Date : 07-Feb-22

Barcode generator (1.0.2)

# Overview

The Barcode Generator component can be used to encode data into a barcode.

The component can encode the data into the following formats:

* CODE128
* CODE39
* EAN13
* EAN8
* EAN5
* EAN2
* UPC
* ITF14
* MSI
* MSI10
* MSI11
* MSI1010
* MSI1110

For further information, you can refer to the [JsBarcode](http://lindell.me/JsBarcode/" \t "_blank) website.

## Use case

### Identify products in retail stores, warehouses and invoice

## Features:

### Generate barcode by given data and format

## Percentage of re-use:

80-90% (Data can be customizable and skins are not customized but can be changed manually)

# Getting Started

## Prerequisites

Before you start using the barcode generator component, ensure you have the following:

* [HCL Foundry](https://manage.hclvoltmx.com/)
* Volt MX Iris

## Platforms Supported

### Mobile

#### iOS

#### Android

### Tablets

#### iOS

#### Android

### PWA

## Importing the app

## You can import the Forge components only into the apps that are of the Reference Architecture type.

## **To import the barcode generator component, do the following:**

## Open your app project in Volt MX Iris.

1. In the Project Explorer, click the **Templates** tab.

Graphical user interface, text, application

Description automatically generated

3.Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.

Graphical user interface, text, application, Teams

Description automatically generated

4.Click **Browse** to navigate to the location of the component, select the component, and then click **Import**. The component and its associated widgets and modules are added to your project.

## 

Once you have imported a component to your project, you can easily add the component to a form. For more information, refer [Add a Component to a Form](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/C_UsingComponents.html#add-a-component-to-a-form).

## Building and previewing the app

## Refer to below links:

### [**Build & Generate Native Apps**](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/Cloud_Build_in_VoltMX_Iris.html#cloud)

### [**Publish Apps to Enterprise App Store**](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/EAS.html#accessing)

# References

## Dynamic Usage

You can also add an **barcode generator** component dynamically. To do so:

1. In the **Project Explorer**, on the **Projects** tab, click **Controllers** section to access the respective **Form Controller**. Create a method and implement the code snippet similar to the sample code mentioned below.

In the code snippet, you can edit the properties of the component as per your requirement. For more information, see Setting Properties.

var barcodegenerator = new com.voltmxmp.barcodegenerator(

{

"clipBounds": true,

"height": "60%",

"id": "barcodegenerator",

"isVisible": true,

"left": "0%",

"skin": "slFbox",

"top": "0%",

"width": voltmx.flex.USE\_PREFFERED\_SIZE

}, {}, {});

/\* Setting the component's Pass Through and Custom properties \*/

barcodegenerator.dataToEncode = "voltmx";

barcodegenerator.barcodeFormat = "CODE128";

barcodegenerator.barcodeWidth = "2";

barcodegenerator.barcodeHeight = "40";

barcodegenerator.backGroundColor = "#ffffff";

barcodegenerator.lineColor = "#000000";

barcodegenerator.displayValue = true;

barcodegenerator.barcodefontStyle = "none";

barcodegenerator.barcodefontFamily = "";

barcodegenerator.barcodefontSize = "";

barcodegenerator.textAlign = "center";

barcodegenerator.textPosition = "bottom";

barcodegenerator.textMargin = "";

barcodegenerator.marginTop = "";

barcodegenerator.marginBottom = "";

barcodegenerator.marginLeft = "";

barcodegenerator.marginRight = "";

/\*Adding the Barcode Generator Arrow component to a Form\*/

this.view.add(barcodegenerator);

1. **Save** the file.

## Properties

You can use a component's **Properties** to customize and configure the elements. These elements can be UI elements, service parameters, and so on. You can set the properties from the Iris's Properties panel on the right-hand side. You can also configure these properties using a JavaScript code.

1. General

[[Open](javascript:void(0);)Data to Encode](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the data the user wants to encode in the generator. |
| **Syntax:** | dataToEncode |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.dataToEncode = "Temenos"; |

[[Open](javascript:void(0);)Barcode Format](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the format of the barcode that the user wants to generate. |
| **Syntax:** | barcodeFormat |
| **Type:** | List Selector |
| **Read/Write:** | Write |
| **Values:** | * CODE128 * CODE39 * EAN13 * EAN8 * EAN5 * EAN2 * UPC * ITF14 * MSI * MSI10 * MSI11 * MSI1010 * MSI1110 |
| **Example:** | this.view.<componentID>.barcodeFormat = "CODE128"; |

