Date : 09-Feb-22

Basic Login Component (1.1.2)

1. **OVERVIEW**

Basic login component comes pre-configured with Foundry user-store identity service. Basic Login component is a login form that every mobile app needs, to authenticate its users. The component contains a login form with fields such as user name, password, and submit button to authenticate the app's users. The fields in the component are configured with necessary constraints and actions.

A. Use case

You can import the Login component into your app (created in VoltMX iris), and achieve the login screen without developing it from scratch. The Login component also facilitates a set of properties, events, and APIs that helps you customize the fields and the functionality.

 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.

1. **GETTING STARTED**
2. Prerequisites

Before you start using the BasicLogin component, ensure the following:

* [HCL Foundry](https://manage.hclvoltmx.com/)
* Volt MX Iris
1. Platforms Supported
2. Mobile
3. iOS
4. Android
5. Tablets
6. PWA (Biometrics not supported)
7. Importing the app

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

**To import the BasicLogin component, do the following:**

 1.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%22%20/l%20%22add-a-component-to-a-form)

1. **REFERENCES**
	1. Properties

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

Logo Properties :

1.Logo Image

|  |  |
| --- | --- |
| **Description:** | Specifies the image for logo. |
| **Syntax:** | logoSource |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for this property is "reusablelogin\_logo.png". |
| **Example:** | this.view.componentID.logoSource = "imagedrag.png"; |

2.Logo Scale Mode

|  |  |
| --- | --- |
| **Description:** | Specifies the scaling mode of the logo image. |
| **Syntax:** | logoScaleMode |
| **Type:** | Constant |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for this property is "constants.IMAGE\_SCALE\_MODE\_MAINTAIN\_ASPECT\_RATIO". |
| **Example:** | this.view.componentID.logoScaleMode = constants.IMAGE\_SCALE\_MODE\_FIT\_TO\_DIMENSIONS; |

Username Properties :

1.Username Input Mode

|  |  |
| --- | --- |
| **Description:** | Specifies a type of input characters accepted by the Username field. |
| **Syntax:** | usernameInputMode |
| **Type:** | Constant |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for this property is " constant.TEXTBOX\_INPUT\_MODE\_ANY"You can specify any one of the following:* Any: accepts all characters.
* Numeric Only: accepts only numbers.
 |
| **Example:** | this.view.componentID.usernameInputMode = constant.TEXTBOX\_INPUT\_MODE\_NUMERIC; |

2.Username Min Characters

|  |  |
| --- | --- |
| **Description:** | Specifies the minimum number of characters to be entered in the Username field. |
| **Syntax:** | usernameMinimumChar |
| **Type:** | Number |
| **Read/Write:** | Read + Write |
| **Remarks:** | * The default value for this property is "8".
* The component throws an exception if you provide any invalid values.
 |
| **Example:** | this.view.componentID.usernameMinimumChar = 8; |

3.Username Max Characters

Description**:** Specifies the maximum number of characters allowed by the Username field. Using the property, you can limit the user to enter not beyond a particular number of characters.

|  |  |
| --- | --- |
| **Syntax:** | usernameMaxChar |
| **Type:** | Number |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for this property is "15". |
| **Example:** | this.view.componentID.usernameMaxChar = 10; |

4.Username Validation Message

**Description:** Specifies a message to be displayed when the length of the text entered in the Username field is less than the specified minimum length (Username Min Characters).

|  |  |
| --- | --- |
| **Syntax:** | usernameValidationMsg |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.usernameValidationMsg = " Username Too Small"; |

**Remarks:** If no message is defined in the property, Invalid Username message will be displayed by default. If user provides invalid inputs (such as undefined, invalid datatype, usernamemaximimChar is less than usernameMinimumChar property etc.), an exception is thrown. The user needs to handle this exception.

This property renders at runtime.

The default value is "Username Too Small".

