Date : 29-May-24

History FORM

version: 1.0.2

1. **Overview**

Let’s you view past records for any entity to understand the asset’s performance and better plan for future use-cases. This is the most common use-case for any work request/service request scenario, in which the user needs access to historical records to understand the task in hand. Can be integrated into any entity in the data model for which historical records need to be maintained and displayed as a reference.

## **Use case**

### Maintaining Records

### Asset’s performance and better plan for future

### Workout or Daily Planner App

### **B. Features**

### A sample form displayed in a recent first format(The sorting order is configurable).

### All the historical inspection values (per service request) displayed in the form

### fully customisable look and skin properties to rebrand and redesign the form for your requirement

### JSON input format

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

70-80% (Data and skins are customized and can be changed manually)

# **Getting Started**

## **Prerequisites**

Before you start using the History Form component, ensure you have the following:

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

## **Platforms Supported**

### Mobile

#### iOS

#### Android

### Tablets

### PWA

## **Importing the Component**

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

## **To import the History Form component, do the following:**

## Open your app project in Volt MX Iris.

2. 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.

A screenshot of a computer

Description automatically generated with medium confidence

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 Iris User Guide.

You can then run your app to see the History Form work in real time.

# **3. References**

## **A. Dynamic Usage**

If you want to use the History Form component dynamically, you will need to import the component into your project Templates. Follow the given steps to do so

### Download the component from HCL VoltMX Marketplace as a zip file.

### Go to the Templates tab in your project explorer.

### Right click on Components and select Import Component.

### Navigate to where you downloaded your zip file and import it into Iris.

After you import the component into your project templates, you can add it to your app dynamically. To do so, follow the given steps

### Access the FormController of the form you want to add the component into.

### Create a function called createComponent(); and write the code inside it to create and configure the component.

### You can refer to the given sample code for more information.

/\*Creating a component's Object \*/

var Historyform = new com.voltmxmp.historyform(

{

"clipBounds": true,

"height": "100%",

"id": " historyform ",

"isVisible": true,

"left": "0dp",

"top": "0dp",

"width": "100%",

"zIndex": 1

}, {}, {});

//Configuring the component properties

Historyform.headerText = "Asset 6A578C";

Historyform.headerImageSrc = "voltmxqfs\_close\_1.png";

Historyform.headerIDText="TRANS";

Historyform.headerCategoryText="Transformers";

/\* add this line for web because no other channels

require scrolling property\*/

Historyform.enableScrollingParent=true;

//Adding the component to the form

this.view.add(Historyform);

},

1. **Save** the file.

## **B. Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the **History Form** component. You can set the properties directly on the **Component** tab or by writing a JavaScript.

[Open](javascript:void(0);)**1. Master Data(masterdata)**

|  |  |
| --- | --- |
| **Description:** | Specifies the list of data sources to the History Form. |
| **Syntax**: | Masterdata |
| **Type:** | Data Grid |
| **Read/Write:** | Write |
| **Example:** | var sampleData=  [  [{  "lblDate": "13th July,",  "lblTime": "8:10 hrs",  "lblInspectionValue":"ins007",  "lblInspectionID":"inspectionID"  },  [{  "lblRowHeader": "Temperature",  "lblRowHeaderValue": "100"  }, {  "lblRowHeader": "Climate",  "lblRowHeaderValue": "Good"  },  {  "lblRowHeader": "Result",  "lblRowHeaderValue": "PASS"  }]  ],  [{  "lblDate": "14th July,",  "lblTime": "8:20 hrs",  "lblInspectionValue":"ins234",  "lblInspectionID":"inspectionID"  },  [{  "lblRowHeader": "Temperature",  "lblRowHeaderValue": "100"  }, {  "lblRowHeader": "Result",  "lblRowHeaderValue": "PASS"  }]  ],  [{  "lblDate": "15th July,",  "lblTime": "8:30 hrs",  "lblInspectionValue":"ins567",  "lblInspectionID":"inspectionID"  },  [{  "lblRowHeader": "Climate",  "lblRowHeaderValue": "Good"  }, {  "lblRowHeader": "Result",  "lblRowHeaderValue": "PASS"  }]  ]  ]; |

2. **Header Text(titleText)**

|  |  |
| --- | --- |
| **Category:** | Pass Through |
| **Description:** | Specifies the title of the History Form. |
| **Syntax**: | titleText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.titleText = "Title"; |

1. **Close Button Visibility(iconVisibility)**

|  |  |
| --- | --- |
| **Category:** | Pass Through |
| **Description:** | Specifies the icon visibity of the History Form. |
| **Syntax**: | iconVisibility |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.iconvisibility =true; |

1. **Close Button Source(iconSrc)**

|  |  |
| --- | --- |
| **Category:** | Pass Through |
| **Description:** | Specifies the close button image of the History Form. |
| **Syntax**: | iconSrc |
| **Type:** | Src |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.iconSrc = "voltmx\_close.png"; |

1. **Asset Text(headerText)**

|  |  |
| --- | --- |
| **Category:** | Pass Through |
| **Description:** | Specifies the Asset Text of the History Form. |
| **Syntax**: | headerText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.headerText = "sample text"; |

1. **ID Text(headerIDText)**

|  |  |
| --- | --- |
| **Category:** | Pass Through |
| **Description:** | Specifies the ID Text of the History Form. |
| **Syntax**: | headerIDText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.headetIDText = "Title"; |

1. **Category(headerCategoryText)**

|  |  |
| --- | --- |
| **Category:** | Pass Through |
| **Description:** | Specifies the Category text of the History Form. |
| **Syntax**: | headerCategoryText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.headerCategoryText = "sample header"; |

1. **Enable Scrolling(flxParent)**

|  |  |
| --- | --- |
| **Category:** | Pass Through |
| **Description:** | Specifies the scrolling of the entire form(flxparent) in web only not for all the channels of the History Form. |
| **Syntax**: | enableScrollingParent |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID. enableScrollingParent = "true"; |

**Note: -**

By default, the enable scrolling feature is disabled if you want to run it in the web then enable that option for PWA

## **C. Events**

-- None of the events are exposed.

## **D. API’s**

This API sets the data to the History Form. You need to give the input data as a parameter.

You need to define this setData in the form controller and call this in the template controller using this syntax

### **1. SetData**

**Description:** This method is used to create records for the component.

**Syntax:** setData(data)

**Parameters:** data (type - json)

**Return Value:** None

**Example:** this.view.componentID.setData (sampleData);

**Remarks:** Keys in the json should always be same.

**Sample Input Format**

You need to add this in the **Form controller**.

postShow:function(){

var sampleData=

[

[{

"lblDate": "13th July,",

"lblTime": "8:10 hrs",

"lblInspectionValue":"ins007",

"lblInspectionID":"inspectionID"

},

[{

"lblRowHeader": "Temperature",

"lblRowHeaderValue": "100"

}, {

"lblRowHeader": "Climate",

"lblRowHeaderValue": "Good"

},

{

"lblRowHeader": "Result",

"lblRowHeaderValue": "PASS"

}]

]

]

this.view.historyform.setData(sampleData);

}

In the given JSON format:

* **title** is a String that specifies the Title of the history form.
* **SampleData** is a JSON Array that represents the number of sessions in an event.

The code that specifies about the calling of set data function in the **Template controller.**  
For example

setData : function(data){

this.view.segTemplate.data=data;

}

# **Revision History**

App version 1.0.2

## A. Known Issues

-

## B. Limitations

* Land Scape mode is not supported