Date: 25 May 2023

google Sheets Data Driven App

VERSION: 1.1.1

# **Overview**

The Worksheet List Collection Library provides you with all the components required to make a Google Sheets Data driven app.

The components included in the collection library are:

* Google Login
* Worksheet List
* Sheet TextBox
* Sheet ListBox
* Sheet Date

To access the Google Sheets data from the app, the user first need to login with his/her Google credentials.

You can use the **Google Login** component to authenticate the user.

 

**Figure 1: Google Login Screen**

You can use the **Worksheet List** component to map a list of records present in Google Sheets table or a range to a **Master-Detail** interface on your app.

 
**Figure 2: Worksheet List Master View**

 
**Figure 3: Worksheet List Detail View**

## **Use case:**

With the app, users can efficiently access, add and update Google Sheets data anytime, anywhere.

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

## 80-90%

The **Master-View** lists all the **worksheet rows** where each row displays four elements or fields from a Google Sheet record. The **Add** floating action button in the **Master-View** allows the user to add a new record to the Google Sheet. You can specify the mapping between worksheet row elements and Google Sheet columns in the Iris’s **Properties** panel.

By selecting a worksheet row in the **Master-View**, the user can navigate to a **Detail-View** which displays the rest of the fields of the selected Google Sheet record. The **Edit** button in the **Detail-View** allows the user to update a record.

The **Master-View** also contains a **Vertical Bar Chart** which you can use to create a more visually informative dashboard. You can turn off the chart from the Iris's **Properties** panel if you don't want to use it.

You can customize the **Detail-View** by dragging and dropping components into the **flxDetail** flex container.

You can add the following components into the **Detail-View**

* Sheet Text-box
* Sheet List-box
* Sheet Date

 
**Figure 4: Detail-View Components**

 
**Figure 5: Sheet List-box**

 
**Figure 6: Sheet Date**

After adding any component into the **Detail-View**, you can map it to the appropriate Google Sheet columns from the Iris's **Properties** panel.

# **Getting Started**

## **Prerequisites**

Before you start using the Worksheet List Collection Library, ensure the following:

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

• Volt MX Iris

## **Platforms Supported**

### Mobile

#### iOS

#### Android

## **Importing the app**

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

##  **To import the** Worksheet List Collection Library **, do the following:**

## Open your app project in Volt MX Iris.

1. In the Project Explorer, click the **My Libraries** tab.

1. Click **Hamburger**, and then select **Import Library**. The **Import Library** dialog box appears.
2. Click **Browse** to navigate to the location of the library, select the library, and then click **Import**. The library and its associated widgets and modules are added to your project.

## A screenshot of a computer  Description automatically generated

Once you have imported a library 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%5Cl%20%22add-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 Worksheet List Collection Library work in real time.

## Note:

## Ensure that you log in before you add the component to the form so that the respective Foundry App gets linked with your app.

# **References**

## **Dynamic Usage**

You can also addWorksheet List Collection Library component dynamically. For example, to add a **Worksheet List** component:

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 worksheetlist = new com.voltmxmp.worksheetlist(

{

 "autogrowMode": voltmx.flex.AUTOGROW\_NONE,

 "clipBounds": true,

 "enableCache": true,

 "height": "100%",

 "id": "worksheetlist",

 "isVisible": true,

 "layoutType": voltmx.flex.FREE\_FORM,

 "left": "0%",

 "masterType": constants.MASTER\_TYPE\_USERWIDGET,

 "skin": "slFbox",

 "top": "0dp",

 "width": "100%"

}, {}, {});

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

worksheetlist.Left = "0%";

worksheetlist.spreadsheetid = "sheet\_id";

worksheetlist.range = "Sheet1!A1:G";

worksheetlist.column1 = "A";

worksheetlist.column2 = "B";

worksheetlist.column3 = "C";

worksheetlist.column4 = "D";

worksheetlist.chartrange = "Sheet2!A1:B4";

/\*adding component to flex\*/

this.add(worksheetlist);

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

# **Authenticating via Google**

To access Google Sheets data from your app, you need an **OAuth 2.0 token** from Google. To get this token, you can use the Identity Services on HCL Foundry.

