**Hosting ZelleSDK native Android views in Flutter app with Platform Views**

* Platform views allow you to embed native views in a Flutter app, so you can apply transforms, clips, and opacity to the native view from Dart.

## **Create the view wrapper**

* To create the view wrapper on the native android folder. Create a **kotlin/Java** wrapper class inside **main** folder and import the below codes.
* Create libs folder inside **app** directory and import the ZelleSDK.aar file inside.
* **Create the view wrapper:**

**import** android.annotation.SuppressLint  
**import** android.content.Context  
**import** android.content.ContextWrapper  
**import** android.view.View  
**import** android.view.ViewGroup  
**import** android.widget.LinearLayout  
**import** androidx.fragment.app.FragmentActivity  
**import** androidx.fragment.app.FragmentContainerView  
**import** com.fiserv.dps.mobile.sdk.bridge.model.Bridge  
**import** com.fiserv.dps.mobile.sdk.bridge.zelleview.BridgeView  
**import** com.fiserv.dps.mobile.sdk.bridge.zelleview.Zelle  
**import** com.fiserv.dps.mobile.sdk.interfaces.GenericTag  
**import** io.flutter.plugin.common.StandardMessageCodec  
**import** io.flutter.plugin.platform.PlatformView  
**import** io.flutter.plugin.platform.PlatformViewFactory

**class** ZelleNativeView(  
 **private val context**: Context,  
 id: Int,  
 **private val creationParams**: Map<String?, Any?>?,  
) : PlatformView, GenericTag {  
 **private var view**: FragmentContainerView? = **null  
 lateinit var bridge** : Bridge  
 **lateinit var linearLayout**: LinearLayout  
  
 @SuppressLint(**"ResourceType"**)  
 **override fun** getView(): View = **view** ?: kotlin.*run* **{  
 val** activity = **context**.*getFragmentActivityOrThrow*()  
 **linearLayout** = LinearLayout(**context**)  
 **linearLayout**.*layoutParams* = LinearLayout.LayoutParams(ViewGroup.LayoutParams.*MATCH\_PARENT*, ViewGroup.LayoutParams.*MATCH\_PARENT*)  
 **linearLayout**.*orientation* = LinearLayout.*VERTICAL* **linearLayout**.*id* = 1234  
 BridgeView.genericTag = **this  
 val** zelle = Zelle(  
 applicationName = **creationParams**!!.get(**"applicationName"**) **as** String?,  
 baseURL = creationParams.get(**"baseUrl"**) **as** String,  
 institutionId = creationParams.get(**"institutionId"**) **as** String,  
 product = **creationParams**.get(**"product"**) **as** String,  
 ssoKey = creationParams.get(**"ssoKey"**) **as** String,  
 fi\_callback = creationParams.get(**"fi\_callback"**) **as** Boolean,  
 appData = **creationParams**.get(**"appData"**) **as** Map<String, Map<String, String>>?,  
 parameters = creationParams.get(**"parameter"**) **as** Map<String, String>  
 )  
 **bridge** = Bridge(**context**, zelle)  
 zelle.preCacheContacts = **true  
 val** bridgeView = **bridge**.view()  
 **val** view = FragmentContainerView(**context**)  
 view.setId(**linearLayout**.*id*)  
 activity.supportFragmentManager.beginTransaction().add(**linearLayout**.*id*, bridgeView, **"BridgeView"** ).commit()  
 **this**.**view** = view  
 view  
 **}  
  
 override fun** dispose() {  
 **view** = **null** }  
  
 **private fun** Context.getFragmentActivityOrThrow(): FragmentActivity {  
 **if** (**this is** FragmentActivity) {  
 **return this** }  
  
 **var** currentContext = **this  
 while** (currentContext **is** ContextWrapper) {  
 **if** (currentContext **is** FragmentActivity) {  
 **return** currentContext  
 }  
 currentContext = currentContext.*baseContext* }  
  
 **throw** IllegalStateException(**"Unable to find activity"**)  
 }  
  
 **override fun** getValue(name: String) {  
  
 }  
  
  
 **override fun** sessionTag(tag: String) {  
 }  
  
}  
  
**class** ZelleNativeViewFactory : PlatformViewFactory(StandardMessageCodec.INSTANCE) {  
 **override fun** create(context: Context?, viewId: Int, args: Any?): PlatformView {  
 **val** creationParams = args **as** Map<String?, Any?>?  
 **return** ZelleNativeView(context!!, viewId, creationParams) }  
}

