Date: 06-Feb-2024

Progress BAR - Check (theme 1) VERsION: 1.1.1

1. **OVERVIEW**

Progress-bar gives instant visual feedback to the user on how far along he/she is in a process. Just drag and drop the component into your app, configure a few properties and that's all- you are ready to go. You can customize the look and feel of the component and change the icons for the states using the component properties. You can also move the progress indicator between the states using the component APIs.

1. **Use case:**

Consider a retail app, where the app updates the status of an order into stages such as: **Items Selected → Order Placed → Payment Confirmed → Items Out for Delivery → Items Delivered.**

1. **Features**

* Ready-to-use component
* Simple and user-friendly UI to keep user updated on his/her progress.
* Determinate Progress bar- supports up to 8 states/steps.
* Easy to customize- use your own icons and color schemes.

1. **Percentage of re-use:**

Approximate 80% 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**

1. **Prerequisites**

Before you start using the Progress Bar - Check (Theme 1) component, ensure the following:

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

1. Volt MX Iris
2. **Platforms Supported**

### Mobile

#### iOS

#### Android

### Tablet & iPad

### PWA

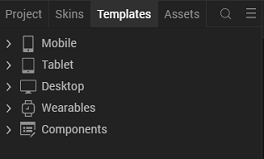
1. **Importing the Component**

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

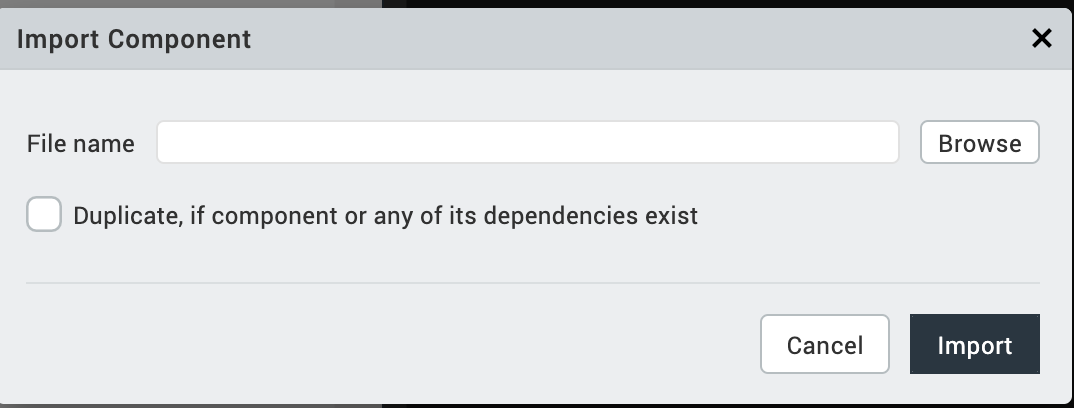
## **To import the** **Progress Bar - Check (Theme 1) component, do the following:**

## Open your app project in Volt MX Iris.

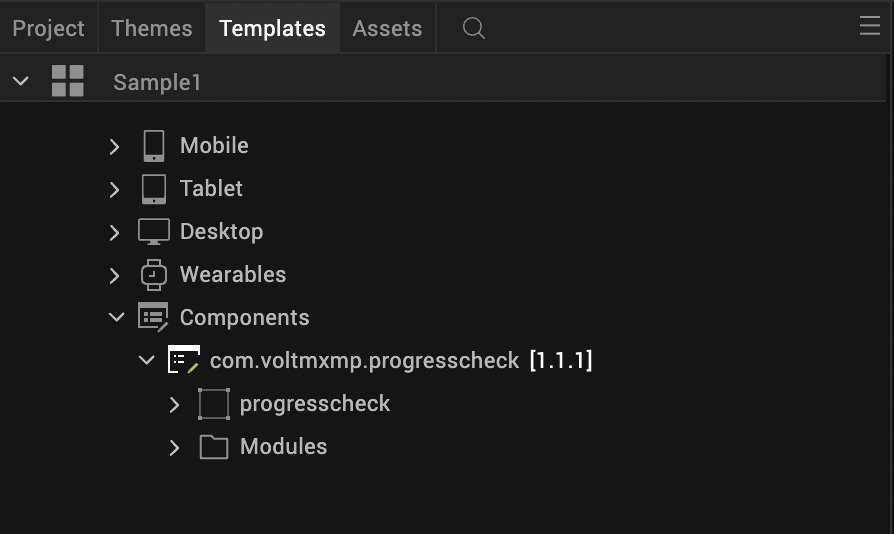
2. In the Project Explorer, click the **Templates** tab.