This collection library contains a GoogleSheets service on HCL Foundry that has its own Identity service, **googleSheetsAuth**. You can use this service to obtain the token, but you need to manually implement the **user-login flow** in the app.

We recommend using the **Google Login** component, which is bundled with the collection library. It has its own UI and it takes care of the Identity Service calls from the client side.

# **Configuring Google Login**

**Google Login** is a standalone component on Foundry. You can find more information about the specific configurations in the **Documentation** section of the [Google Login](https://community.kony.com/marketplace/items/google-login) component.

The **Google Login** component comes with the **reusableGoogleLogin** Identity Service which should be linked with your Integration Service, **konyMpGoogleSheets** on HCL Foundry.

You can link the two services by following the given steps:

1. Login to your VoltMX account.
2. Open the HCL Foundry app linked with your project.
3. Click on the **Integration** tab under **Configure Services**.
4. Select the **konyMpGoogleSheets** service.
5. Click on the drop down menu under **Identity Service for Backend Token** and select the **reusableGoogleLogin** service.
6. For testing on SPA, clone the existing Identity service. In the **Advanced** section, under **Redirect URL on successful authentication**, select **Any URL**.



Identity service for SPA is **reusableGoogleLoginSPA.**

In Integration service, select identity service for backend token as per requirement.

You also need to update the scope in the Identity Service. You can achieve this by following the given steps:

1. Click on the **Identity** Tab under **Configure Services**.
2. Select the **reusableGoogleLogin** service.
3. Add the two given values under **Scope**, use **Space** as a separator.
	1. https://www.Googleapis.com/auth/spreadsheets.readonly
	2. <https://www.Googleapis.com/auth/spreadsheets>

# **Worksheet List Master-View**

## **Properties**

## The properties provided on the **Component** tab allows you to customize the elements in the Worksheet List Collection Library. 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**

**1. Spreadsheet Id**

|  |  |
| --- | --- |
| **Description:** | Specifies the id of Google sheet from where data is required. |
| Syntax: | spreadsheetid |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.spreadsheetid = "spreadsheet\_id"; |
| **Note:** | The ID of the Google Sheet is in the URL, as shown belowFor more information you can refer to the [Spreadsheet ID Concept](https://developers.google.com/sheets/api/guides/concepts#spreadsheet_id). |

**2. Data Range in A1 Notation**

|  |  |
| --- | --- |
| **Description:** | Specifies the range of Google sheet for fetching data. It should be in A1 Notation. |
| Syntax: | range |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.range = "Sheet1!A1:D8"; |
| **Note:** | The A1 notation in the given example, **Sheet1!A1:D8**, refers to a group of cells in the spread sheet.For more information you can refer to the [A1 Notation Concept](https://developers.google.com/sheets/api/guides/concepts#a1_notation). |

 **3. Left**

|  |  |
| --- | --- |
| **Description:** | Specifies the **left** value of the detail container. |
| Syntax: | Left |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value for the property is “Use Google Maps”. |

**4. Add Row Button Visibility**

|  |  |
| --- | --- |
| **Description:** | Specifies the visibility of floating action button on the bottom right. |
| Syntax: | isVisibleAddRow |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.isVisibleAddRow = true; |

**5****. Header Name**

|  |  |
| --- | --- |
| **Description:** | Specifies the header name of the component. |
| Syntax: | headerName |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.headerName = "Leads"; |

###  **Worksheet Rows**

**6.** **Element1 Column**

|  |  |
| --- | --- |
| **Description:** | Specifies the column name of Google sheet which should be mapped to element1 in the list row. |
| Syntax: | column1 |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.column1="A"; |

**7. Element2 Column**

|  |  |
| --- | --- |
| **Description:** | Specifies the column name of Google sheet which should be mapped to element2 in the list row. |
| Syntax: | Column2 |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.column2="B"; |

**8. Element3 Column**

|  |  |
| --- | --- |
| **Description:** | Specifies the column name of Google sheet which should be mapped to element3 in the list row. |
| Syntax: | column3 |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.column3="C"; |

