

SAP Fiori Mobile (v2.1.x)

Introduction

What are hybrid mobile apps?

Hybrid mobile apps are web applications wrapped up their own browser which can run on any of the mobile platforms (Android, iOS, Windows etc.). The big advantage of a hybrid app is that it has a single code base, whereas a native app must be re-written for each platform.

Apache [Cordova](#) is an open-source framework for developing hybrid apps. To use native features (beyond those already available via html5,) apps can make use of plugins. These plugins handle the platform-specific code and are available for single or multiple platforms. Plugins are available to access the GPS, accelerometer or camera, for example.

SAP have written their own plugins, branded as [Kapsel](#), to handle things like logon and offline OData. These bring an enterprise flavour to hybrid apps.

What is Fiori Mobile?

Until recently SAP's recommended approach was to use the Hybrid Application Toolkit (or HAT) in order to build hybrid apps. The app build took place on the developer's PC or Mac. The HAT gained a reputation for being tricky to set up and that hindered the use of hybrid apps within the SAP ecosystem.

Fiori Mobile has at its core a cloud-build service, so there is no need to install the HAT. We can trigger the build from Web IDE. We specify the (already deployed) Fiori app(s) that we want to package and after a few minutes we can download the .apk (for Android) and .ipa (for iOS) files ready to be installed on mobile devices.

Does it work?

The cloud build service works well. It typically build for both platforms in about 8 minutes without issues. There have only been a couple of occasions when builds failed due to technical problems.

There are some facets to the technology which don't feel mature yet. One area is the offline OData features. When using Fiori Mobile things work differently to the more conventional hybrid and native apps. One example would be the destinations, because Fiori Mobile apps use a generated destination which reference the SAP Cloud Platform (CP) portal service.

My point is not that offline OData seems immature, rather that the offline features can be tricky when used with Fiori Mobile apps. We are still receiving assistance from SAP to get the offline features of our app optimised. This is a shame, because shared data and the handling of delta requests, for example, are two features which justify the 'middleware' approach to offline that SAP have taken.

Authentication has also been a challenge. On a previous project I used the [Fiori Client](#) in conjunction with SAP Cloud Identity (now officially [SAP Cloud Platform Identity Authentication](#)). The Fiori Client stored the username and password on the device so that the user didn't need to enter them every time their session timed out. We haven't achieved that yet with our Fiori Mobile app.

Account and Services

If you have production account browse on [HCP Cockpit](#) and log in. If you use trial account log in [here](#)

As we said before there is two ways to build SAP Fiori mobile application:

1. Using **Fiori Mobile Service**. Note that this service have been discontinued from March 31, 2018. Strategic features available in the mobile service for SAP Fiori have been incorporated into SAP Cloud Platform mobile service for development and operations. However you can use this service if you have old account that already has enabled that service. Using this service require two other services to be enabled: **Portal Service** and **SAP Web IDE Full-Stack Service**.

☰

SAP Cloud Platform Cockpit

Overview

Applications >

Services

Solutions

SAP HANA / SAP ASE >

Connectivity >

Security >

Repositories >

Usage Analytics

Members

Platform Roles

Useful Links

Internet of Things

Remote Data Sync

Enabled

Synchronize data between remote databases and a cloud SAP HANA database.

Mobile

Mobile Services, preview

Enabled

Run integration and regression tests and explore new mobile features. NOT FOR PRODUCTIVE USE.

Runtimes & Containers

HTML5 Applications

Enabled

Develop and run HTML5 applications in a cloud environment.

User Experience

Portal

Enabled

Create role based, multi-channel sites to access business apps and content.

Services

Solutions

SAP HANA / SAP ASE >

Connectivity >

Security >

Repositories >

Usage Analytics

Members

Platform Roles

Remote Data Sync

Enabled

Synchronize data between remote databases and a cloud SAP HANA database.

Mobile

Mobile Services, preview

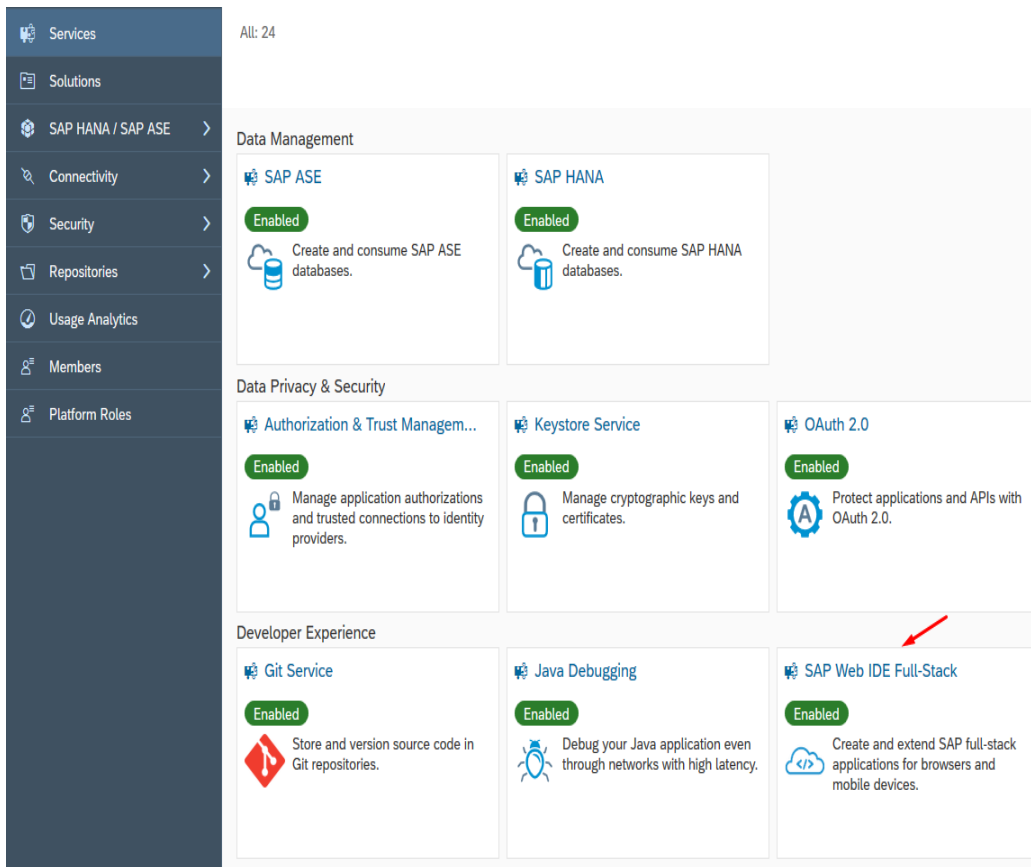
Enabled

Run integration and regression tests and explore new mobile features. NOT FOR PRODUCTIVE USE.

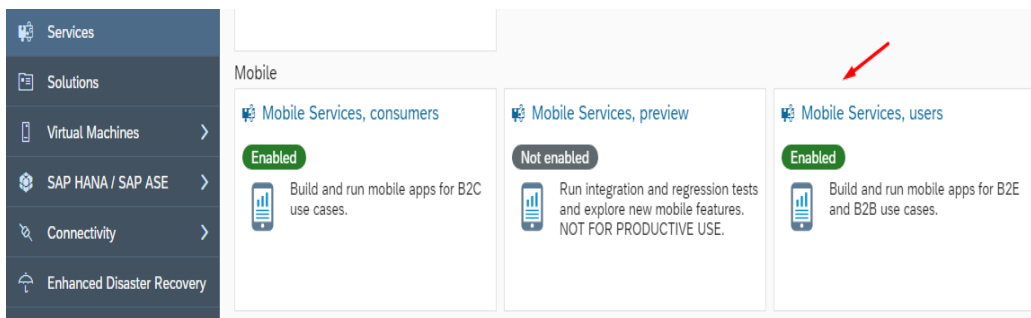
SAP Fiori Mobile

Enabled

Optimize, build, manage, and monitor SAP Fiori apps on mobile devices.



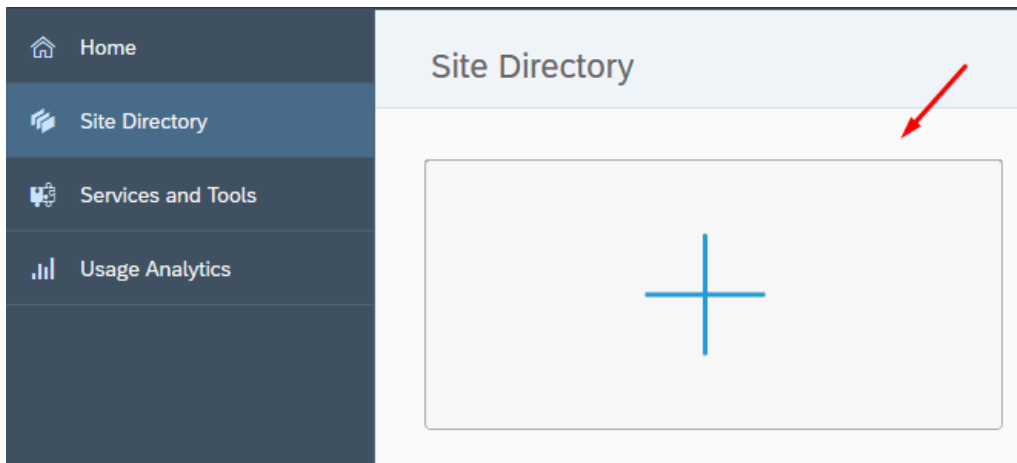
- Using **HAT** (Hybrid Application Toolkit) extension in **SAP Web IDE Full-Stack Service**. Using this extension require **Mobile Services** to be enabled.