Password Properties :

1.Password Input Mode

|  |  |
| --- | --- |
| **Description:** | Specifies a type of input characters accepted by the Password field. Using the property, you can restrict the Password field to accept only required characters. For example, only numbers. |
| **Syntax:** | passwordInputMode |
| **Type:** | Constant |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for this property is " constant.TEXTBOX\_INPUT\_MODE\_ANY".You can specify any one of the following:* Any: accepts all characters.
* Numeric Only: accepts only numbers.
 |
| **Example:** | this.view.componentID.passwordInputMode = constants.TEXTBOX\_INPUT\_MODE\_NUMERIC; |

2.Password Min Characters

|  |  |
| --- | --- |
| **Description:** | Specifies the minimum number of characters to be entered in the Password field. |
| **Syntax:** | passwordMinimumChar |
| **Type:** | Number |
| **Read/Write:** | Read + Write |
| **Remarks:** | * The default value for this property is "8".
* This property renders at runtime.
* The component throws an exception if you provide any invalid values.
 |
| **Example:** | this.view.componentID.passwordMinimumChar = 10; |

3.Password Validation Message

|  |  |
| --- | --- |
| **Description:** | Specifies a message to be displayed when the length of the text entered in the Password field is less than the specified minimum length (Password Min Characters). |
| **Syntax:** | passwordValidationMsg |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | If no message is defined in the property, Password Too Small message will be displayed by default. |
| **Example:** | this.view.componentID.passwordValidationMsg = "Enter Valid Password"; |

Sign-in Properties :

1.Submit Button Text

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed on the Submit button. |
| **Syntax:** | submitButtonText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value of the property is SIGN IN . |
| **Example:** | this.view.componentID.submitButtonText = "Sign In"; |

2.Forgot Password Text

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed on the Forgot Password button. |
| **Syntax:** | forgotPasswordText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value of the property is Forgot Password . |
| **Example:** | this.view.componentID.forgotPasswordText = "Forgot Password?"; |

3.Enable Remember Me

|  |  |
| --- | --- |
| **Description:** | Controls whether or not the Remember Me check box is visible to the user. |
| **Syntax:** | remembermeProperty |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for this property is "true". |
| **Example:** | this.view.componentID.remembermeProperty = true; |

4.Remember Me Unselected Image

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be set as unselected check box for the RememberMe field. |
| **Syntax:** | rememberMeUnSelectedSource |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for the property is "reusable\_checkbox\_unselected.png".Before setting the property:* Ensure that the image file exists in the *workspace\resources\common* directory.
* The file name of the image should not contain any upper case characters.
* Specify the image file name along with the extension.
 |
| **Example:** | this.view.componentID.rememberMeUnSelectedSource = "imagedrag.png"; |

5.Remember Me Selected Image

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be set as selected check box for the RememberMe field. |
| **Syntax:** | rememberMeSelectedSource |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for the property is "reusable\_checkbox\_selected.png".Before setting the property:* Ensure that the image file exists in the *workspace\resources\common* directory.
* The file name of the image should not contain any upper case characters.
* Specify the image s file name along with the extension.
 |
| **Example:** | this.view.componentID.rememberMeSelectedSource = "imagedrag.png"; |

Backend Service Configuration :

Identity Service Name

|  |  |
| --- | --- |
| **Description:** | Specifies the identity service name. |
| **Syntax:** | providerName |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for this property is "userrepository". |
| **Example:** | this.view.componentID.providerName = "userrepository"; |

Skin Section

This section provides details on the properties related to the skin of the **Basic Login** component.

1.Component Background Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the background skin of the component. |
| **Syntax:** | skinBackground |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.skinBackground = "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

2.Top Container Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change top container skin of the component. |
| **Syntax:** | logoBackground |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.logoBackground = "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

3.Login Card Container Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the login card skin of the component. |
| **Syntax:** | loginCardBackground |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.loginCardBackground = "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

4.Username Input Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the username input skin of the component. |
| **Syntax:** | sknUsernameInput |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.sknUsernameInput = "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

5.Username Underline Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the username underline skin of the component. |
| **Syntax:** | sknUsernameUnderline |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.sknUsernameUnderline = "btn\_skin"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

6.Password Underline Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the password underline skin of the component. |
| **Syntax:** | sknPasswordUnderline |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.sknPasswordUnderline = "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

7.Login Button Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the login button skin of the component. |
| **Syntax:** | sknbtnLogin |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.sknbtnLogin= "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

8.Forgot Password Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the forgot password skin of the component. |
| **Syntax:** | sknForgotPassword |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.sknForgotPassword = "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

9.Remember Me Skin

|  |  |
| --- | --- |
| **Description:** | This property enables the consumer to change the remember me skin of the component. |
| **Syntax:** | sknRememberMe |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.sknRememberMe = "skin\_name"; |
| **Remarks:** | Skin assigned to the property is accepted only by same widget type. |

* 1. Events

You can define events to be executed when an action is performed. You can configure the events directly on the Actions tab or by writing a JavaScript. To configure the events on the Actions tab, click the Edit button against each event.