**9. Element4 Column**

|  |  |
| --- | --- |
| **Description:** | Specifies the column name of Google sheet which should be mapped to element4 in the list row. |
| Syntax: | column4 |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.column4="D"; |

**Note:**All the **Worksheet Rows** properties only support values from A to Z.

### **Chart Properties**

**10. Chart Visibility**

|  |  |
| --- | --- |
| **Description:** | Specifies whether chart should be visible or not. |
| Syntax: | chartVisibility |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.chartVisibility = true; |

**11. Data Range in A1 Notation**

|  |  |
| --- | --- |
| **Description:** | Specifies the range of Google sheet for fetching data for the chart. It should be in A1 Notation. |
| Syntax: | chartrange |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.chartrange = "Sheet1!A1:D8"; |
| **Note:** | The A1 notation in the given example, **Sheet1!A1:D8**, refers to a group of cells in the spread sheet.For more information you can refer to the [A1 Notation Concept](https://developers.google.com/sheets/api/guides/concepts#a1_notation). |

### **Additional Properties:**

You can access additional configurations for the **Vertical Bar Chart** by following the given steps:

1. Go to the **Templates** tab in the **Project Explorer** panel of the Iris.
2. Expand **worksheetlist** -> **flxList** -> **flxChart**. Select the **verticalbar** component.
3. Modify the required properties under the **Component** tab in the **Properties** panel on the right side of Iris.

You can find more information about the specific configurations in the **Documentation** section of the [Vertical Bar Chart - Distributed Series](https://community.kony.com/marketplace/items/vertical-bar-chart-%20-distributed-series) component.

### **Events:**

**12. onClickLogOut**

|  |  |
| --- | --- |
| **Description:** | This event will be invoked when user clicks on the log out button in the header. |
| Syntax: | onClickLogout() |
| **Example:** | this.view.componentID.onClickLogOut = function(){ alert("Logout Clicked!");}.bind(this); |

### **APIs:**

**13. fetchAndSetData**

This API enables the user to fetch the data from google sheet and set it to the Worksheet List.

|  |  |
| --- | --- |
| Syntax: | fetchAndSetData() |
| **Parameters:** | None |
| **Return value:** | None |
| **Example:** | this.view.componentID.fetchAndSetData(); |

# **Worksheet List Detail-View**

By default, the **Detail-View** is not visible on the Iris canvas. To see the **Detail-View**, you can either turn on the **BVR mode** or set the **Left** property to **0%**.

**Detail-View** is in an empty flex container, **flxDetail**. You can drag and drop three types of components into it:

* [Sheet Textbox](https://docs.kony.com/marketplace/V8Marketplace/Content/Marketplace/worksheetlistlibrary.htm#SheetTextBox)
* [Sheet Listbox](https://docs.kony.com/marketplace/V8Marketplace/Content/Marketplace/worksheetlistlibrary.htm#SheetListBox)
* [Sheet Date](https://docs.kony.com/marketplace/V8Marketplace/Content/Marketplace/worksheetlistlibrary.htm#SheetDate)

These components are available in the collection library.

## **SheetTextBox**

A **SheetTextBox** component fetches data from the specified Google Sheet column and displays it in a Textbox.

### **General Properties**

**14. Label Text**

|  |  |
| --- | --- |
| **Description:** | Specifies the text for the header of the textbox. |
| Syntax: | labelText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.labelText ="NAME"; |

**15. Textbox Placeholder**

|  |  |
| --- | --- |
| **Description:** | Specifies the placeholder value for the textbox. |
| Syntax: | tbxPlaceholder |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.tbxPlaceholder = "John Smith"; |

**16. Column Name**

|  |  |
| --- | --- |
| **Description:** | Specifies the column of the Google sheet to which the sheet textbox component should be mapped. |
| Syntax: | columnName |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.columnName = "A"; |

## **SheetListBox**

A **SheetListBox** component fetches data from the specified column range of the Google Sheet and displays them in a drop down list in **Edit mode**.