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



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**

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 Progress Bar - Check (Theme 1) work in real time.

# **References**

## **Dynamic Usage**

You can also add a Progress Bar - Check (Theme 1) 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 a component's Object \*/

var ProgressCheck = new com.voltmxmp.progresscheck(

{

"clipBounds":true,

"height": "11%",

"id": "ProgressCheck",

"isVisible": true,

"left": "0dp",

"top":"0dp",

"width": "100%",

"zIndex": 1

}, {}, {});

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

ProgressCheck.stateCount = "4";

ProgressCheck.bgColor = "4785F4";

ProgressCheck.activeIcon = "voltmxmp\_pck\_active.png";

ProgressCheck.inactiveIcon = "voltmxmp\_pck\_inactive.png";

ProgressCheck.separatorBaseColor = "000000";

ProgressCheck.separatorFillColor = "FFFFFF";

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

this.view.add(ProgressCheck);

/\* Creating the component with the specified properties \*/

this.view.ProgressCheck.createProgressbar();

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.

## **Properties**

## The properties provided on the **Component** tab allow you to customize the UI elements in the **Progress Bar - Check (Theme 1)** component. You can set the properties directly on the **Component** tab or by writing a JavaScript.

#### **1. General**

**1. Number of states**

|  |  |
| --- | --- |
| **Description:** | Specifies the number of states or steps in the Progress Bar. |
| **Syntax:** | stateCount |
| **Type:** | * List Selector * String |
| **Read/Write:** | Read |
| **Remarks:** | * You can write data to this property only during the creation of the component. * The number of states must be between 2 and 8. * If you provide an invalid value, the component throws an exception that you need to handle. |
| **Default Value:** | "4" |
| **Example:** | /\* During creation of the component \*/  this.view.componentID.stateCount = "4"; |

#### **2. Image Properties**

## **i. Active Icon**

|  |  |
| --- | --- |
| **Description:** | Specifies the image that you want to set for the active state of the icon. |
| **Syntax:** | voltmxmp\_pck\_active.png |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | /\* During creation of the component \*/  this.view.componentID.activeIcon = "voltmxmp\_pck\_active.png"; |

## **ii. Inactive Icon**

|  |  |
| --- | --- |
| **Description:** | Specifies the image that you want to set for the inactive state of the icon. |
| **Syntax:** | voltmxmp\_pck\_inactive.png |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | /\* During creation of the component \*/  this.view.componentID.inactiveIcon = "voltmxmp\_pck\_inactive.png"; |

#### **3. Color Properties**

**i. Background Color**

|  |  |
| --- | --- |
| **Description:** | Specifies background color of the component in a Hexadecimal Format. |
| **Syntax:** | bgColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Values:** | You can provide two types of color codes:   * 6-digit Hexadecimal Code. For example: ffffff * 8-digit Hexadecimal Code (with permitted alpha position). For example: ffffff00 |
| **Default Value:** | "4785f4" |
| **Remarks:** | * You can write data to this property only during the creation of the component. * Make sure that you type the Hexadecimal Code without the pound (#) symbol. * If you provide an invalid value, the component throws an exception that you need to handle. |
| **Example:** | /\* During creation of the component \*/  this.view.componentID.bgColor = "4785f4"; |