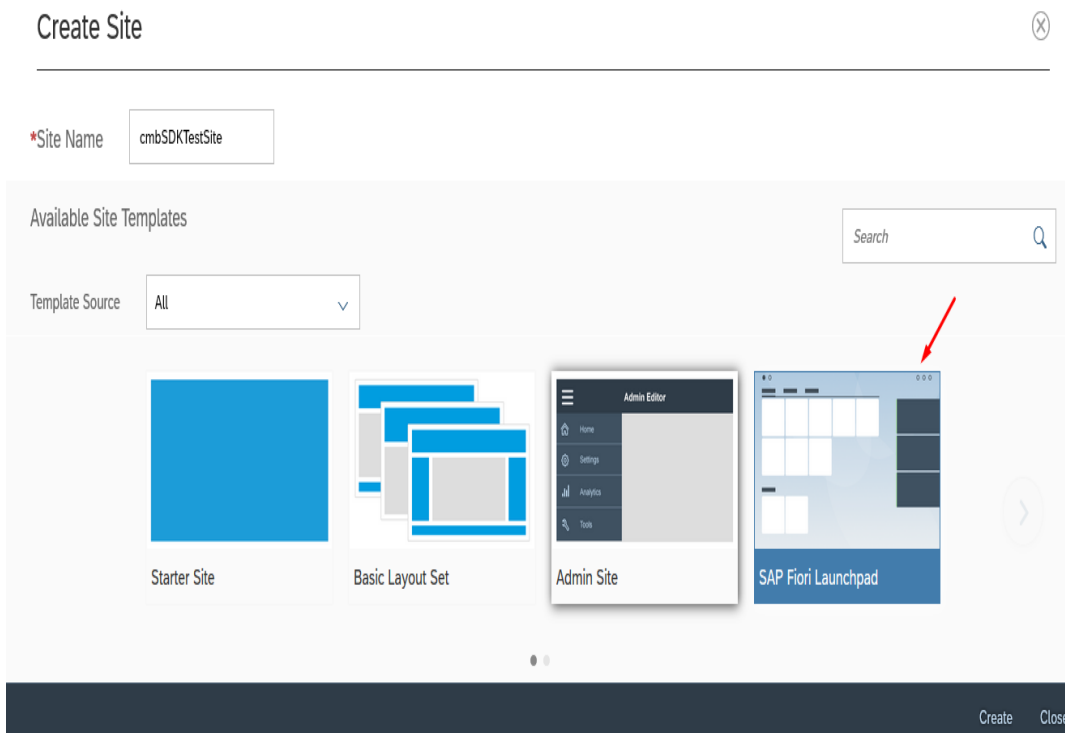
Create Fiori Launchpad

If we use SAP Fiori Mobile Service to build mobile application first we need to create and publish site where application will be deployed. Otherwise if we use HAT skip this step and continue with Create Fiori Project

Go to **Portal Service** and create new Site Directory



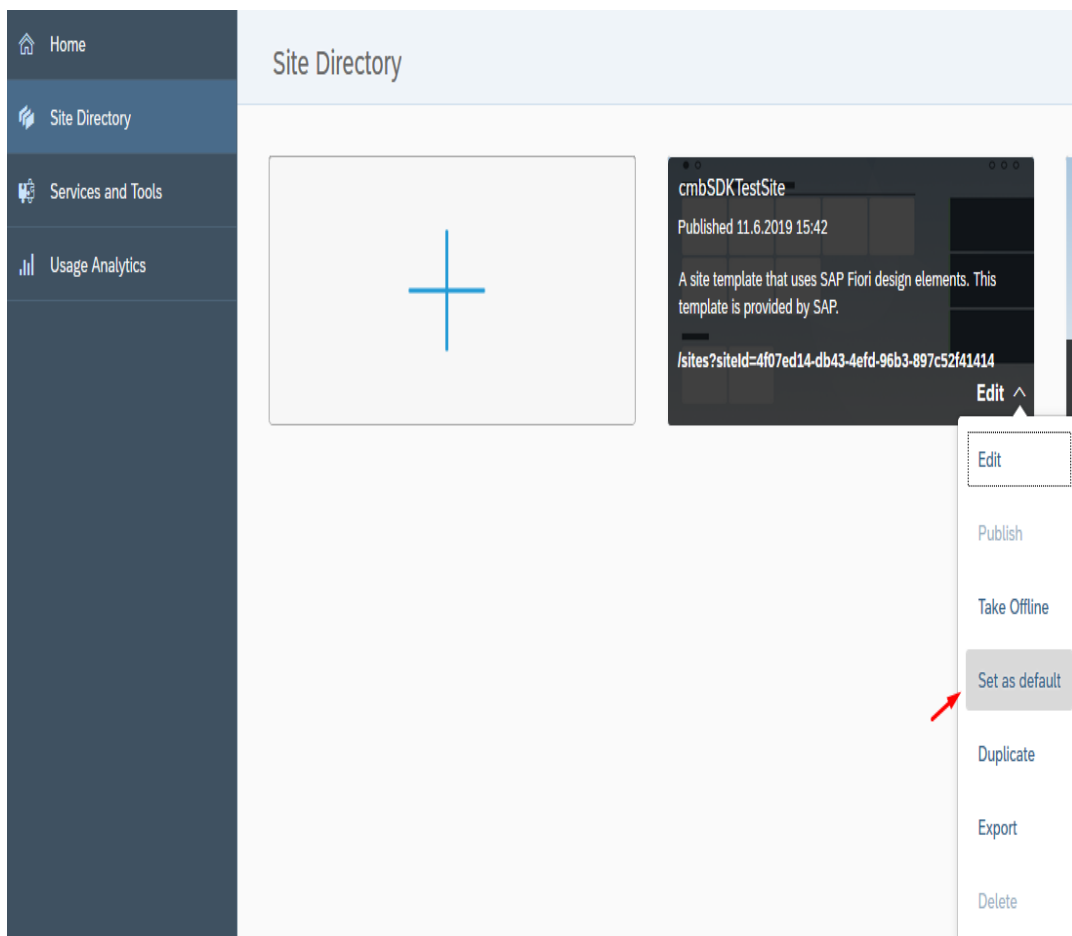
A popup will be displayed. Enter your **Site Name**, choose **SAP Fiori Launchpad** as Template Source and click Create



Following this you will be redirected to SAP Fiori Configuration Cockpit. Close the tab and go back to Admin Space for Portal Service.

Now we need to Publish this Site and Set as default.

The screenshot displays the Cognex Site Directory interface. On the left is a dark sidebar with navigation links: Home, Site Directory, Services and Tools, and Usage Analytics. The main area is titled 'Site Directory' and contains a large light gray box with a blue plus sign for adding new sites. To the right, a site card for 'cmbSDKTestSite' is shown. The card includes the text 'Offline 11.6.2019 15:40' and a description: 'A site template that uses SAP Fiori design elements. This template is provided by SAP.' An 'Edit' button with a dropdown arrow is located at the bottom right of the card. A context menu is open, listing the following actions: Edit, Publish (highlighted with a red arrow), Take Offline, Set as default, Duplicate, Export, and Delete.



Create Fiori Project

Once we create and publish cmbSDKTestSite we can create our Fiori project (if we use Fiori Mobile Service to build mobile application, otherwise if we use HAT we don't need to publish site on Launchpad and creating Fiori project will be our first step).

Go to Services tab again, open SAP Web IDE Full-Stack service, and click on Go to Service



Service: SAP Web IDE Full-Stack - Overview

Enabled

Service Description

With SAP Cloud Platform Web IDE Full-Stack, you can easily develop, test, and upgrade apps for business users. Create applications rapidly and deliver an outside world or build SAP Fiori apps, create new SaaS solutions, extend SAP S/4HANA cloud, and build IoT apps for SAP Leonardo, using the UI development toolkit for HTML5, the SAP HANA toolset, and Java programming language and technologies. Integrate with SAP Cloud Platform, such as SAP Fiori Cloud apps, Git integration, mobile services, IoT services, and more.

Take Action

[Configure Service](#)

[Logs](#)

[Go to Service](#)



A new tab will be opened. From here we will start to create our Fiori project.