### **General Properties**

**17. Title Text**

|  |  |
| --- | --- |
| **Description:** | Specifies the text for the title of the ListBox component. |
| Syntax: | textTitle |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.textTitle = "Status"; |

**18. ListBox Visibility**

|  |  |
| --- | --- |
| **Description:** | Specifies the visibility for the ListBox component. |
| Syntax: | isVisiblelistbox |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.isVisiblelistbox = false; |

**19. Placeholder**

|  |  |
| --- | --- |
| **Description:** | Specifies the placeholder of the ListBox component. |
| Syntax: | placeholder |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.placeholder = "open"; |

**20. Data Column Name**

|  |  |
| --- | --- |
| **Description:** | Specifies the column that corresponds to the data that should be fetched and mapped to the ListBox component. |
| Syntax: | columnName |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.columnName = "D"; |

**21. Column Range for ListBox in A1 Notation**

|  |  |
| --- | --- |
| **Description:** | Specifies the range of the columns from which the ListBox options should be fetched. The app user sees these options in the drop down list that appears in the **Edit mode**. It should be in A1 Notation. |
| Syntax: | columnRange |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.columnRange = "Sheet20!A1:N7"; |
| **Note:** | The A1 notation in the given example, **Sheet20!A1:N7**, refers to a group of cells in the spread sheet.For more information you can refer to the [A1 Notation Concept](https://developers.google.com/sheets/api/guides/concepts#a1_notation). |

### **SheetListBox APIs**

**22. fetchAndSetListBoxData**

|  |  |
| --- | --- |
| **Description:** | This API fetches the ListBox **options** from the Google sheets based on the specified [Column Range](https://docs.kony.com/marketplace/V8Marketplace/Content/Marketplace/worksheetlistlibrary.htm#listrange), and sets them to the ListBox. |
| Syntax: | fetchAndSetListBoxData() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Example:** | this.view.componentID.fetchAndSetListBoxData(); |

**23. getComponentType**

|  |  |
| --- | --- |
| **Description:** | This API returns the type of the component. It returns **"listBoxComponent"** for this component. |
| Syntax: | getComponentType() |
| **Parameters:** | None |
| **Return Value:** | *listBoxComponent [String]:* *The component type for the SheetListBox Component.* |
| **Example:** | var componentType = this.view.componentID.getComponentType(); |

**24. getLabelData**

|  |  |
| --- | --- |
| **Description:** | This API returns the **Label** data of the component. It returns the label data as a String. |
| Syntax: | getLabelData() |
| **Parameters:** | None |
| **Return Value:** | *label text [String]:* *The text shown in the component's label.* |
| **Example:** | var lblData = this.view.componentID.getLabelData(); |

**25. setLabelData**

|  |  |
| --- | --- |
| **Description:** | This API sets the **Label** data of the component. You should pass the data as a string parameter. |
| Syntax: | setLabelData("text") |
| **Parameters:** | *text [String]:* *The text shown in the component's label.* |
| **Return Value:** | None |
| **Example:** | this.view.componentID.setLabelData("open"); |

**26. getSpreadsheetId**

|  |  |
| --- | --- |
| **Description:** | This API returns the Spreadsheet ID used to fetch data from. It returns the Spreadsheet ID as a string. |
| Syntax: | getSpreadsheetId() |
| **Parameters:** | None |
| **Return Value:** | *Spreadsheet ID [String]:* *The Spreadsheet ID of the Google Sheet used to fetch the data.* |
| **Example:** | var sheetid = this.view.componentID.getSpreadsheetId(); |

**27. setSpreadsheetId**

|  |  |
| --- | --- |
| **Description:** | This API sets the value of the Spreadsheet ID to fetch the data from. You should pass the Spreadsheet ID as a String parameter. |
| Syntax: | setSpreadsheetId("spreadsheet ID") |
| **Parameters:** | *Spreadsheet ID [String]:* *The Spreadsheet ID of the Google Sheet used to fetch the data.* |
| **Return Value:** | None |
| **Example:** | this.view.componentID.setSpreadsheetId("4e5rvgbhefunajk879dfnhjiec"); |