* **Wire the wrapper:**

In the MainActivity file, add these lines to wire your newly created native view:

**override fun** configureFlutterEngine(flutterEngine: FlutterEngine) {  
 flutterEngine.platformViewsController  
 .registry  
 .registerViewFactory(**"zelleviewtype"**, ZelleNativeViewFactory())  
}

* Finally, modify your build.gradle file to require ZelleSDK minimal Android SDK versions and implement the required dependencies:

android {

defaultConfig {

minSdkVersion 24 // if using virtual display.

}

}

dependencies **{** implementation **'androidx.appcompat:appcompat:1.3.1'** implementation **'com.google.android.material:material:1.4.0'** implementation **'androidx.constraintlayout:constraintlayout:2.1.0'** implementation(**'com.journeyapps:zxing-android-embedded:4.3.0'**) **{** transitive = **false}** implementation **'com.google.zxing:core:3.4.0'** implementation **'androidx.core:core-ktx:1.6.0'** implementation fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**, **'\*.aar'**])  
**}**

* Set the Launch theme for .MainActivity in manifest file:

<**activity  
 android:name=".MainActivity"  
 android:exported="true"  
 android:launchMode="singleTop"  
 android:theme="@style/LaunchTheme"  
 android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|locale|layoutDirection|fontScale|screenLayout|density|uiMode"  
 android:hardwareAccelerated="true"  
 android:windowSoftInputMode="adjustResize"**>

* Inside res folder on styles.xml file create a LauchTheme by importing the material dependency to handle Dark/Light Mode.

<**style name="LaunchTheme" parent="Theme.MaterialComponents.DayNight.NoActionBar"**>

<**item name="android:windowBackground"**>@drawable/launch\_background</**item**>  
</**style**>

**Hosting ZelleSDK native iOS views in Flutter app with Platform Views**

## **Create the view wrapper**

* To create the view wrapper on the native iOS folder. Create a **swift** wrapper class inside **Runner** folder and import the below codes.
* Import the ZelleSDK.xcframework file inside the Runner folder.
* **Create the view wrapper:**

import Flutter

import UIKit

import ZelleSDK

**class** ZelleNativeViewFactory: NSObject, FlutterPlatformViewFactory {

**private** **var** messenger: FlutterBinaryMessenger

**private** **var** viewController:UIViewController

**init**(messenger: FlutterBinaryMessenger, viewController:UIViewController ) {

**self**.messenger = messenger

**self**.viewController = viewController

**super**.init()

}

**public** **func** createArgsCodec() -> FlutterMessageCodec & NSObjectProtocol {

**return** FlutterStandardMessageCodec.sharedInstance()

}

**func** create(

withFrame frame: CGRect,

viewIdentifier viewId: Int64,

arguments args: **Any**?

) -> FlutterPlatformView {

**return** ZelleNativeView(

frame: frame,

viewIdentifier: viewId,

arguments: args,

binaryMessenger: messenger, viewController: viewController)

}

}

**class** ZelleNativeView: NSObject, FlutterPlatformView {

**private** **var** \_view: UIView

**init**(

frame: CGRect,

viewIdentifier viewId: Int64,

arguments args: **Any**?,

binaryMessenger messenger: FlutterBinaryMessenger?,

viewController:UIViewController

) {

\_view = UIView()

**super**.init()

// iOS views can be created here

createNativeView(view: \_view, frame: frame, viewController: viewController, args: args)

}

**func** view() -> UIView {

**return** \_view

}

**func** createNativeView(view \_view: UIView, frame: CGRect, viewController:UIViewController, args : **Any**?){

**let** arguments = args **as**? Dictionary<String, **Any**>

**let** zelle = Zelle(

applicationName: (arguments?["applicationName"] **as**? String)!,

baseUrl: (arguments?["baseUrl"] **as**? String)!,

institutionId: (arguments?["institutionId"] **as**? String)!,

product: (arguments?["product"] **as**? String)!,

ssoKey: (arguments?["ssoKey"] **as**? String)!,

parameters: (arguments?["parameter"] **as**? Dictionary<String, String>)!

