Date: 30 Sept 2024

SIGNATURE CAPTURE

VERSION: 2.0.3

1. **OVERVIEW**

Signature Capture is a custom widget that can capture signatures and other written content and store it as images.

## **Use case**

### A delivery app where the user needs to sign into the app to authorize and accept the delivery.

## **Features**:

* The images can be stored either on the **device**, or on a **Network File System**.
* This component uses NFI and doesn’t work on Iris Preview.
* Signature canvas for capture.
* Signature saved as image (png or jpeg).
* Image can be saved on device or on network File System (NFS).
* Canvas and pen color are customizable.

## **Percentage of re-use:**

Approximate 85% of reuse.

# **GETTING STARTED**

## **Prerequisites**

Before you start using the **Signature Capture** component, ensure the following:

• [HCL Foundry](https://manage.hclvoltmx.com/)

• Volt MX Iris

## **Platforms** **Supported**

## Mobile

1. iOS
2. Android

## Tablet, iPad

### **Note:**

1. While building the app for Android in Debug and Release mode, in order to save signatures to the device. You need to add READ\_PHONE\_STATE and WRITE\_EXTERNAL\_STORAGE permissions in project settings.

### Graphical user interface, text, application, email Description automatically generated

2. Enter the given snippet in Child tag entries under <manifest > tag

<uses-permission android:name="android.permission.READ\_MEDIA\_IMAGES" />

## **Configuring Native Settings (iOS)**

Follow the given steps to enable the permissions.

1. From the Project explorer, navigate to the Assets tab.

2. Right click Media and select Resource Location to open the project resources folder.

3. In the browser window that opens, navigate to the common folder.

4. Open the infoplist\_configuration.json file with a text or code editor.

5. Add the given code at the end of the file. You can replace the values with your own descriptions.

{

"NSPhotoLibraryUsageDescription": "To add signature to device"

}



6. Save the file.

## **Importing the App**

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

## **To import the Signature Capture 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

1. Right-click **Components**, and then select **Import Component**. The **Import**

**Component** dialog box appears.

Graphical user interface, text, application, Teams

Description automatically generated

## 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**

After performing all the above steps, you can build your app and run it on your device. For more information, you can refer to the [Building and Viewing an Application](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/Cloud_Build_in_VoltMX_Iris.html#cloud) section of the Volt MX User Guide.

You can then run your app to see the Signature Capture work in real time.

# **REFERENCES**

## **Dynamic Usage**

You can also add the **Signature Capture** 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.
2. To Save the image in Network File System please change the component property save To Device as save to Network File System. In Dynamic initialization of the component the signatureCapture.saveTo property should be as **Network File System** (signatureCapture.saveTo **=** "Network File System") to save the image to NFS

/\* Creating signatureCapture component instance \*/

var signatureCapture = new com.voltmx.signaturecapture({

id: "signaturecapture",

isVisible: true,

top:"0dp",

left:"0dp",

width:"100%",

height:"60%",

clipBounds: true,

autogrowMode: voltmx.flex.AUTOGROW\_NONE,

layoutType: voltmx.flex.FLOW\_VERTICAL,

skin: "slFbox",

zIndex:1

},{},{});

/\* Setting component's properties \*/

signatureCapture.penColor = "000000";

signatureCapture.canvasBackground = "FFFFFF";

signatureCapture.saveSignature = true;

signatureCapture.saveAs = "png";

signatureCapture.saveTo = "Device";

/\*Adding the Signature Capture component to a form\*/

this.view.add(signatureCapture);

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

1. Save the file

4. For NFS call the Object service in the app init event as Follows

function initApp(){

var serviceName = "objectServiceName";

var serviceType = "online";

var currObj = voltmx.sdk.getCurrentInstance();

var objSvc = currObj.getObjectService(serviceName, {

"access": serviceType

});

var fileStorage= objSvc.getFileStorage();

this.\_fileStoreObj = fileStorage;

if (fileStorage === null) {

alert("File Storage Init Failed !");

} else {

//alert("File Storage Init successfull !");

}

}

## **Properties**

The properties provided on the **Component** tab allows you to customize the elements in the **Signature Capture** component. These elements can be UI elements, service parameters, and so on. You can set the properties from the Volt MX Iris Properties panel on the right-hand side. You can also configure these properties using a JavaScript code.

### **General Properties**

1. **Pen Color Hexcode (penColor)**

| **Description:** | Specifies the color of the pen that is used to draw on the canvas. |
| --- | --- |
| **Syntax**: | penColor |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.penColor= "000000"; |
| **Remarks:** | The default value for the property is “000000”. |

1. **Background Color Hexcode (canvasBackground)**

| **Description:** | Specifies the background color of the signature canvas. |
| --- | --- |
| **Syntax**: | canvasBackground |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.canvasBackground= "FFFFFF"; |
| **Remarks:** | The default value for the property is “FFFFFF”. |

1. **Canvas Width(canvasWidth)**

| **Description:** | Specifies the width of the signature canvas. |
| --- | --- |
| **Syntax**: | canvasWidth |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.canvasWidth= "94%"; |
| **Remarks:** | The default value for the property is "94%". |