- Click on New Project from Template

General Information

Register to SAP Fiori Launchpad

General Information

*Provider Account	portal (flprnc)							
*Application Name	MyFirstFioriApp.MyFirstFioriApp.MyFirstFioriApp							
Description	<input type="text"/>							
Intent	<table> <thead> <tr> <th><input type="checkbox"/></th> <th>Semantic Object</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>MyFirstFioriApp</td> <td>Display</td> </tr> </tbody> </table>		<input type="checkbox"/>	Semantic Object	Action	<input type="checkbox"/>	MyFirstFioriApp	Display
<input type="checkbox"/>	Semantic Object	Action						
<input type="checkbox"/>	MyFirstFioriApp	Display						

[More information on intent properties.](#)

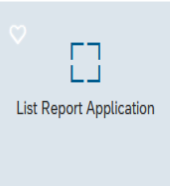
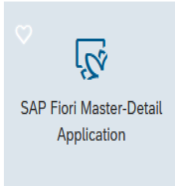
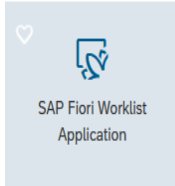
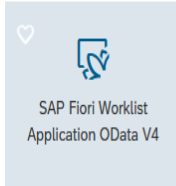
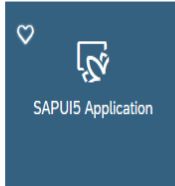
- Choose SAPUI5 Application from Template Selection

New SAPUI5 Application

Template Selection

Search Environment Category Sort By SAPUI5 Version

Neo Featured Sort by name SAP Innovation (1.66)

 List Report Application
  SAP Fiori Master-Detail Application
  SAP Fiori Worklist Application
  SAP Fiori Worklist Application OData V4
  SAPUI5 Application

Creates a SAPUI5 project including an optional view. You can choose one of the following view types: XML, JSON, JavaScript, and HTML. For more information, see the relevant [Fiori Design Guidelines](#).

- Set your Project Name and Namespace

New SAPUI5 Application

Basic Information

Project Name* MyFirstFioriApp

App Descriptor Data

Namespace* MyFirstFioriApp

- Choose the View Type and set the View Name. In our sample we will use XML View. Click Finish and your Fiori project is created.

New SAPUI5 Application

Template Customization

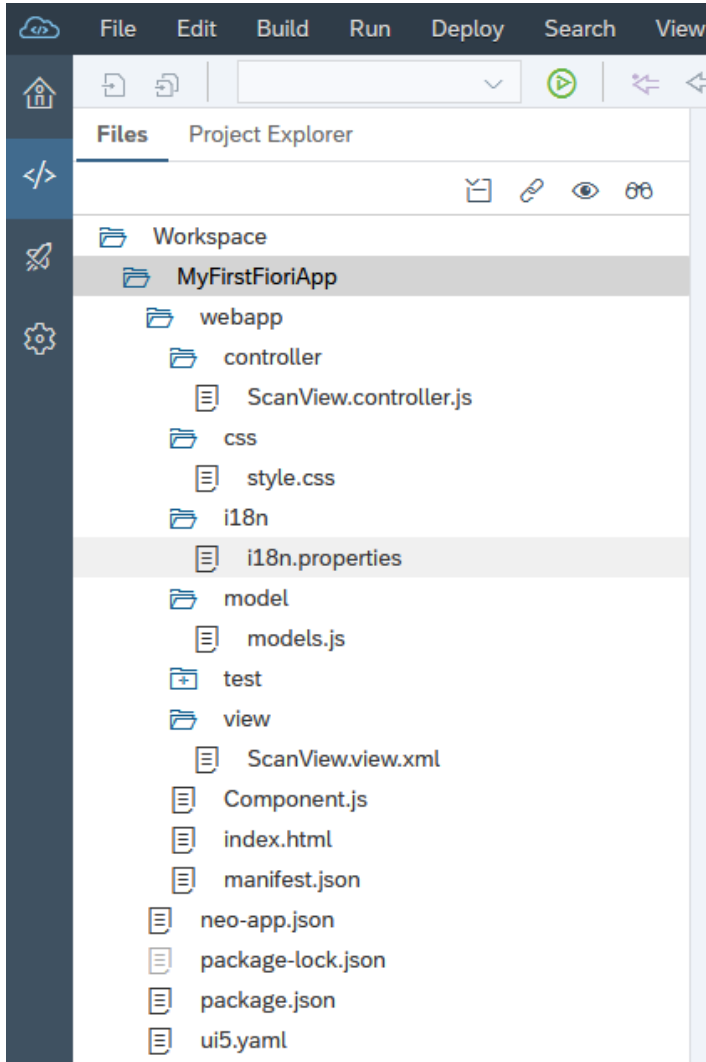
Initial View Details

View Type* XML

View Name ScanView

How to use cmbSDK cordova plugin

In the SAP Web IDE Workspace we can see all files that our Fiori project includes.



Open *Component.js* file.

Component.js is the first point of our application, we can say that it serves as index which encapsulates all our applications details, i.e. view names, routing details, main view, applications type(Full Screen or SplitApp), application service configuration etc..

Here in **init** function we will configure our plugin. Set properties that we need, set callback functions, call some methods, etc..

```
sap.ui.define([
    "sap/ui/core/UIComponent",
    "sap/ui/Device",
    "MyFirstFioriApp/MyFirstFioriApp/model/models"
], function (UIComponent, Device, models) {
    "use strict";

    var oEventBus = sap.ui.getCore().getEventBus();
    var scannerIsInitialized = false;

    return UIComponent.extend("MyFirstFioriApp.MyFirstFioriApp.Component", {

        metadata: {
            manifest: "json"
        },
    });
});
```

```

/**
 * The component is initialized by UI5 automatically during the startup of the app and calls the init method
 * @public
 * @override
 */
init: function () {
    // call the base component's init function
    UIComponent.prototype.init.apply(this, arguments);

    // enable routing
    this.getRouter().initialize();

    // set the device model
    this.setModel(models.createDeviceModel(), "device");

    if (!scannerIsInitialized) {

        scannerIsInitialized = true;
        window.readerConnected = 0;
        window.scannerActive = false;

        cmbScanner.addOnResume(function (result) {
            cmbScanner.setAvailabilityCallback((readerAvailability) => {
                if (readerAvailability === cmbScanner.CONSTANTS.AVAILABILITY_AVAILABLE) {
                    oEventBus.publish("ScanView", "SetConnectionStatus", {
                        statusText: "AVAILABLE"
                    });
                    cmbScanner.connect((result) => {});
                } else {
                    oEventBus.publish("ScanView", "SetConnectionStatus", {
                        statusText: "NOT AVAILABLE"
                    });
                }
            });
            if (Device.os.android) {
                cmbScanner.connect((result) => {});
            }
        });

        cmbScanner.addOnPause(function (result) {
            if (Device.os.android) {
                cmbScanner.disconnect();
            }
            cmbScanner.setAvailabilityCallback();
        });

        cmbScanner.setPreviewContainerPositionAndSize(0, 0, 100, 50);

        cmbScanner.setConnectionStateDidChangeOfReaderCallback((connectionState) => {
            if (connectionState === cmbScanner.CONSTANTS.CONNECTION_STATE_CONNECTED) {
                oEventBus.publish("ScanView", "SetConnectionStatus", {
                    statusText: "CONNECTED"
                });
                if (window.readerConnected !== connectionState) {
                    cmbScanner.setSymbologyEnabled("SYMBOL.DATAMATRIX", true);
                    cmbScanner.setSymbologyEnabled("SYMBOL.C128", true);
                    cmbScanner.sendCommand("SET_TRIGGER.TYPE 2");
                }
            } else if (connectionState === cmbScanner.CONSTANTS.CONNECTION_STATE_DISCONNECTED) {
                oEventBus.publish("ScanView", "SetConnectionStatus", {
                    statusText: "DISCONNECTED"
                });
            } else if (connectionState === cmbScanner.CONSTANTS.CONNECTION_STATE_CONNECTING) {
                oEventBus.publish("ScanView", "SetConnectionStatus", {
                    statusText: "CONNECTING"
                });
            } else if (connectionState === cmbScanner.CONSTANTS.CONNECTION_STATE_DISCONNECTING) {
                oEventBus.publish("ScanView", "SetConnectionStatus", {
                    statusText: "DISCONNECTING"
                });
            }
            window.readerConnected = connectionState;
        });

        cmbScanner.setAvailabilityCallback((readerAvailability) => {
            if (readerAvailability === cmbScanner.CONSTANTS.AVAILABILITY_AVAILABLE) {
                oEventBus.publish("ScanView", "SetConnectionStatus", {
                    statusText: "AVAILABLE"
                });
            }
        });
    }
}