)

**lazy** **var** bridge: Bridge = {

Bridge(

config: zelle,

viewController: viewController

)

}()

**self**.\_view = bridge.view(frame: frame)

}

}

* **Wire the wrapper:**

In the AppDelegate file, add these lines to wire your newly created view:

**weak** **var** registrar = **self**.registrar(forPlugin: "zelle")

**let** factory = ZelleNativeViewFactory(messenger: registrar!.messenger(), viewController: **self**.window.rootViewController! )

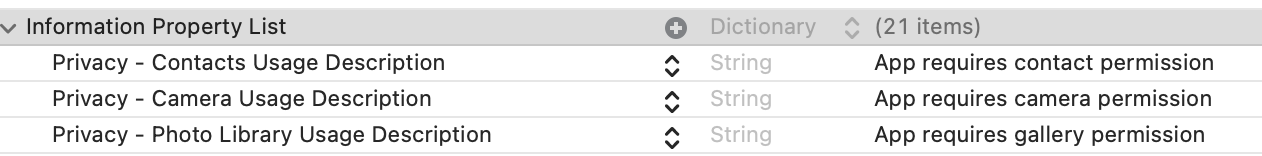
registrar!.register(

factory,

withId: "zelleviewtype")

**return** **super**.application(application, didFinishLaunchingWithOptions: launchOptions)

* Finally, to access the SDK functionalities like accessing the Camera, Contact and Photo Gallery it is mandatory to ask permission from the user.
* To achieve that provide the permission contents in **Info.plist** file to intimate user while accessing SDK required functionalities.



**Flutter:**

* Create a file named \_MyZellePageState.dart.
* Implement the following code inside flutter code:

**const** String viewType = **'zelleviewtype'**;  
  
Widget \_buildAndroid(BuildContext context, Map<String, **dynamic**> params) {  
 **return** AndroidView(  
 viewType: viewType,  
 layoutDirection: TextDirection.**ltr**,  
 creationParams: params,  
 creationParamsCodec: **const** StandardMessageCodec(),  
 );  
}  
  
Widget \_buildIoS(BuildContext context, Map<String, **dynamic**> params) {  
 **return** UiKitView(  
 viewType: viewType,  
 layoutDirection: TextDirection.**ltr**,  
 creationParams: params,  
 creationParamsCodec: **const** StandardMessageCodec(),  
 );  
}

**class** \_MyZellePageState **extends** State<MyHomePage> {  
  
 bool **launchZelleUI** = **false**;  
  
 TextEditingController **\_applicationNameController** = TextEditingController();  
 TextEditingController **\_baseUrlController** = TextEditingController();  
 TextEditingController **\_institutionIdController** = TextEditingController();  
 TextEditingController **\_productController** = TextEditingController();  
 TextEditingController **\_ssoKeyController** = TextEditingController();  
  
 FocusNode **\_applicationNameFocus** = FocusNode();  
 FocusNode **\_baseUrlFocus** = FocusNode();  
 FocusNode **\_institutionIdFocus** = FocusNode();  
 FocusNode **\_productFocus** = FocusNode();  
 FocusNode **\_ssoKeyFocus** = FocusNode();  
  
 **void** \_lauchZelle() **async**{  
 setState(() {  
 **launchZelleUI** = **true** ;  
 });  
 }  
  
 **void** \_backButton() **async**{  
 setState(() {  
 **launchZelleUI** = **false** ;  
 });  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 Map<String, String> map = **new** HashMap();  
 map[**'param1'**] = **"1234"**;  
 map[**'param2'**] = **"something"**;  
 map[**'param3'**] = **"abc123"**;  
  
 Map<String,Map<String, String>> pdData=**new** LinkedHashMap();  
 *//contact* Map<String, String> contact\_pd = **new** LinkedHashMap();  
 contact\_pd[**'title'**] = **"CONTACT TITLE"**;  
 contact\_pd[**'message'**] = **"CONTACT MESSAGE"**;  
 *//camera* Map<String, String> camera\_pd = **new** LinkedHashMap();  
 camera\_pd[**'title'**] = **"CAMERA TITLE"**;  
 camera\_pd[**'message'**] = **"CAMERA MESSAGE"**;  
 *//gallery* Map<String, String> gallery\_pd = **new** LinkedHashMap();  
 gallery\_pd[**'title'**] = **"GALLERY TITLE"**;  
 gallery\_pd[**'message'**] = **"GALLERY MESSAGE"**;  
  
 pdData[**'pd\_contact'**] = contact\_pd;  
 pdData[**'pd\_camera'**] = camera\_pd;  
 pdData[**'pd\_gallery'**] = gallery\_pd;  
  