1. **Canvas Height(canvasHeight)**

| **Description:** | Specifies the height of the signature canvas. |
| --- | --- |
| **Syntax**: | canvasHeight |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.canvasHeight= "75%"; |
| **Remarks:** | The default value for the property is "75%". |

1. **Save Signature as Image(saveSignature)**

| **Description:** | Specifies whether the component should save the signature. |
| --- | --- |
| **Syntax**: | saveSignature |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.saveSignature = true; |
| **Remarks:** | The default value for the property is true. |

1. **Save Button Visibility(isSaveVisibile)**

| **Description:** | Toggles the visibility of the Save Button. |
| --- | --- |
| **Syntax**: | isSaveVisibile |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.isSaveVisibile= true; |
| **Remarks:** | The default value for the property is true. |

1. **Clear Label Visibility (isCancelVisible)**

| **Description:** | Toggles the visibility of the Clear Label. |
| --- | --- |
| Syntax: | isCancelVisible |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.isCancelVisible= true; |
| **Remarks:** | The default value for the property is true. |

1. **Save Button Text(textSave)**

| **Description:** | Specifies the text that you want to display on the save button. |
| --- | --- |
| **Syntax**: | textSave |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.textSave= "SAVE"; |
| **Remarks:** | The default value for the property is “SAVE”. |

1. **Clear Label Text (textCancel)**

| **Description:** | Specifies the text that you want to display on the clear label. |
| --- | --- |
| **Syntax**: | textCancel |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.signatureCapture.textCancel="Clear"; |
| **Remarks:** | The default value for the property is “Clear”. |

### 

### **Save Properties**

1. **Save To(saveTo)**

| **Description:** | Specifies where the component should save the image. |
| --- | --- |
| **Syntax**: | saveTo |
| **Type:** | * String * List Selector |
| **Read/Write:** | Read + Write |
| **Values:** | * Device * Network File System |
| **Example:** | this.view.signatureCapture.saveTo= "Device";  this.view.sigantureCapture.saveTo="Network File System"; |
| **Remarks:** | On Iris, you can select this value from a drop-down list. |

1. **Image Save Format(saveAs)**

| **Description:** | Specifies the file format of the saved image. |
| --- | --- |
| **Syntax**: | saveAs |
| **Type:** | * String * List Selector |
| **Read/Write:** | Read + Write |
| **Values:** | * Png * Jpeg |
| **Example:** | this.view.signatureCapture.saveAs= "png"; |
| **Remarks:** | On Iris, you can select this value from a drop-down list. |

**Skins Section**

1. **Save Button Skin (skinSave)**

| **Description:** | Specifies the skin of the save button. |
| --- | --- |
| **Syntax**: | skinSave |

1. **Clear Label Skin (skinCancel)**

| **Description:** | Specifies the skin of the clear label. |
| --- | --- |
| **Syntax**: | skinCancel |

## **Events**

1. **On Save Image Success(onSaveImageSuccess)**

| **Description:** | Invoked when the component successfully saves an image. |
| --- | --- |
| **Syntax**: | onSaveImageSuccess |
| **Parameters:** | response[string]:  Response from the platform on which the components save the image |
| **Example:** | this.view.componentID.onSaveImageSuccess = function(response)  {  alert(response);  }.bind(this); |

1. **On Save Image Failure(onSaveImageFailure)**

| **Description:** | Invoked when the component failed to save the image. |
| --- | --- |
| **Syntax**: | onSaveImageFailure |
| **Parameters:** | response[string]:  Information about the failure of saving the image |
| **Example:** | this.view.componentID.onSaveImageFailure = function(response)  {  alert(response);  }.bind(this); |

1. **On Error Callback(onErrorCallback)**

| **Description:** | Invoked when an error occurs in the component. |
| --- | --- |
| **Syntax**: | onErrorCallback |
| **Parameters:** | response[string]:  Information about the exception raised by the component |
| **Example:** | this.view.componentID.onErrorCallback = function(response)  {  alert(response);  }.bind(this); |

1. **On Check Validity(onCheckValidity)**

| **Description:** | Invoked when save signature as image property is set as false. |
| --- | --- |
| **Syntax**: | onCheckValidity |
| **Parameters:** | bool[boolean]:  Checks whether the signature is present or not. |
| **Example:** | this.view.componentID.onCheckValidity = function(bool)  {  alert(bool);  }.bind(this); |

## **API’s**

### **getSignatureFromDevice**

| **Description:** | Fetches the decrypted base64 of the signature from the device. |
| --- | --- |
| **Syntax**: | getSignatureFromDevice |
| **Parameters:** | None |
| **Return Value:** | signature[string]:  decrypted base64 value of the signature. |
| **Example:** | this.view.signatureCapture.getSignatureFromDevice(); |

# **REVISION HISTORY**

App version :2.0.3

## **Known Issues**

## NA

## **Limitations -** The maximum values for the width and height of the canvas are

## Width: 338Dp or 94%

## Height: 320Dp or 75%