```

```

        });
        cmbScanner.connect((result) => {});
    } else {
        oEventBus.publish("ScanView", "SetConnectionStatus", {
            statusText: "NOT AVAILABLE"
        });
    }
});

cmbScanner.setActiveStartScanningCallback((result) => {
    if (result === true)
        window.scannerActive = true;
    else
        window.scannerActive = false;
});

cmbScanner.setPreviewOptions(cmbScanner.CONSTANTS.PREVIEW_OPTIONS.DEFAULTS | cmbScanner.COMMON_OPTIONS);

cmbScanner.setCameraMode(cmbScanner.CONSTANTS.CAMERA_MODES.NO_AIMER);

var sdkKEY = "";
if (Device.os.android)
    sdkKEY = this.getModel("i18n").getProperty("MX_MOBILE_LICENSE_ANDROID");
else
    sdkKEY = this.getModel("i18n").getProperty("MX_MOBILE_LICENSE_IOS");

cmbScanner.registerSDK(sdkKEY);

cmbScanner.loadScanner(0, (result) => {
    cmbScanner.connect((result) => {});
});

oEventBus.subscribe("Component", "LoadScanner", this.loadScanner, this);
},
loadScanner: function (sChanel, sEvent, oData) {
    cmbScanner.disconnect((result) => {
        cmbScanner.loadScanner(oData.selectedDevice, (result) => {
            cmbScanner.connect((result) => {});
        });
    });
},
onExit: function () {
    oEventBus.unsubscribe("Component", "LoadScanner", this.loadScanner, this);
    cmbScanner.disconnect();
    cmbScanner.setAvailabilityCallback();
}
});
});

```

cmbScanner is an object that represents our plugin. With this object we can access all [API methods](#) and Constants from our plugin.

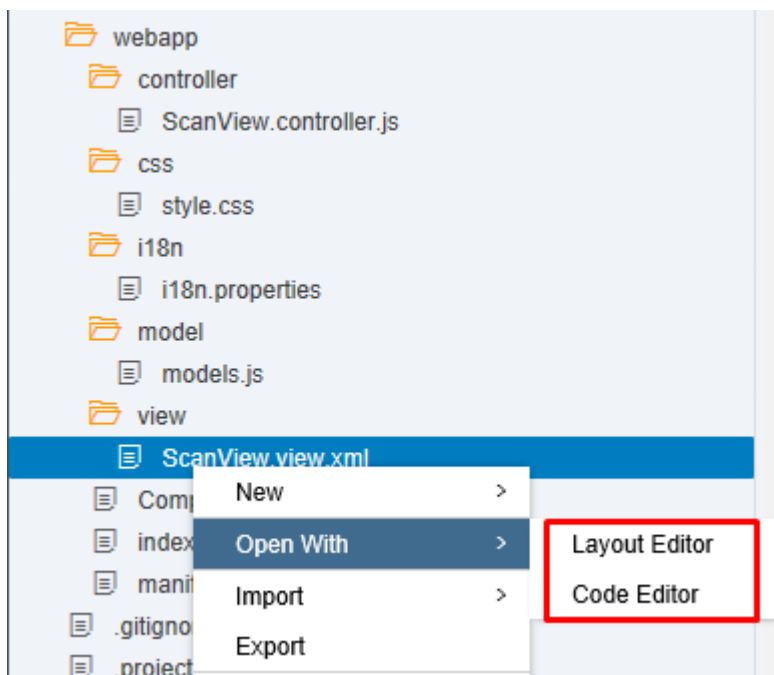
- **scannerIsInitialized** - variable to make sure that we set some functions for initialization only one time
- **window.readerConnected** - global variable that is changed every time when the reader connection state is changed
- **window.scannerActive** - global variable that shows us if scanning is active or not
- **cmbScanner.addOnPause** - in this method we disconnection from reader device on Android (on iOS this is handled automatically), and stop listening to availability changing
- **cmbScanner.addOnResume** - in this method we try to connect with reader device again and set callback for availability changing
- **cmbScanner.setPreviewContainerPositionAndSize** - to set the size and position of the preview container. Learn more [here](#).
- **cmbScanner.setConnectionStateDidChangeOfReaderCallback** - all **connect/disconnect** events should be handled within the callback function set with this API method. If connection state is **connected** we set some reader device settings with [API methods](#) or directly with [sendCommand\(\)](#) method. Here in this example we enable symbologies and set trigger type. Also here we should Licensing the SDK. We will speak about this later in this section
- **cmbScanner.setAvailabilityCallback** - to monitor for availability of the MX device. If MX device is available we try to connect. Learn more [here](#).
- **cmbScanner.setActiveStartScanningCallback** - to monitor the state of the scanner. Learn more [here](#).
- **cmbScanner.loadScanner** - to get a scanner up and running. Learn more [here](#).

Here we are only configuring reader device, handling connections, availability, etc.. We will do scanning and getting results in other views. If we want to notify users about every connection state changed or other info about reader device we will be using [EventBus](#) sap object. With this object we can publish function and call them from any view in the application. In this example on connection state changed with EventBus we are calling **SetConnectionStatus** function that is implemented in view where scanning is performed and set label text to show user current connection state.

Now open **ScanView.view.xml** and add this code:

```
<mvc:View controllerName="MyFirstFioriApp.MyFirstFioriApp.controller.ScanView" xmlns:mvc="sap.ui.core.mvc" displayBlock="tr
xmlns:core="sap.ui.core">
  <Shell id="shell">
    <App id="app">
      <pages>
        <Page id="page" title="{i18n>title}">
          <subHeader>
            <Toolbar id="__toolbar2" width="100%">
              <content>
                <FlexBox id="__box0" width="100%" alignContent="Center" ali
                  <items>
                    <Select id="selectActiveDevice" items="{/De
                      <core:Item text="{text}" key="{key}"
                    </Select>
                  </items>
                </FlexBox>
                <Label id="lblStatus" text="DISCONNECTED" width="100%" text
              </content>
            </Toolbar>
          </subHeader>
          <content>
            <FlexBox id="flexBoxContainer" width="100%" alignContent="Start" alignItems
          </content>
          <footer>
            <Toolbar id="__toolbar1" width="100%">
              <content>
                <Button id="btnScan" press="btnScanPress" text="Scan" width
              </content>
            </Toolbar>
          </footer>
        </Page>
      </pages>
    </App>
  </Shell>
</mvc:View>
```

You can design view with CodeEditor or with LayoutEditor.



After that open **ScanView.controller.js** and add this code

```

sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function (Controller) {
    "use strict";

    var oEventBus = sap.ui.getCore().getEventBus();
    var oModel = new sap.ui.model.json.JSONModel();
    oModel.setData({
        Devices: [{
            key: "0",
            text: "MX Device"
        }, {
            key: "1",
            text: "Mobile Camera"
        }]
    });

    return Controller.extend("MyFirstFioriApp.MyFirstFioriApp.controller.ScanView", {
        onInit: function () {
            this.getView().setModel(oModel);
        },

        onAfterRendering: function () {
            window.scannerActive = false;
            switch (window.readerConnected) {
                case cmbScanner.CONSTANTS.CONNECTION_STATE_CONNECTED:
                    this.getView().byId("lblStatus").setText("CONNECTED");
                    break;
                case cmbScanner.CONSTANTS.CONNECTION_STATE_DISCONNECTED:
                    this.getView().byId("lblStatus").setText("DISCONNECTED");
                    break;
                case cmbScanner.CONSTANTS.CONNECTION_STATE_CONNECTING:
                    this.getView().byId("lblStatus").setText("CONNECTING");
                    break;
                case cmbScanner.CONSTANTS.CONNECTION_STATE_DISCONNECTING:
                    this.getView().byId("lblStatus").setText("DISCONNECTING");
                    break;
                default:
                    this.getView().byId("lblStatus").setText("UNKNOWN");
                    break;
            }
            oEventBus.subscribe("ScanView", "SetConnectionStatus", this.setConnectionStatus, this);

            cmbScanner.setResultCallback((result) => {

                if(result && result.readResults && result.readResults.length > 0){

                    result.readResults.forEach((item, index) => {

                        if (item.goodRead == true) {
                            //Perform some action on barcode read
                            //example:
                            var verticalLayoutContainer = new sap.ui.layout.VerticalLayout(null, {
                                width: "100%"
                            }).addStyleClass("sapUiSmallMarginTop");
                            verticalLayoutContainer.addContent(new sap.m.Label({
                                text: item.symbologyString + ":",
                                textAlign: "Begin",
                                design: "Bold"
                            })).addStyleClass("sapUiSmallMarginBegin"));
                            verticalLayoutContainer.addContent(new sap.m.Text({
                                text: item.readString,
                                textAlign: "Begin"
                            })).addStyleClass("sapUiSmallMarginBegin"));
                            this.getView().byId("flexBoxContainer").addItem(verticalLayoutContainer);
                        }
                        else{
                            //Perform some action when no barcode is read or just leave it empty
                        }
                    });
                }
            });

            btnScanPress: function () {
                if (window.readerConnected === cmbScanner.CONSTANTS.CONNECTION_STATE_CONNECTED) {

