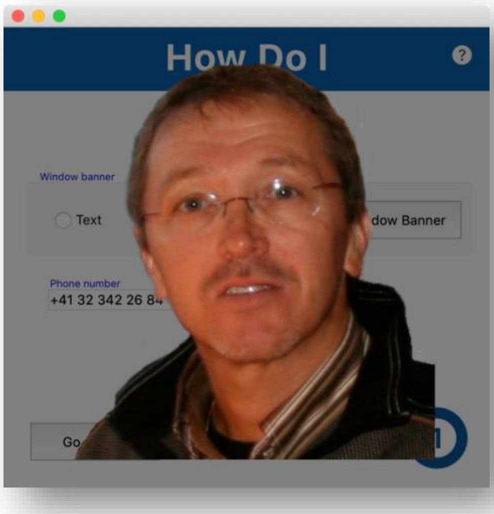


The use of the "window" window banner is particularly of interest for two situations. If a processing operation is started and it may take some time, it is recommended to inform the user and to prevent any possible actions on his part on the current form.

The second possible usage is to display an image. We can thus imagine displaying an overlay image (semi-transparent black background) allowing the user to get the necessary information on the use of fields, buttons, etc. in a complex form to be understood (figure on the left). Or display a picture of a face or an object, etc. (right figure).

Note: The "window" type will mainly block events related to mouse interactions such as click, focus, etc. since it affects the image in the foreground. However, interactions with the keyboard such as tabulation cannot be stopped by the banner. So be careful to manage the behaviors that the banner cannot prevent.

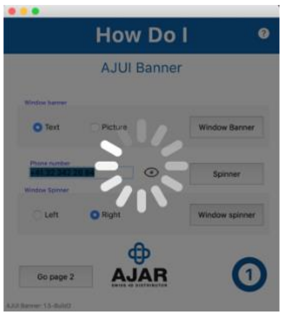




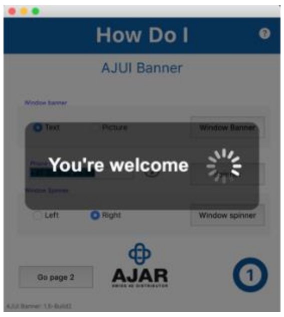
1.2.3 Types « spinner » and « windowSpinner »

Spinners are extensions of the “window” banner. The « spinner » is an image representing a kind of wheel that is animated to suggest a loading time to the user. We use a Worker to achieve the animation effect.

The first type allows you to display the spinner alone in the center of the screen and by adding a background as for the window type.



The second type "windowSpinner" will do as the previous one except that a text box is included and the spinner will be located to the right or left of the text.



1.2.4 Limitations

Since the SVG language does not provide the possibility of vertical alignment within the text boxes (TextArea) and for our own sake of consistency, we have decided to vertically center the text by setting a constraint :

- **The text must fit on a single line !**

1.3 Subform Compatibility

The differents types of banners cannot be displayed inside a subform. However, you can define and call the drawBanner within a subform and it will be displayed in the main form. Don't forget to put the picture variable at the main form level. (see Getting Started Chapter)

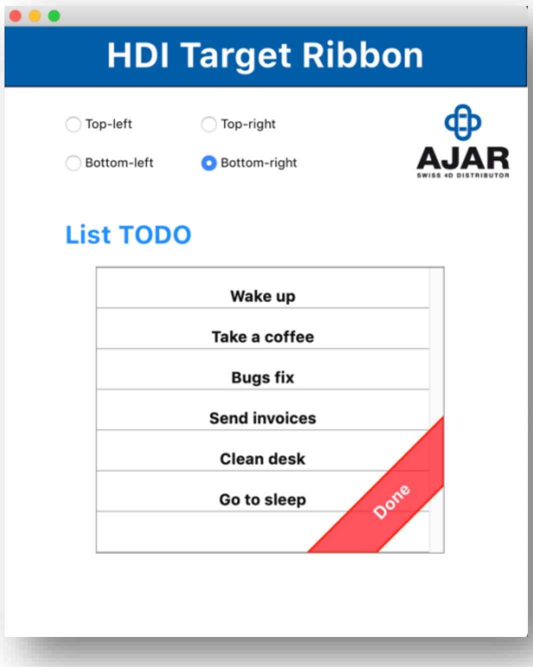
1.4 CornerRibbon – specific positioning

Since version 1.6 of AJUI Banner, it is possible to associate the ribbon banner directly to a form object or a specific area. For this purpose, two new formulas are available :

