24 Sep 2021

INFO CARDS (1.0.1)

1. Overview

Info cards let you display basic details and past historical values for any entry in the database.

## Use case

### During Inspections/ workorders, technicians/engineers need to view past values for each measurement point.

## Features:

## Info cards display past values in simple horizontal scrollable thumbnail format, hence easing the navigation across the performance records.

## Percentage of re-use:

Approximate % of reuse. It sets an expectation of how much can be used out of the box, and how much needs to be customized for a specific app.

# Getting Started

## Prerequisites

Before you start using the Info Cards component, ensure 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 Info Cards 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.

## Text Description automatically generated with low 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" \l "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 Info Cards work in real time.

# References

## Dynamic Usage

You can also add **Info Cards** 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.



/\* Creating InfoCard component instance \*/

var infoCard= new com.voltmx.infocard(

{

id: "infocard",

isVisible: true,

height: "100%",

top: "0dp",

left: "0dp",

width: "100%",

layoutType: voltmx.flex.FREE\_FORM,

autogrowMode: voltmx.flex.AUTOGROW\_NONE,

skin: "voltmxqfsSknInfoCard",

clipBounds:true,

zIndex:1

}, {}, {});

/\*Setting the component's properties\*/

infoCard.cardTitle ="#amb557766";

infoCard.cardName = "Ambient Temperature";

infoCard.cardDescription = "Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

infoCard.recordSectionLabel = "Previous Value";

infoCard.hasValues = true;

/\*Adding the Info Card component to a Form\*/

this.view.add(infoCard);

/\*Calling APIs from code \*/

/\* Sample data object\*/

data=[{time:"7:00 PM",

date: "19th Jun 2021",

value: "Validating the product",

responseType:"Null"

},{time:"8:00 PM",

date: "20th Jun 2021",

value: "Testing",

responseType:"Null"

}];

this.view.infocard.initialize(data);

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

2. Save the file

## Properties

The properties provided on the **Component** tab allows you to customize the elements in the **Info Card** 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. Card Title (cardTitle)

|  |  |
| --- | --- |
| **Description:** | Specifies the title for the card. |
| Syntax: | cardTitle |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard.cardTitle= "#amb557766"; |
| **Remarks:** | The default value for the property is “#amb557766”. |

1. Card Name(cardName)

|  |  |
| --- | --- |
| **Description:** | Specifies the name of the card. |
| Syntax: | cardName |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard.cardName= "Ambient Temperature"; |
| **Remarks:** | The default value for the property is “Ambient Temperature”. |

3. Card Description(cardDescription)

|  |  |
| --- | --- |
| **Description:** | Specifies the description of the card. |
| Syntax: | cardDescription |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard.cardDescription= "Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book."; |
| **Remarks:** | The default value for the property is "Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.". |

4.Record Section Label(recordSectionLabel)

|  |  |
| --- | --- |
| **Description:** | Specifies the label value of the card. |
| Syntax: | recordSectionLabel |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. recordSectionLabel = "Previous Values"; |
| **Remarks:** | The default value for the property is "Previous Values". |

**5. Enable Scrolling(enableScrolling)**

|  |  |
| --- | --- |
| **Description:** | Specifies scroll bars are needed or not for main flex. |
| Syntax: | enableScrolling |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. enableScrolling = true; |
| **Remarks:** | The default value for the property is true. |

**6. Contains Values(****hasValues)**

|  |  |
| --- | --- |
| **Description:** | Specifies if the values (date , time …) to be displayed on the card should be visible or not. |
| Syntax: | **hasValues** |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infoCard.hasValues= true; |
| **Remarks:** | The default value for the property is true. |

**7. Record Values (****recordValues)**

|  |  |
| --- | --- |
| **Description:** | Specifies the list of record values that you want to add. |
| Syntax: | **recordValues** |
| **Type:** | Datagrid, Array of JSON |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infoCard.recordValues= [{  time:"7:00 PM",  date: "19th Jun 2021",  value: "Validating the product",  responseType:"Null"  },  {  time:"8:00 PM",  date: "20th Jun 2021",  value: "Testing",  responseType:"Null"  }]; |
| **Remarks:** | This property does not reflect on the preview canvas of Voltmx Iris. |

**Skins Section**

**8. InfoCard Skin(infoCardSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the infocard flex. |
| Syntax: | infoCardSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. infoCardSkin= “voltmxqfsSknInfoCard”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknInfoCard”. |

**9. MainFlex Skin (flxMainSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the main flex. |
| Syntax: | flxMainSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. flxMainSkin= “voltmxqfsSknFlxMain”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknFlxMain”. |

**10.** **MeasurementID Skin (mesaurementIdSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the measurementID label. |
| Syntax: | mesaurementIdSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. mesaurementIdSkin = “voltmxqfsSknLblMeasurementID”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknLblMeasurementID”. |

**11. MeasurementName Skin (measurementNameSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the measurementName label. |
| Syntax: | measurementNameSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. infoCardSkin= “voltmxqfsSknLblMeasurementID”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknLblMeasurementID”. |

**12. MeasurementDesc Skin (measurementDescSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the measurementDesc label. |
| Syntax: | measurementDescSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. measurementDescSkin = “voltmxqfsSknLblMeasurementDescription”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknLblMeasurementDescription”. |

**13.** **PreviousValues Skin (previousValuesSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the previousValues label. |
| Syntax: | previousValuesSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. previousValuesSkin = “voltmxqfsSknLblMeasurementID”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknLblMeasurementID”. |

**14.** **Flex Values (flxValuesSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the Flex Values flexes. |
| Syntax: | flxValuesSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. flxValuesSkin= “voltmxqfsSknFlxValues”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknFlxValues”. |

**15****.** **Label Values (lblValuesSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the Label Values. |
| Syntax: | lblValuesSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. lblValuesSkin= “voltmxqfsSknLblValuesNB”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknLblValuesNB”. |

**16.** **Label Date Time (lblDateTimeSkin)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the Datetime label. |
| Syntax: | previousValuesSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. infocard. lblDateTimeSkin = “voltmxqfsSknLblDate”; |
| **Remarks:** | The default value for the property is “voltmxqfsSknLblDate”. |

## API’s

### initialize

|  |  |
| --- | --- |
| **Description:** | This method is used to create records for the component |
| Syntax: | initialize(data) |
| **Parameters:** | Data (type-json) [Optional] |
| **Return Value:** | None |
| **Example:** | this. view. infocard. initialize (data); |
| **Remarks:** | Keys in the json should always be same. |

# Revision History

App version 1.0.1:

## Limitations

1.Landscape mode is not supported in all channels.

2. If browser height is minimized then UI gets distorted.