```

```

        if (window.scannerActive === true) {
            cmbScanner.stopScanning();
        } else {
            cmbScanner.startScanning();
        }
    },

    setConnectionStatus: function (sChanel, sEvent, oData) {
        this.getView().byId("lblStatus").setText(oData.statusText);
    },

    activeDeviceChanged: function () {

        cmbScanner.disconnect((result) => {
            oEventBus.publish("Component", "LoadScanner", {
                selectedDevice: parseInt(this.getView().byId("selectActiveDevice").getSelectedKey())
            });
        });
    },

    onExit: function () {
        oEventBus.unsubscribe("ScanView", "SetConnectionStatus", this.setConnectionStatus, this);
        if (window.readerConnected === cmbScanner.CONSTANTS.CONNECTION_STATE_CONNECTED) {
            if (window.scannerActive === true) {
                cmbScanner.stopScanning();
            }
        }
        cmbScanner.setResultCallback((result) => {
            return false;
        });
    }
});
});

```

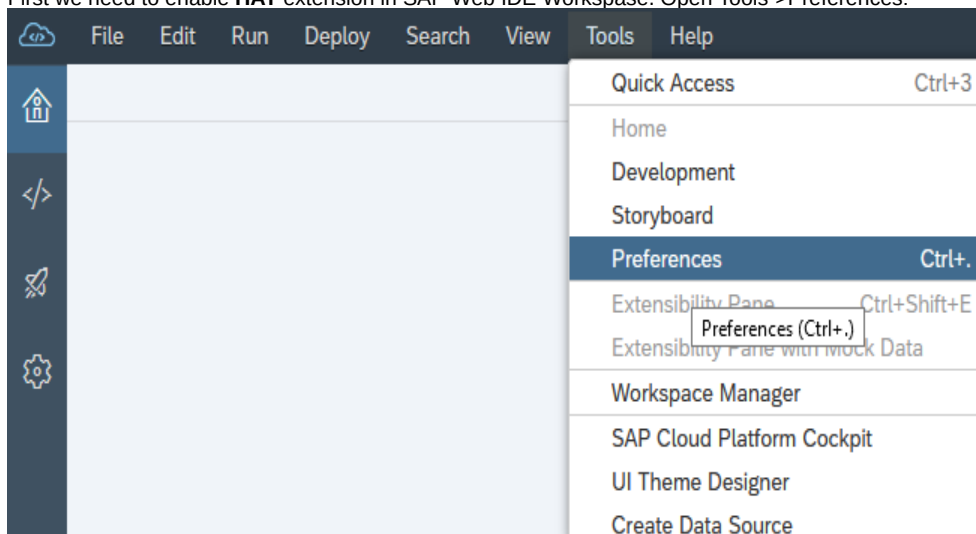
In this view we have a Scan button to [startScanning\(\)/stopScanning\(\)](#) and [cmbScanner.setResultCallback](#) to handle successful scan results.

Build Fiori Mobile Application

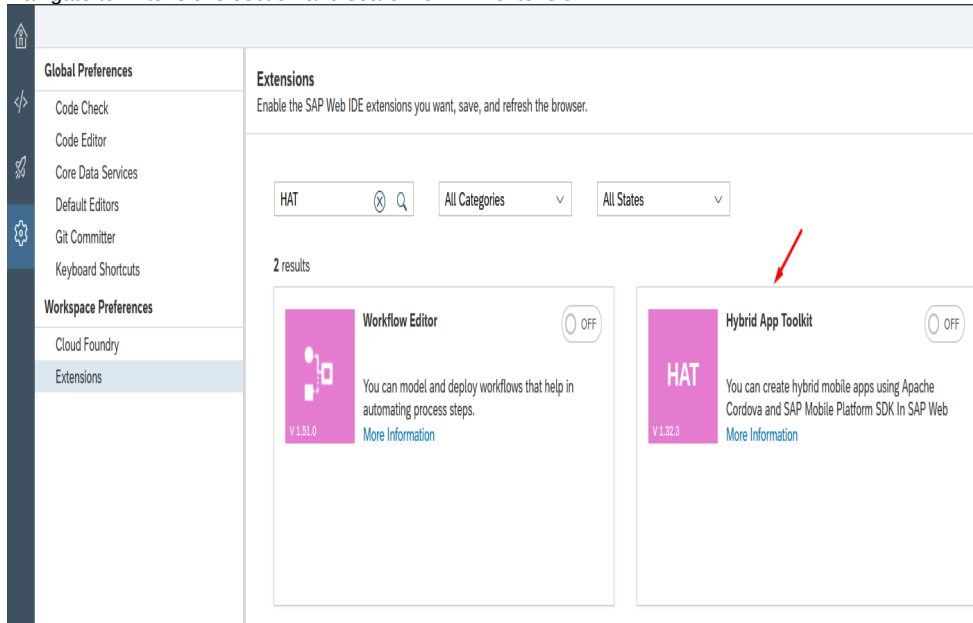
In this section we will explain how to build Fiori Mobile Application (generate .ipa and .apk files).

1. Using **HAT** (Hybrid Application Toolkit)

- First we need to enable **HAT** extension in SAP Web IDE Workspace. Open Tools->Preferences:



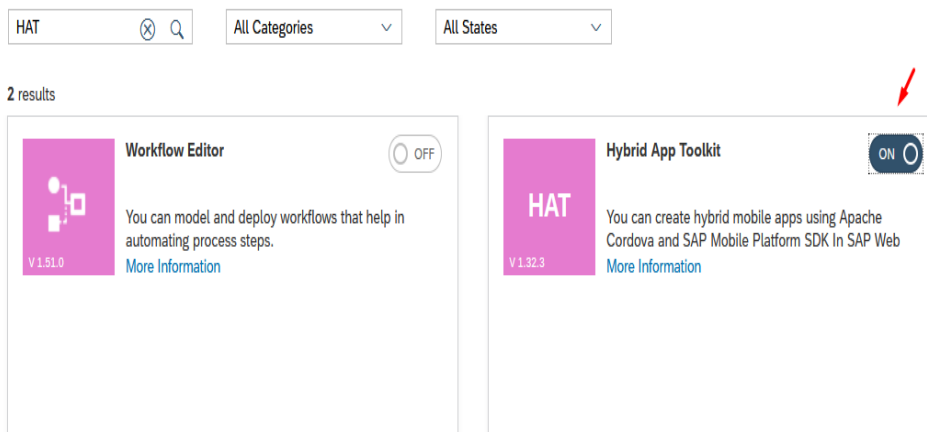
- Navigate to Extensions section and search for HAT extension:



By default HAT extension is disabled. Enable HAT extension and click Save. After enabling we will need to refresh SAP Web IDE.

Extensions

Enable the SAP Web IDE extensions you want, save, and refresh the browser.



- After enabling HAT go again in Tools->Preferences. If you have both Fiori Mobile and Mobile Services enabled you will have options to select one that will be used as Cloud Build Service. Please choose **Mobile Services** and click Save:

Global Preferences

Code Check

Code Editor

Core Data Services

Default Editors

Git Committer

Keyboard Shortcuts

Workspace Preferences

Cloud Foundry

Extensions

Hybrid Application Toolkit

Hybrid Application Toolkit

Hybrid App Toolkit Settings

Cloud Build Service

☐ SAP Fiori Mobile (deprecated)

☒ Mobile Services

☐ Enable Local Add-on Features

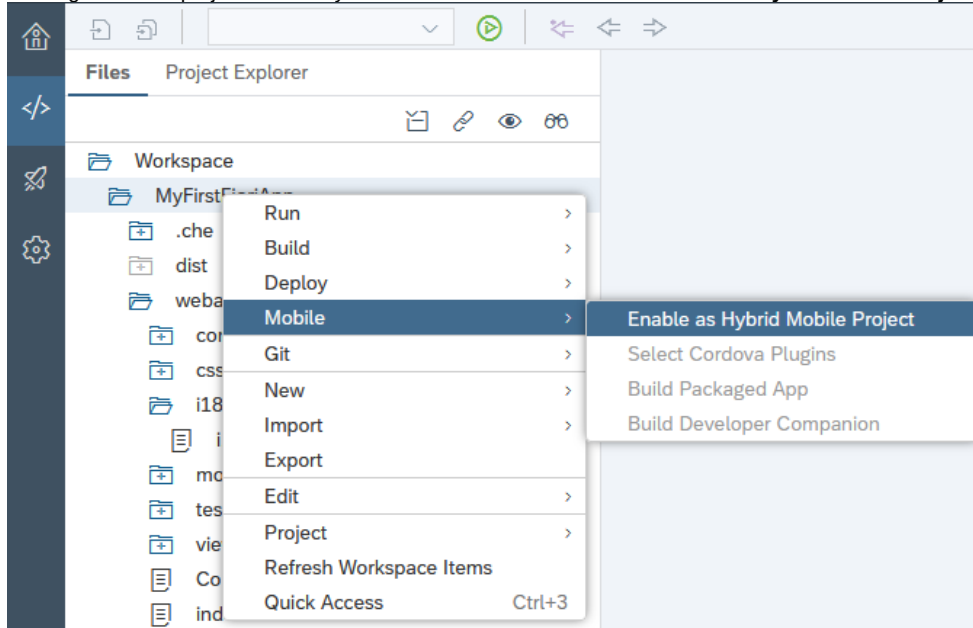
The Local Add-on features have reached end-of-maintenance. The feature will continue to be available for use as to the local add-on will not be accepted.