 *// https://dhayalu-fiserv.github.io/demo/index.html  
 // https://jayjt11.github.io/Sdk/index.html* **final** Map<String, **dynamic**> params = <String, **dynamic**>{};  
 params[**'applicationName'**] = **\_applicationNameController**.**text**;  
 params[**'baseUrl'**] = **\_baseUrlController**.**text**;  
 *// params['baseUrl'] = Platform.isAndroid ?'https://dhayalu-fiserv.github.io/demo/index.html' : 'https://jayjt11.github.io/Sdk/index.html';* params[**'institutionId'**] = **\_institutionIdController**.**text**;  
 params[**'product'**] = **\_productController**.**text**;  
 params[**'ssoKey'**] = **\_ssoKeyController**.**text**;  
 params[**'fi\_callback'**] = **true**;  
 params[**'appData'**] = pdData;  
 params[**'parameter'**] = map;  
  
 **return** SafeArea(  
 child: Scaffold(  
 resizeToAvoidBottomInset: **false**,  
 appBar: AppBar(  
 title: Text(**'Embedded Native View'**) ,  
 leading: BackButton(  
 color: Colors.*white*,  
 onPressed: \_backButton ,  
 ),  
 centerTitle: **true**,  
 ),  
 body: !**launchZelleUI** ?Column(  
 mainAxisSize: MainAxisSize.**min**,  
 children: [  
 SizedBox(height: 10,),  
 Text(**"Welcome to Zelle"**),  
 Padding(  
 padding: EdgeInsets.all(10),  
 child: TextFormField(  
 controller: **\_applicationNameController**,  
 focusNode: **\_applicationNameFocus**,  
 onFieldSubmitted: (v) {  
 FocusScope.*of*(context).requestFocus(**\_baseUrlFocus**);  
 },  
 decoration: InputDecoration(  
 border: OutlineInputBorder(),  
 hintText: **'Enter application name'**,  
 ),  
 ),  
  
 ),  
 Padding(  
 padding: EdgeInsets.all(10),  
 child: TextFormField(  
 controller: **\_baseUrlController**,  
 focusNode: **\_baseUrlFocus**,  
 onFieldSubmitted: (v) {  
 FocusScope.*of*(context).requestFocus(**\_institutionIdFocus**);  
 },  
 decoration: InputDecoration(  
 border: OutlineInputBorder(),  
 hintText: **'Enter baseUrl'**,  
 ),  
 ),  
  
 ),  
 Padding(  
 padding: EdgeInsets.all(10),  
 child: TextFormField(  
 controller: **\_institutionIdController**,  
 focusNode: **\_institutionIdFocus**,  
 onFieldSubmitted: (v) {  
 FocusScope.*of*(context).requestFocus(**\_productFocus**);  
 },  
 decoration: InputDecoration(  
 border: OutlineInputBorder(),  
 hintText: **'Enter institutionId'**,  
 ),  
 ),  
  
 ),  
 Padding(  
 padding: EdgeInsets.all(10),  
 child: TextFormField(  
 controller: **\_productController**,  
 focusNode: **\_productFocus**,  
 onFieldSubmitted: (v) {  
 FocusScope.*of*(context).requestFocus(**\_ssoKeyFocus**);  
 },  
 decoration: InputDecoration(  
 border: OutlineInputBorder(),  
 hintText: **'Enter product'**,  
 ),  
 ),  
  
 ),  
 Padding(  
 padding: EdgeInsets.all(10),  
 child: TextFormField(  
 controller: **\_ssoKeyController**,  
 focusNode: **\_ssoKeyFocus**,  
 decoration: InputDecoration(  
 border: OutlineInputBorder(),  
 hintText: **'Enter ssoKey'**,  
 ),  
 ),  
  
 ),  
 ElevatedButton(onPressed: \_lauchZelle, child:  
 Text(  
 **"Launch Zelle"** )),  
 SizedBox(height: 10,),  
 ],  
 ) : Platform.*isAndroid* ? \_buildAndroid(context, params) : \_buildIoS(context, params) ,  
 ),  
 );  
 }  
}