**28. getSelectedValue**

|  |  |
| --- | --- |
| **Description:** | This API returns the selected value from the list box. It returns the value as a String. |
| Syntax: | getSelectedValue() |
| **Parameters:** | None |
| **Return Value:** | *Selected Value [String]:* *The selected value from the drop down list.* |
| **Example:** | var listvalue = this.view.componentID.getSelectedValue(); |

**29. resetSelectedValue**

|  |  |
| --- | --- |
| **Description:** | This API resets the selected value to an **empty string**. |
| Syntax: | resetSelectedValue() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Example:** | this.view.componentID.resetSelectedValue(); |

## **SheetDate**

A **SheetDate** component fetches a date from the specified column of the Google Sheet and displays them in am **mm/dd/yyyy** format. It displays the date on a Calendar Widget in **Edit mode**.

### **General Properties**

**30. Title Text**

|  |  |
| --- | --- |
| **Description:** | Specifies the text for the title of the date component. |
| Syntax: | textTitle |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.textTitle = "Estimated Date"; |

**31. Calendar Widget Visibility**

|  |  |
| --- | --- |
| **Description:** | Specifies the visibility of the calendar widget. |
| Syntax: | isVisibleCalendar |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.isVisibleCalendar = true; |

**32. Calendar Placeholder**

|  |  |
| --- | --- |
| **Description:** | Specifies the placeholder of the calendar widget. |
| Syntax: | placeholderCal |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.placeholderCal = "02/01/2018"; |

**33. Column Name**

|  |  |
| --- | --- |
| **Description:** | Specifies the column that corresponds to the data that should be fetched and mapped to the sheet date component. |
| Syntax: | columnName |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.columnName = "D"; |

### **SheetDate APIs**

**34. getComponentType**

|  |  |
| --- | --- |
| **Description:** | This API returns the type of the component. It returns **"sheetDateComponent"** for this component. |
| Syntax: | getComponentType() |
| **Parameters:** | None |
| **Return Value:** | *sheetDateComponent [String]:* *The component type for the SheetDate component.* |
| **Example:** | this.view.componentID.getComponentType(); |

**35. getLabelData**

|  |  |
| --- | --- |
| **Description:** | This API returns the **Label** data of the component. It returns the label data as a String. |
| Syntax: | getLabelData() |
| **Parameters:** | None |
| **Return Value:** | *label text [String]:* *The text shown in the component's label.* |
| **Example:** | var lblData = this.view.componentID.getLabelData(); |

**36. setLabelData**

|  |  |
| --- | --- |
| **Description:** | This API sets the **Label** data of the component. You should pass the data as a string parameter. |
| Syntax: | setLabelData("text") |
| **Parameters:** | *text [String]:* *The text shown in the component's label.* |
| **Return Value:** | None |
| **Example:** | this.view.componentID.setLabelData("open"); |

**37. getSelectedValue**

|  |  |
| --- | --- |
| **Description:** | This API returns the selected date from the calendar. It returns the date as a String. |
| Syntax: | getSelectedValue() |
| **Parameters:** | None |
| **Return Value:** | *Selected Date [String]:* *The selected in the calendar widget.* |
| **Example:** | var userDate = this.view.componentID.getSelectedValue(); |

**38. resetSelectedValue**

|  |  |
| --- | --- |
| **Description:** | This API resets the selected value to an **empty string**. |
| Syntax: | resetSelectedValue() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Example:** | this.view.componentID.resetSelectedValue(); |

**39. setSelectedDate**

|  |  |
| --- | --- |
| **Description:** | This API sets the selected date to the widget. You should pass this date as a String parameter. |
| Syntax: | setSelectedDate("dateString") |
| **Parameters:** | *date [String]:* *The date should be set on the calendar widget.* |
| **Return Value:** | None |
| **Example:** | this.view.componentID.setSelectedDate("12/01/2017"); |

# **Publishing the App in HCL Foundry**

After adding the **Worksheet List** component to your app and configuring the necessary configurations, you must publish the app to HCL Foundry.

# **Revision History**

App version 1.1.0:

## **Limitations**

1. Landscape mode is not supported.

2. If browser height is minimized UI gets distorted.