We recommend using the Cloud Build Service to continue building your hybrid apps with the latest versions of the

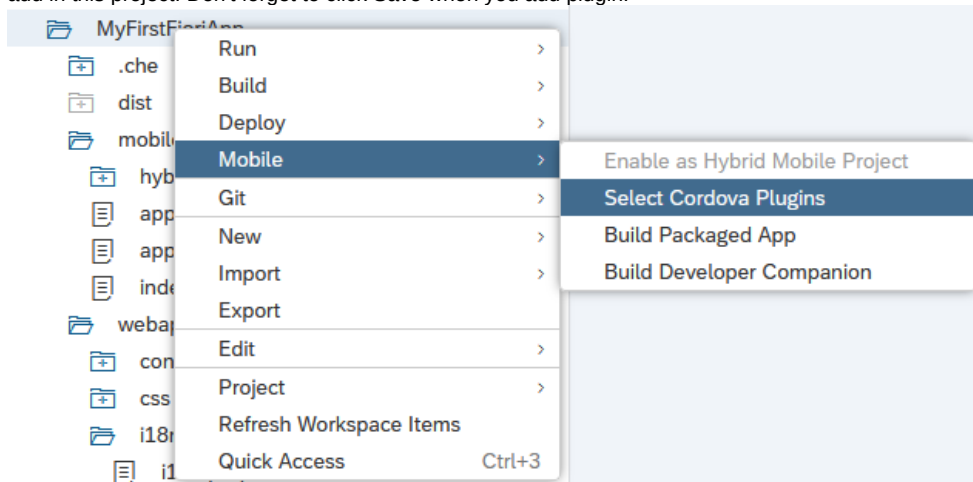
Save

If you don't see these radio buttons you don't need to do anything. **Mobile Service** will be enabled by default.

- Now right click on project root and you will see Mobile section. Click **Enable as Hybrid Mobile Project**





- You can see that some new files are added in our project that allow us to build this project as mobile application. Before build we need to add our **cmbsdk-cordova** plugin. Click on Select Cordova Plugins, find cmbsdk-cordova plugin from public plugins and add in this project. Don't forget to click Save when you add plugin.



Select Cordova Plugins

Selected (9) **Public (1)** Kapsel (20)

Select Platform
cmbsdk-cordova

Plugin ID	Platform	Version	Description	Last Updated On	Actions
cmbsdk-cordova		1.2.14	CMB Scanner Cordova Plugin	6 Jun 2019	<div> Add View Details </div> 



Save

Close

Select Cordova Plugins

Selected (10) **Public (1)** Kapsel (20)

Select Platform
cmbsdk-cordova

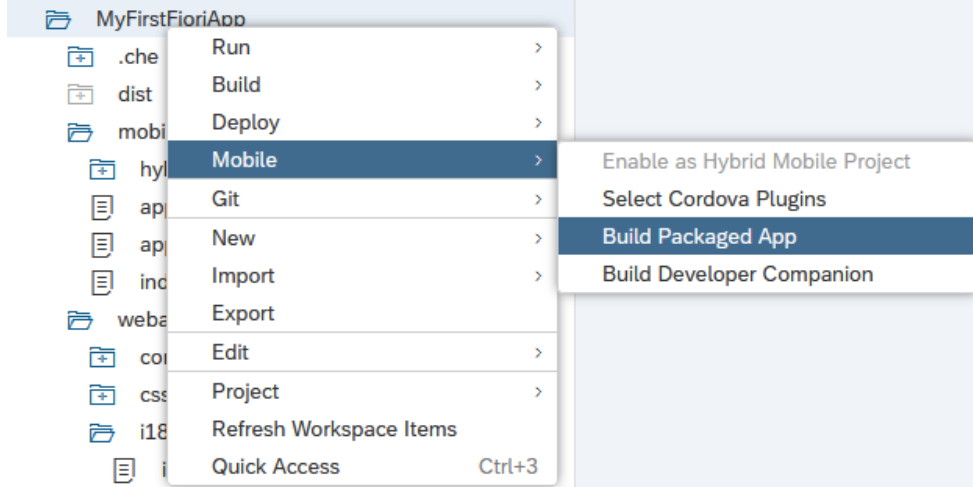
Plugin ID	Platform	Version	Description	Last Updated On	Actions
cmbsdk-cordova		1.2.14	CMB Scanner Cordova Plugin	6 Jun 2019	



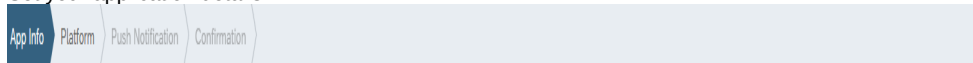
Save

Close

- We are ready to build our mobile application and generate .ipa and .apk files. Click on **Build Packaged App** from Mobile menu.



- Set your application details



Configure Mobile Build Settings

App Info

*App Name	<input type="text" value="MyFirstFioriApp"/>	App Version (?)	<input type="text" value="Leave blank for default versioning"/>
Description	<input type="text" value="Enter app description"/>	Landscape Splash	<input type="text" value="Upload a landscape app splash"/> ...
Icon	<input type="text" value="Upload an app icon"/> ...	Portrait Splash	<input type="text" value="Upload a portrait app splash"/> ...

152px X 152px

2048px X 1536px

1536px X 2048px

- Choose platform or both iOS and Android platforms, select signing profiles (or create one if you don't have) and set some common options

Configure Mobile Build Settings

Platform



Signing Profile

iOSSapAppDev v

Minimum OS Version

11.0 v



Signing Profile

AndroidSapAppDev v

Minimum OS Version ?

5.0 v

Common Options

☐ Build debug-enabled binaries ?

☒ Disable privacy screen ?

☐ Save Xcode and or Android Studio projects ?

Packaged SAPUI5 Version

1.64.1 v

Previous

Next

- Start building process

App Info > Platform > Push Notification > **Confirmation**

Configure Mobile Build Settings

Confirmation

Click the Build button to save the app settings and start building the project.

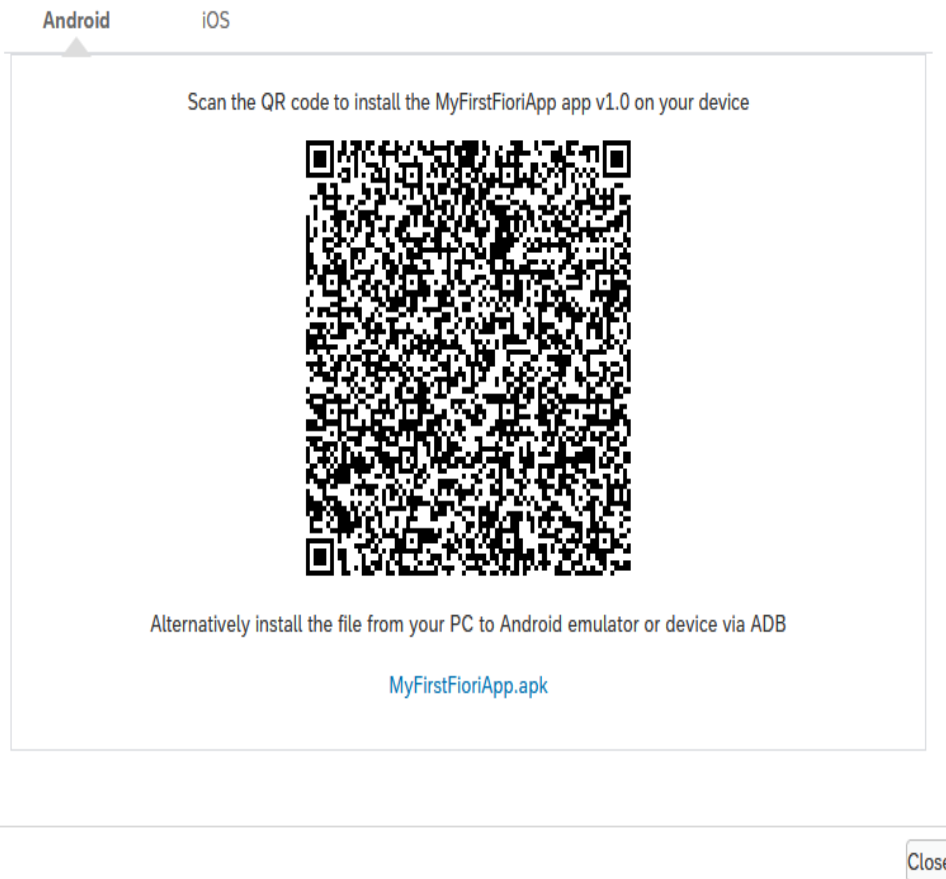
When the build finishes, a result dialog will show. If the build succeeds, you will be able to download and install the app on your device.