1. Barcode Image

[[Open](javascript:void(0);)Barcode Single Bar Width](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the width of a single bar of the barcode. |
| **Syntax:** | barcodeWidth |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.barcodeWidth = "2"; |

[[Open](javascript:void(0);)Barcode Height](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the height of the barcode. |
| **Syntax:** | barcodeHeight |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.barcodeHeight = "40"; |

[[Open](javascript:void(0);)Background Color](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the background color of the barcode. |
| **Syntax:** | backGroundColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.backGroundColor = "#ffffff"; |

[[Open](javascript:void(0);)Line Color](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the colors of the bars of the barcode. |
| **Syntax:** | lineColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.lineColor = "#000000"; |

[[Open](javascript:void(0);)Barcode Top Margin](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the top margin of the barcode. |
| **Syntax:** | marginTop |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | If you do not specify any margin, the component will pick the default margin. |
| **Example:** | this.view.<componentID>.marginTop = ""; |

[[Open](javascript:void(0);)Barcode Bottom Margin](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the bottom margin of the barcode. |
| **Syntax:** | marginBottom |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | If you do not specify any margin, the component will pick the default margin. |
| **Example:** | this.view.<componentID>.marginBottom = ""; |

[[Open](javascript:void(0);)Barcode Left Margin](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the left margin of the barcode. |
| **Syntax:** | marginLeft |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | If you do not specify any margin, the component will pick the default margin. |
| **Example:** | this.view.<componentID>.marginLeft = ""; |

[[Open](javascript:void(0);)Barcode Right Margin](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the right margin of the barcode. |
| **Syntax:** | marginRight |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | If you do not specify any margin, the component will pick the default margin. |
| **Example:** | this.view.<componentID>.marginRight = ""; |

1. Barcode Text

[[Open](javascript:void(0);)Display Value](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies whether you want to display the text of the barcode. |
| **Syntax:** | displayValue |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.displayValue = true; |

[[Open](javascript:void(0);)Font Style](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font style of the text of the barcode. |
| **Syntax:** | barcodefontStyle |
| **Type:** | List Selector |
| **Read/Write:** | Write |
| **Values:** | * none * bold * italic * bold italic |
| **Example:** | this.view.<componentID>.barcodefontStyle = "none"; |

[[Open](javascript:void(0);)Font Family](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font family of the text of the barcode. |
| **Syntax:** | barcodefontFamily |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | If you do not specify any font, the component will pick the default font. |
| **Example:** | this.view.<componentID>.barcodefontFamily = "Arial"; |

[[Open](javascript:void(0);)Font Size](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the font size of the text of the barcode. |
| **Syntax:** | barcodefontSize |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | If you do not specify any font size, the component will pick the default font size. |
| **Example:** | this.view.<componentID>.barcodefontSize = "12"; |

[[Open](javascript:void(0);)Text Align](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the alignment of the text of the barcode. |
| **Syntax:** | textAlign |
| **Type:** | List Selector |
| **Read/Write:** | Write |
| **Values:** | * left * center * right |
| **Example:** | this.view.<componentID>.textAlign = "center"; |
|  |  |

[[Open](javascript:void(0);)Text Position](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the position of the text of the barcode. |
| **Syntax:** | textPosition |
| **Type:** | List Selector |
| **Read/Write:** | Write |
| **Values:** | * top * bottom |
| **Example:** | this.view.<componentID>.textPosition = "bottom"; |
|  |  |

[[Open](javascript:void(0);)Text Margin](javascript:void(0);)

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Specifies the margin of the text with respect to the barcode. |
| **Syntax:** | textMargin |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | If you do not specify any margin, the component will pick the default margin. |
| **Example:** | this.view.<componentID>.textMargin = ""; |

## Events

### errorCallback

|  |  |
| --- | --- |
| **Description:** | Invoked when an error occurs within the component. |
| **Syntax:** | errorCallback |
| **Parameters:** | *exception [JSON]* : Details about the exception. |
| **Example:** | this.view.<componentID>.errorCallback = function(exception) {  alert("exception: "+JSON.stringify(exception));  } |

Uncaught errors

For uncaught errors, you can add your own handling on function window.onerror() in barcode.html.

## API’s

### generate

This API generates the barcode in the drag and drop area of the component.

**Category**

Custom

**Syntax**

generate

**Parameters**

None

**Return Value**

None. It displays the generated barcode.

# Revision History

App version 1.0.2:

## Known Issues

N/A

## Limitations

N/A