This section provides details about each event that help you define the actions by writing a JavaScript.

1.onErrorCallback Event

**Description:** The event is invoked when an exception or an error occurs.

The event also provides an error object with the error code and message. You can refer to the given table to handle the error based on the error code.

|  |  |
| --- | --- |
| **Syntax:** | onErrorCallback(error) |
| **Remarks:** | The component throws an exception if you do not define this event and you must handle it. |
| **Example:** | this.view.componentID.onErrorCallback = function(error){if (error && error.message) { this.alertPopup(error.message); }else { alert("onerror"); alert('Error: ' + JSON.stringify(error)); }} |

ErrorCode table for the error object in **onErrorCallback** Event.

|  |  |
| --- | --- |
| **Code** | **Message** |
| 90001 | No Internet Connection Available |
| 90002 | Authorization object null - Connect to MF |
| 90003 | Login Success but loginSuccessEvent not defined |
| 90004 | Login Failed but loginFailureEvent not defined |
| 90005 | Username too small (This message changes based on the Username validation message) |
| 90006 | Password too small (This message changes based on the Password validation message) |

2.usernameOnDone

|  |  |
| --- | --- |
| **Description:** | The event is invoked when a user taps the keyboard action key of the Username field. |
| **Syntax:** | usernameOnDone() |
| **Example:** | this.view.componentID.usernameOnDone= function(){ alert( “username on done” );} |

3.loginFailure

|  |  |
| --- | --- |
| **Description:** | The event is invoked when the login is failure. In the event, define actions to be executed if login is failure. |
| **Syntax:** | loginFailureEvent(error) |
| **Example:** | this.view.componentID.loginFailureEvent= function (error){ const msg = error && error.details && error.details.message ? error.details.message : 'Login Failed'; this.alertPopup(msg); }, |

4.loginSuccess

|  |  |
| --- | --- |
| **Description:** | The event is invoked when the login is successful. In the event, define actions to be executed if login is successful. |
| **Syntax:** | loginSuccessEvent(response) |
| **Example:** | this.view.componentID. loginSuccessEvent = function(response){alert(“Login Success” );} |

5.passwordOnDone

|  |  |
| --- | --- |
| **Description:** | The event is invoked when a user taps the keyboard action key of the Password field. |
| **Syntax:** | passwordOnDone() |
| **Example:** | this.view.componentID.passwordOnDone= function(){alert( “Password on done”); } |

6.remembermeOnSelection Event

|  |  |
| --- | --- |
| **Description:** | The event is invoked when the Remember Me check box is toggled. In the event, define actions to be executed when the Remember Me check box is selected and unselected. |
| **Syntax:** | remembermeOnSelection() |
| **Example:** | this.view.componentID.remembermeOnSelection = function(){ alert( “Remember me is clicked” ); } |

7.submitOnClick

|  |  |
| --- | --- |
| **Description:** | The event is invoked when the Submit button is clicked/tapped. In the event, define actions to be executed when the Submit button is clicked. |
| **Syntax:** | submitOnClick() |
| **Example:** | this.view.componentID.submitOnClick= function(){ alert( “Login button is clicked” ); } |

* 1. APIs

The following are the APIs pertaining to the Basic Login component.

1.getPassword

The API fetches the text specified in the Password field.

**Syntax**

getPassword()

**Parameters**

None

**Return Value**

*Password [String]:*
Returns the text entered in the Password field.

**Example**

var pwd = this.view.componentID.getPassword();

if (pwd.length() < 4) {

 alert("Invalid Password Entered. Please try again.");

}

2.getUsername

The API fetches the text specified in the Username field.

**Syntax**

getUsername()

**Parameters**

None

**Return Value**

*Username [String]:*
Returns the text entered in the Username field.

**Example**

var username = this.view.componentID.getUsername();

if (!this.validateUsername(username)){

alert("Invalid user name. Please try again.");}

1. **REVISION HISTORY**

App version 1.1.2