Previous

Build

- When building process is finished popup from where you can download .ipa and .apk files will be shown

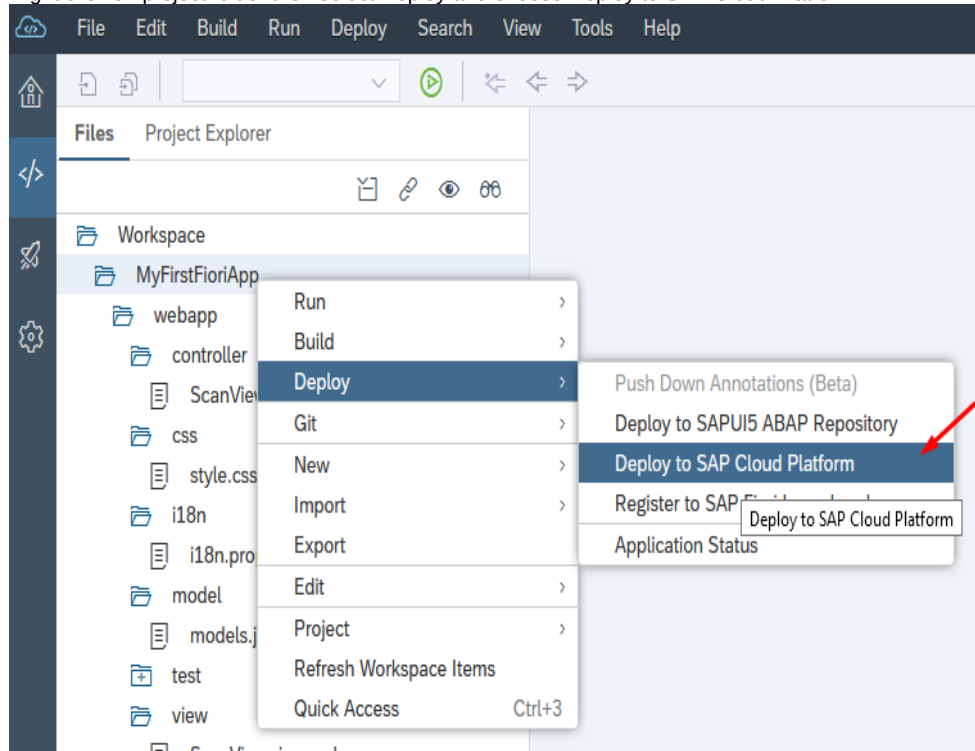
The MyFirstFioriApp Packaged App Build Results



2. Using **Fiori Mobile Service**

Before we start to build our Fiori Mobile Application with Fiori Mobile Service we need to deploy our project on SAP HANA Cloud Platform.

- Right click on project folder then select Deploy and choose Deploy to SAP Cloud Platform



- Set application details and click Deploy

Deploy Application to SAP Cloud Platform

A build will be triggered for the project. The build results will be deployed.

Application Details

☒ Deploy a new application ☐ Update an existing application

*Account

Project Name

*Application Name

Version Management

*Version ☒ Activate

- After a successful deploy, the next step is to Register to SAP Fiori launchpad that we've created in the previous section.

Successfully Deployed

Version 1.0.0 has been created.

[Open the active version of the application](#)

[Open the application's page in the SAP Cloud Platform cockpit](#)

You can now register the application to SAP Fiori launchpad.

Register to SAP Fiori launchpad

Close

- Choose account and set application name

General Information

Tile Configuration

Assignment

Confirmation

Register to SAP Fiori Launchpad

General Information

*Provider Account

portal (flpnwc)

*Application Name [?]

MyFirstFioriApp.MyFirstFioriApp.MyFirstFioriApp

Description

Intent

<input type="checkbox"/>	Semantic Object	Action
<input type="checkbox"/>	MyFirstFioriApp	Display

[More information on intent properties.](#)

- Set Type (Static by default), set Title, and Subtitle

General Information

Tile Configuration

Assignment

Confirmation

Register to SAP Fiori Launchpad

Tile Configuration

*Type

Static

*Title

cmbSDK Fiori App

Subtitle

Sample

Icon

sap-icon://approvals

- Choose Site that we create before, catalog, and group (sample catalog and group is created by default)

General Information > Tile Configuration > **Assignment** > Confirmation

Register to SAP Fiori Launchpad

Assignment

*Site	cmbSDKTestSite
*Catalog	Sample Catalog
*Group	Sample Group

- Click finish on Confirmation tab and if everything is fine the popup below will be shown. Click OK to close this popup. With these steps we deploy our project on HCP and now we can continue to build our Fiori Mobile Application

Successfully Registered

×

The application was successfully registered to SAP Fiori launchpad.

[Open the registered application](#)

[Open SAP Fiori launchpad](#)

OK

- Go to Services tab again, open Fiori Mobile service, and click on Go to Admin Console

 Service: SAP Fiori Mobile - Overview

Enabled

Service Description

The mobile service for SAP Fiori is an end-to-end solution designed to help mobile users. Use the build/packaging service to transform web content into mobile apps. Use the app lifecycle management service to manage the app lifecycle, runtime services to support enterprise app services to provide insights into adoption, usage, and app performance.

Take Action

[Configure Fiori Mobile](#)

[Configure Mobile Packager](#)

[Go to Admin Console](#)

[Go to Mobile Place](#)

- Open Manage Plugins section and upload **cmbSDK-cordova** plugin as custom plugin. You can download latest plugin on this [link](#).

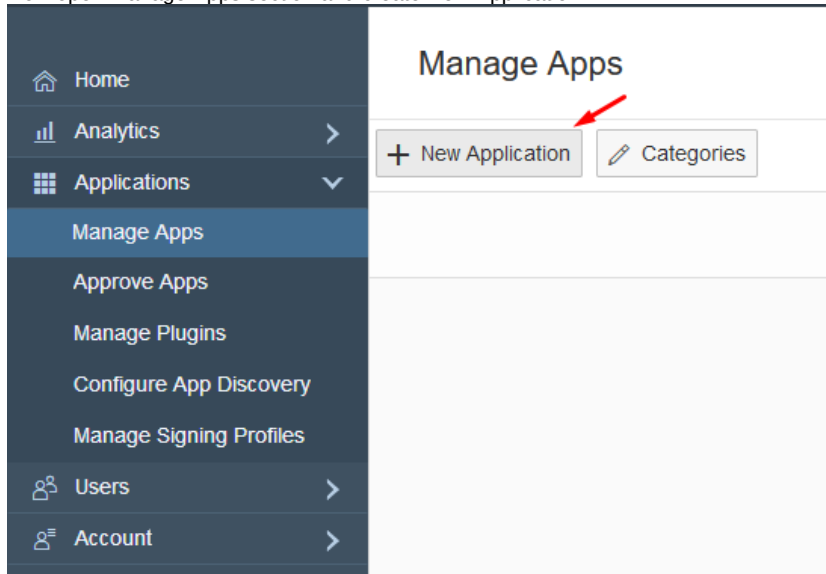
The screenshot shows the 'Manage Plugins' section of the Cognex interface. The left sidebar contains navigation links: Home, Analytics, Applications (expanded), Manage Apps, Approve Apps, Manage Plugins (selected), Configure App Discovery, Manage Signing Profiles, Users, and Account. The main content area is titled 'Manage Plugins' and has tabs for 'CUSTOM PLUGINS' and 'PUBLIC PLUGINS'. Below these tabs are filters for 'All Custom (0)' and 'Recommended (0)'. A '+ Custom Plugin' button is visible. A modal dialog titled 'Add Custom Plugin' is open, showing a 'Select any one' section with 'Source File' (containing 'cmbSDK_Cordova_v2.1.3.12014.zip' and a 'Browse...' button) and 'Source URL' (empty). A note at the bottom of the dialog refers to the 'Managing Custom Plugins' section in the help guide. 'OK' and 'Cancel' buttons are at the bottom right of the dialog.

- Next manage your signing profiles. Open Manage Signing Profiles section and create new ones if you don't have already

The screenshot shows the 'Manage Signing Profiles' section. The left sidebar is the same as the previous screenshot, with 'Manage Signing Profiles' selected. The main content area is titled 'Manage Signing Profiles' and shows a 'Platforms' filter with 'All' selected, indicating '4 Profiles'. Below the filter is a table of profiles:

Platform	Profile Name	Uploaded By	Generated By	Expiry Date / Status
iOS	iOSSapAp...	[Redacted]	[Redacted]	Expiry Date: May 27, 2020
Android	AndroidSa...	[Redacted]	[Redacted]	Expiry Date: Jun 18, 2044
iOS	NAT_Daniel	[Redacted]	[Redacted]	Expired On: May 11, 2018
Android	NAT	[Redacted]	[Redacted]	Expiry Date: Oct 6, 2042

- Now open Manage Apps section and create New Application



- Select first option

[Apps home](#) / Add new application

What's Your Fiori App Scenario?