- **RibbonTargetName**
- **RibbonTargetCoordinates**

The first formula expects the name of a form object. The component will retrieve the coordinates of it, in order to position the ribbon at one of its four angles (top-left, top-right, bottom-left and bottom-right). Warning, the form object must be part of the main form.

The second formula allows you to pass the coordinates directly (left, top, right, bottom) in order to create an area that will receive the ribbon at one of its four corners. This method allows you to bypass the limit of the first method by specifying, for example, coordinates that correspond to the position of a subform object (see the 4D **CONVERT COORDINATES** method for converting coordinates between subform and form).


HDI Target Ribbon

Top-left Top-right
Bottom-left [✓ Bottom-right]

List TODO

Wake up
Take a coffee
Bugs fix
Send invoices
Clean desk
Go to sleep

2 Component methods

2.1 New AJUI_Banner

This method will allow you to generate an instance of AJUI Banner as an object. This object will include all the properties that can be used to create a banner and the formulas to modify its values. We will discuss each property and the formula related to each one in the next chapter..

2.1.1 Function : DrawBanner

It is a formula of the instance allowing to launch the creation and the positionning of the banner.

We can call him by using *MyInstance.DrawBanner( )*. Its purpose is to retrieve all the properties contained in the object (the banner instance) that called it. It will then create a picture based on an SVG that is generated on the basis of the different properties that you defined. And finally, it will move your picture variable to the appropriate place in your form and associate the generated picture with it..

2.2 Dialog_HDI_AJUI_Banner

This HDI (How Do I) method allows you to use a shared form of the component. This is an example form that we provide so you can get an overview of the component's rendering (the screenshots in Chapter 1 are made using the HDI form).

2.3 AJUIB_Info

This method returns the information about the version of your component.

3 Properties list

A banner has a set of properties that will allow you to define its appearance in the host form.

In this chapter, we will review the different existing properties accessible by a formula that acts as Setter and also as Getter if no parameters are passed to them.

All formulas can be invoked at the first level of the object.

3.1 Banner

Properties related to the banner at the second level of the object: **MyBanner.**banner

Nom	Type	Description	Par défaut	Formule
name	string	name of the banner if type is *window* or name of the cornerRibbon if the type is cornerRibbon	AJUI_Banner_2use or AJUI_cr_2use	BannerName
bannerBGColor	string	Background color of the banner. It is possible to define an opacity rate (%). See default value	black:50	BannerBGColor
bannerHeight	longint	Height of the banner.	150	BannerHeight
bannerWidth	longint	Width of the banner.	400	BannerWidth
borderColor	string	Color of the banner border.	darkgray	BorderColor
borderSize	longint	Size of the banner border.	0	BorderSize
cornerRadius	longint	Corner radius.	15	CornerRadius
position	string	Banner position for the "cornerRibbon" type. Possible position: top-left | top-right | bottom-left | bottom-right	bottom-right	BannerPosition
type	string	Type of banner to use. Possible type: cornerRibbon | window spinner | windowSpinner	cornerRibbon	BannerType

Banner – Formulas and their parameters

Formula name	Parameter(s)
BannerName	- Banner name (string)
BannerBGColor (couleur)	- Background color (string)
BannerHeight (hauteur)	- Banner height (longint)
BannerWidth (largeur)	- Banner width (longint)
BorderColor (couleur)	- Border color (string)
BorderSize (taille)	- Edge size (longint)
CornerRadius (arrondi)	- Corner radius (longint)
BannerPosition (position)	- "cornerRibbon" banner position (string)
BannerType (type)	- Banner type (string)

3.2 Text

Properties related to the text at the second level of the object: **MyBanner.text**