**ii. Separator Base Color**

|  |  |
| --- | --- |
| **Description:** | Specifies the color of the Separator Line (in Hexadecimal Format) when it is inactive. |
| **Syntax:** | separatorBaseColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Values:** | You can provide two types of color codes:   * 6-digit Hexadecimal Code. For example: ffffff * 8-digit Hexadecimal Code (with permitted alpha position). For example: ffffff00 |
| **Default Value:** | "000000" |
| **Remarks:** | * You can write data to this property only during the creation of the component. * Make sure that you type the Hexadecimal Code without the pound (#) symbol. * If you provide an invalid value, the component throws an exception that you need to handle. |
| **Example:** | /\* During creation of the component \*/  this.view.componentID.separatorBaseColor="000000"; |

**iii. Separator Fill Color**

|  |  |
| --- | --- |
| **Description:** | Specifies the color of the Separator Line (in Hexadecimal Format) when it is **active**. |
| **Syntax:** | separatorFillColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Values:** | You can provide two types of color codes:   * 6-digit Hexadecimal Code. For example: **ffffff** * 8-digit Hexadecimal Code (with permitted alpha position). For example: **ffffff00** |
| **Default Value:** | "ffffff" |
| **Remarks:** | * You can write data to this property only during the creation of the component. * Make sure that you type the Hexadecimal Code without the pound (#) symbol. * If you provide an invalid value, the component throws an exception that you need to handle. |
| **Example:** | /\* During creation of the component \*/  this.view.componentID.separatorFillColor="ffffff"; |

## **Events**

The component invokes events when its corresponding action is performed. You can configure any logic you want the component to perform whenever an event occurs.

You can configure the events directly on the Actions tab or by writing a JavaScript, For more information, refer to [Add Actions](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/working_with_Action_Editor.html) in the Volt MX Iris User Guide.

**i. errorCallback**

|  |  |
| --- | --- |
| **Description:** | Invoked when an error occurs within the component. |
| **Syntax:** | errorCallback(error) |
| **Parameters:** | *error [JSON] :* Information about the error, such as the error code and error message. |
| **Example:** | this.view.componentID.errorCallback = function(error)  {  alert("Error: "+JSON.stringify(error));  }.bind(this); |

## **APIs**

#### **i. goNext**

|  |  |
| --- | --- |
| **Description:** | Moves the progress indicator to the next state. |
| **Syntax:** | goNext() |
| **Parameters:** | None. |
| **Return Value:** | None. |
| **Remarks:** | If you call this API when the progress indicator is at the last state, the component throws an exception. |
| **Example:** | this.view.componentID.goNext(); |

**ii. goPrev**

|  |  |
| --- | --- |
| **Description:** | Moves the progress indicator to the previous state. |
| **Syntax:** | goPrev() |
| **Parameters:** | None. |
| **Return Value:** | None. |
| **Remarks:** | If you call this API when the progress indicator is at the first state, the component throws an exception. |
| **Example:** | this.view.componentID.goPrev(); |

**iii. getCurrentState**

|  |  |
| --- | --- |
| **Description:** | Fetches the current state of the progress indicator. |
| **Syntax:** | getCurrentState() |
| **Parameters:** | None. |
| **Return Value:** | *currentState [Number]* : The index of the current state. |
| **Example:** | var currentState = this.view.componentID.getCurrentState(); |

**iv. createProgressBar**

|  |  |
| --- | --- |
| **Description:** | Creates or refreshes the component. |
| **Syntax:** | createProgressBar() |
| **Parameters:** | None. |
| **Return Value:** | None. |
| **Remarks:** | Use this API to create or refresh the component in the following scenarios:   * After you dynamically add the component to a form. * After you change the width of the component at run time. * After a breakpoint change, to enable [responsive behaviour](https://docs.kony.com/konylibrary/visualizer/visualizer_user_guide/Default.htm#Responsive_Design_8_2.htm). |
| **Example:** | var currentState = this.view.componentID.createProgressBar(); |

## **REVISION HISTORY**

App version 1.1.1

## **Limitations**

* The component does not support landscape mode.
* The component does not provide an option to control the speed of animation.