Tell us about the number of Fiori apps you want to put into a single binary

- ☒ I want to create a local launchpad with only the apps I want to mobilize.
Choose this if you want to create a custom, local launchpad with multiple Fiori applications. We'll download the UI for the apps into the binary when we build it, so it's there when the app launches. You'll need Fiori 1.30+ compliant applications and be running the Fiori Server UI Add-on 2.0.
- ☐ I want to mobilize an existing launchpad or Fiori application.
Choose this if you have a single Fiori URL you want to mobilize, or if you are not running the Fiori Server UI Add-on 2.0.

Previous

Next

Cancel

- Select your Fiori Cloud Edition as Fiori Server

[Apps home](#) / [Add new application](#)

Select Your Fiori Server

Fiori Server Name	Virtual Host with URL	Status
Fiori Cloud Edition		Available
GM4		Available

Previous

Next

Cancel

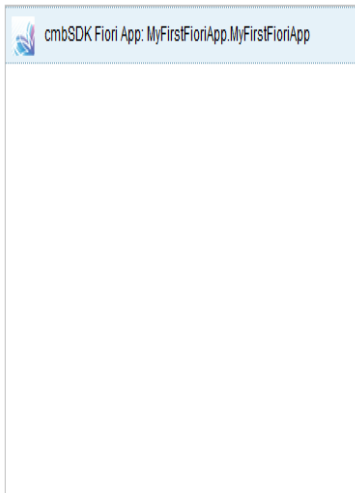
- Select application that we created before (with double click or drag&drop)

[Apps home](#) / [Add new application](#)

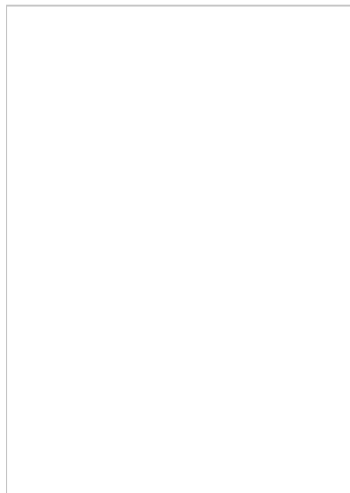
Define your Fiori Application

☒ Select from available Fiori Applications

Available Applications



Selected Applications



☐ Upload advanced configuration file

File Name

Browse...

File Validation

Previous


Next

Cancel

- Set you application name and signing profiles. At the bottom of page you have question "Are you ready to build the application ?". Select "No, I would like to customize my application".

Build Your Fiori Application

152px X 152px



*Application Name

ombSDK Sample App

Uplo...

Connection Security

Users will be authenticated with SAML each time the application is used.

Build Options

☒ iOS
 Signing Profile:

IOSAppDev

Minimum OS Version:

11.0

☒ Android
 Signing Profile:

AndroidAppDev

Minimum OS Version:

5.0

☐ Enable Android and/or iOS project(s) download after build

☐ Enable verbose logging

☐ Build debug-enabled binaries

Note: If you would like to debug your Android application with Chrome and/or your iOS application with Safari, enable this option. For Android, your application will be automatically signed with an internal debug signing profile. For iOS, select a developer signing profile to sign your application to enable you to debug.

For iOS, an iOS signing profile must be selected to enable the build option.

Cached Content


☐ Clear cached content prior to initiating build

Note: SAP caches the SAPUI5 runtime referenced by your application in order to reduce build time. If you believe this content may have been updated since the last time an application was built referencing this SAP Fiori front-end server, enable this option.

Multimedia

Splash Screen - Portrait


1536px X 2048px



Upload

Splash Screen - Landscape

2048px X 1536px



Upload

Are you ready to build the application?

☐ Yes
 ☒ No, I would like to customize my application

Previous

Next

Cancel

- Window where we can continue with app customization will be opened. From here we will add plugin that we uploaded before and disable passcode screen. You can set some others parameters per your needs. Also important thing here is to save every change that will be made.

Apps home / cmbSDK Sample App



Selected (0) | Recommended (0) | Public (2130) | Custom (1)

Select Platform
▼

Search Plugins

Plugin ID	Platform	Version	Description	Last Updated On	Actions
cmb-sdk-cordova-plugin		1.2.14	CMB Scanner Cordova Plugin	18th June 2019	<div> Add </div> <div> View Details </div>



Selected (1) | Recommended (0) | Public (2130) | Custom (1)

Select Platform
▼

Search Plugins

Plugin ID	Platform	Version	Description	Last Updated On	Actions
cmb-sdk-cordova-plugin		1.2.14	CMB Scanner Cordova Plugin	18th June 2019	




- We are ready to build our mobile application and generate .ipa and .apk files. Navigate to Platforms and click **Build All** button if you want to build both iOS and Android platform together, or you can build one by one from Actions grid column.

Apps home / cmbSDK Sample App



State	Debug Enabled	Operating System	Form Factor	App Version	Built On	Certificate Expiration	Actions
Ready to build		iOS	Tablet/Phone	1.0			
Ready to build		ANDROID	Tablet/Phone	1.0			

 Build All
 → Your application is ready to be built!

 Save
  Delete

Build popup will be shown. Click Build button and wait while building process finish.

Build Summary

Application Information

Application Name:

cmbSDK Sample App

Build Options

- | | | |
|--|--|---|
| <input checked="" type="checkbox"/> iOS | Signing Profile: iOSSapAppDev ▼ | Minimum OS Version: 11.0 ▼ |
| <input checked="" type="checkbox"/> Android | Signing Profile: AndroidSapAppDev ▼ | Minimum OS Version: 5.0 ▼ |
| <input type="checkbox"/> Enable Android and/or iOS project(s) download after build | | |
| <input type="checkbox"/> Enable verbose logging | | |
| <input type="checkbox"/> Build debug-enabled binaries | | |

Note: If you would like to debug your Android application with Chrome and/or your iOS application with Safari, enable this option. For Android, your application will be automatically signed with an internal debug signing profile. For iOS, select a developer signing profile to sign your application to enable you to debug.

For iOS, an iOS signing profile must be selected to enable the build option.


Cached Content

- ☐ Clear cached content prior to initiating build

Note: SAP caches the SAPUI5 runtime referenced by your application in order to reduce build time. If you believe this content may have been updated since the last time an application was built referencing this SAP Fiori front-end server, enable this option.

Email Notification

- ☐ Send me an email notification when my applications are built

 Build Cancel

- When building process is finished you can download binary (.ipa or .apk) file and install application on your mobile device

Apps home / cmbSDK Sample App

Insight » Details » Groups » Multimedia » Plugins » App Settings » **Platforms** » Categories » Owner info

State	Debug Enabled	Operating System	Form Factor	App Version	Built On	Certificates	Actions
New Signed		ANDROID	TabletPhone	1.0	Jun 18, 2019		
New Signed		IOS	TabletPhone	1.0	Jun 18, 2019	5/27/202	
Ready to build		IOS	TabletPhone	1.1		5/27/202	
Ready to build		ANDROID	TabletPhone	1.1			

Build All Your application is ready to be built!

Save Delete

Licensing the SDK

If you plan to use the **cmbSDK** to do mobile scanning with a smartphone or a tablet (without the MX mobile terminal), the SDK requires the installation of a license key. Without a license key, the SDK will still operate, although scanned results will be blurred (the SDK will randomly replace characters in the scan result with an asterisk character).

Contact your Cognex Sales Representative for information on how to obtain a license key including trial licenses which can be used for 30 days to evaluate the SDK.

After obtaining your license key open `i18n.properties` file in your project on SAP WEB IDE service, and set your obtained keys.

Files Project Explorer

Workspace

- MyFirstFioriApp
 - .che
 - dist
 - webapp
 - controller
 - css
 - i18n
 - i18n.properties**
 - model

i18n.properties x

```

1 title=Title
2 appTitle=MyFirstFioriApp
3 appDescription=App Description
4 MX_MOBILE_LICENSE_ANDROID=Android_Key
5 MX_MOBILE_LICENSE_IOS=iOS_Key
  
```

Then go back to the `Component.js` file and check this code:

```
...  
  
var sdkKEY = "";  
if (Device.os.android)  
    sdkKEY = this.getModel("i18n").getProperty("MX_MOBILE_LICENSE_ANDROID");  
else  
    sdkKEY = this.getModel("i18n").getProperty("MX_MOBILE_LICENSE_IOS");  
  
cmbScanner.registerSDK(sdkKEY);  
  
...
```

You can see that you read this key from `i18n.properties` and call **`cmbScanner.registerSDK`** method to register SDK with your license key.