Name	Type	Description	Default value	Formula
message	string	Message that will be displayed in the banner	AJUI Banner	Message
textColor	string	Color of the text	White	TextColor
textFontName	string	Name of the font. Some fonts may not be SVG compatible	Arial	TextFontName
textFontSize	longint	Font size	32	TextFontSize
textFontStyle	string	SVG property that can be applied at the text level. Possible property: normal | italic | oblique | inherit	normal	TextFontStyle
textFontWeight	string	SVG property that can be applied at the text level. Possible property: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	bold	TextFontWeight

Text – Formulas and their parameters

Formula name	Parameter(s)
Message (string)	- Message from the banner (string)
TextColor (Color)	- Text color (string)
TextFontName (font)	- Name of the font (string)
TextFontSize (size)	- Font size (longint)
TextFontStyle (style)	- Font style (string)
TextFontWeight (weight)	- Font weight (string)

3.3 Window

Properties related to the window on the second level of the object: **MyBanner.**window

Name	Type	Description	Default value	Formula
windowBGColor	string	Background color of the "window" banner. It is possible to define an opacity rate (%). See default value	black:50	WindowBGColor
windowRef	longint	Reference to the form to be filled in with the banner. The value 0 represents the current form.	0	WindowRef

Window – Formulas and their parameters

Formula name	Parameter(s)
WindowBGColor (color)	- Background color (string)
WindowRef (number)	- Reference to the window form (longint)

3.4 Picture

Properties related to the image at the second level of the object: **MyBanner.**picture

Name	Type	Description	Default value	Formula
isPicture	boolean	If true, the banner hold a picture, if not the banner hold a text. Only applies to the "window" banner	False	IsPicture
picturePath	string	File path of the picture to be displayed. The picture must be in the 4D application's resource folder and the path is always relative. Example : "Images/info.png" (*used the 4D folder separator constant to build the string path in a clean way*)	empty string	PicturePath

Picture – Formulas and their parameters

Formula name	Parameter(s)
IsPicture (use picture)	- Use picture or text (boolean)
PicturePath (file path)	- File path in the "resources" folder (string)

3.5 Spinner

Properties related to the spinner at the second level of the object : **MyBanner.banner.**spinner

Name	Type	Description	Default value	Formula
scale	real	The scaling of the spinner in relation to the window. This does not apply to the type "windowSpinner" because its size will adapt according to the text box	0.5	SpinnerScale
side	string	Alignment of the spinner to the text. Possible value : left | right	right	WindowSpinnerSide

Picture – Formules et leurs paramètres

Formula name	Parameter(s)
SpinnerScale (scale)	- Spinner scale » (real)
WindowSpinnerSide (align)	- Spinner align (string)

3.6 Ribbon

Properties related to the ribbon at the second level of the object: **MyBanner.banner.ribbon**

Name	Type	Description	Default value	Formula
targetName	longint	Name of the form object that will receive the ribbon	Empty string	RibbonTargetName
TargetCoordinates	object	Coordinates of an area that will receive the ribbon.	0 for each coordinate	RibbonTargetCoordinates

Picture – Formules et leurs paramètres

Formula name	Parameter(s)
RibbonTargetName (name)	- Nom de l'objet de formulaire (string)
RibbonTargetCoordinates (left, top, right, bottom)	- Left coordinate (longint) - Top coordinate (longint) - Right coordinate (longint) - Bottom coordinate (longint) In Getter mode, the formula returns an object containing all four coordinates.

4 Banner management formulas

The banner instance contains three formulas to manage the life cycle of a banner.

- **DrawBanner** : this formula allows you to launch the banner generation based on the properties and make the image form object receiving it visible.
- **HideBanner** : This method allows you to hide the image form object linked to the instance. It will also execute the "StopSpinner" formula according to the type of banner in the instance.
- **StopSpinner** : This formula allows you to interrupt the animation of the spinner using a flag stored in the 4D Storage. It also unloads the image form object to avoid edge effects.

Warning : Spinners use a flag stored in Storage 4D. Be careful not to delete this property: *Storage.AJUIB_spinner.activate*.

5 Getting Started

AJUI Banner is a fairly simple component to handle if you have assimilated the different properties available in the object.

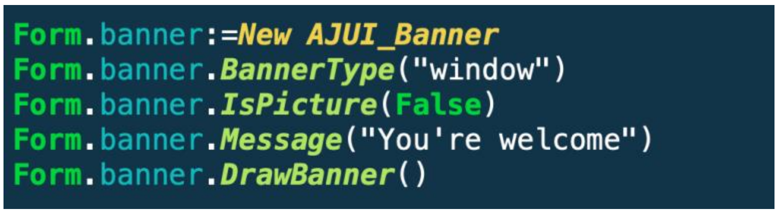
To start with, we will briefly present three code extracts that allow you to create banners.

In all cases, you must add a **form object** that will contain the banner and that must meet the following conditions:

- This is an object form associated with an picture variable.
- It must be called *AJUI_Banner* and be put in place on the first page of your form.
- If it is on another page of the form, add the page number to its property name (eg *AJUI_Banner2* for page 2, *AJUI_Banner3* for page 3, etc.).
- Generally the form object is positioned in the foreground except for specific cases according to your needs..

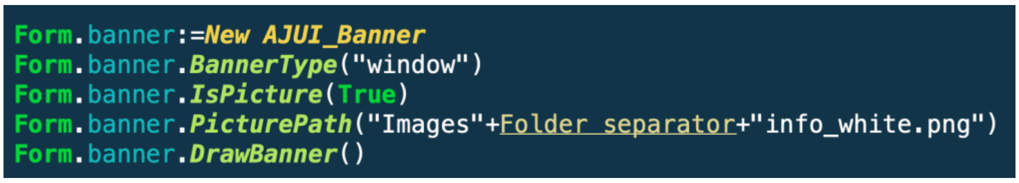
The management of the picture visibility is not handled by the component. We advise you to set the form object to invisible by default and make it visible when required. We also recommend that you avoid using the form object in page zero (even if it is possible to do so) because the form objects of the other pages often override it.

Case 1: "Window" banner with text



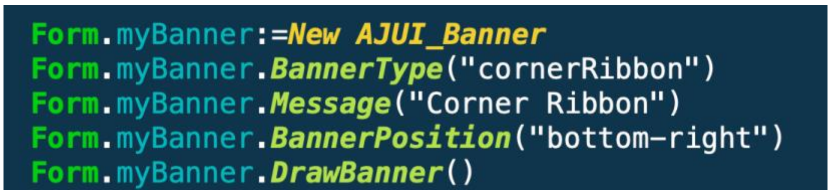
```
Form.banner:=New AJUI_Banner
Form.banner.BannerType("window")
Form.banner.IsPicture(False)
Form.banner.Message("You're welcome")
Form.banner.DrawBanner()
```

Case 2: "Window" banner with picture



```
Form.banner:=New AJUI_Banner
Form.banner.BannerType("window")
Form.banner.IsPicture(True)
Form.banner.PicturePath("Images"+Folder separator+"info_white.png")
Form.banner.DrawBanner()
```

Case 3: Tag banner positioned at the bottom right



```
Form.myBanner:=New AJUI_Banner
Form.myBanner.BannerType("cornerRibbon")
Form.myBanner.Message("Corner Ribbon")
Form.myBanner.BannerPosition("bottom-right")
Form.myBanner.DrawBanner()
```

You are then free to try out the different possibilities of banner construction and model them at your convenience.

Reminder on restrictions: *The component requires the text to fit on a single line, so make sure to adapt the size of the banner to the size of your text.*

For the "spinner" and "windowSpinner" types, you just have to reproduce what you did for the "window" type by just changing the name of the type. Also try to change the scale for the " spinner " and modify the alignment for " windowSpinner ". Don't forget to use the "StopSinner" formula to stop the animations.

6 Conclusion

The purpose of this document was to present the theoretical principles of the component as well as the different methods, formulas and properties at your disposal to be able to manage the construction and animation of tooltips.

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. For more practical information and a better visualization of the tooltip generation rendering, we invite you to try the test application (lab) that is provided with the component.

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