ticketmaster®

Presence SDK

Android Integration

October 16, 2017

Change log

Changes (10/16/2017 Release 1.2.0)

- Added support for entering verification code for linking TMR account for
- Fixed seat description selection for posting flow for Archtics
- Fixed intermittent logout issue
- Separated the login flow from main SDK so it can just be used for login and fetching valid access token.
- Fixed other in-field issues

Changes (10/05/2017 Release 1.1.1)

- Added support for choosing between different payout methods during Resale flow for Archtics.

Changes (09/29/2017 Release 1.1.0)

- Added support for Apigee and made integration with SDK much simpler.
- Added support for seat selection in Transfer and Resale flow.
- Added 3 new helper methods for checking login status
- Fixed few minor issues in the SDK.

Changes (09/13/2017 Release 1.0.0)

- Resolved login issue of getting stuck in log-in screen after logging in and clicking“authorize” button.
- Resolved login issue of getting 401 status when clicking on an event
- Handled session expiry error
- Fixed crashlytics crashes
- Fixed crash due to multiple loading of TmxMainView
- Fixed missing seats at group selection view in transfer/resale flow
- Fixed in-field crash/issues
- Fixed edit resale operation with 4 digit price
- Added forgot password on log-in UI **(08/24/2017)**
- Fixed duplicate ticket card in listing pending state **(09/05/2017)**
- Added price breakdown in ticket details **(09/05/2017)**
- Supported upsell items **(09/05/2017)**
- Fixed payment account delete operation failure in concurrent log-in scenario **(09/05/2017)**
- Restored barcode immediately after resale/transfer cancel **(09/12/2017)**
- Fix to automatically refresh the event list view once user logs in to second server **(09/12/2017)**
- Bundle and unbundle multi tickets resale or transfer operations **(09/12/2017)**
- Added support for password recovery for teams still on old account manager **(09/13/2017)**

What You Need

To integrate Presence sdk in your application, you will need the following aar file:

- PresenceSDK-release-1.1.1.1.aar

Supported API levels

- API level 16 ~ 25

Release Notes

Requirements

- Supported API level 16 ~ 25

What's New?

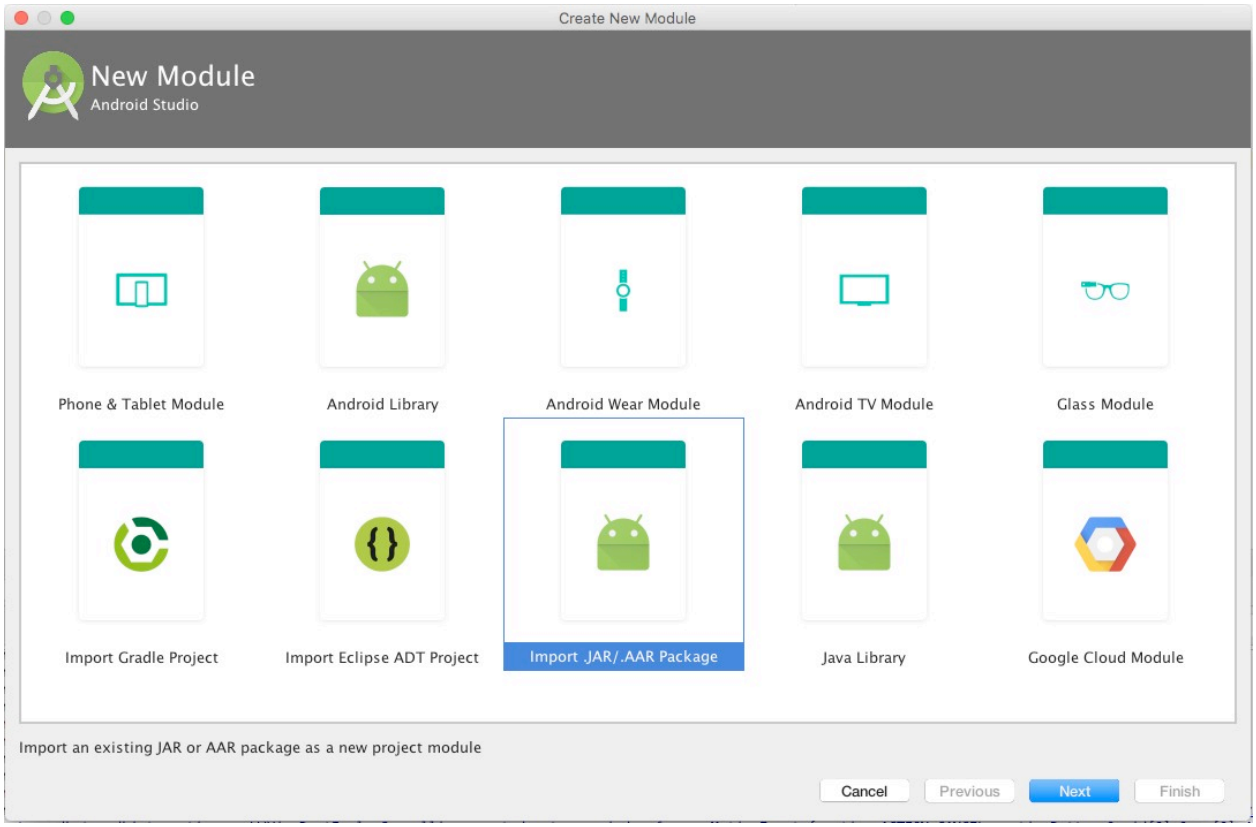
- Resolved login issue of getting stuck in log-in screen after logging in and clicking "authorize" button
- Resolved login issue of getting 401 status when clicking on an event
- Fixed missing seats group selection view in transfer/resale flow
- Added handling of session expiry error
- Fixed in-field crash/issues
- Fixed edit resale operation with 4 digit price

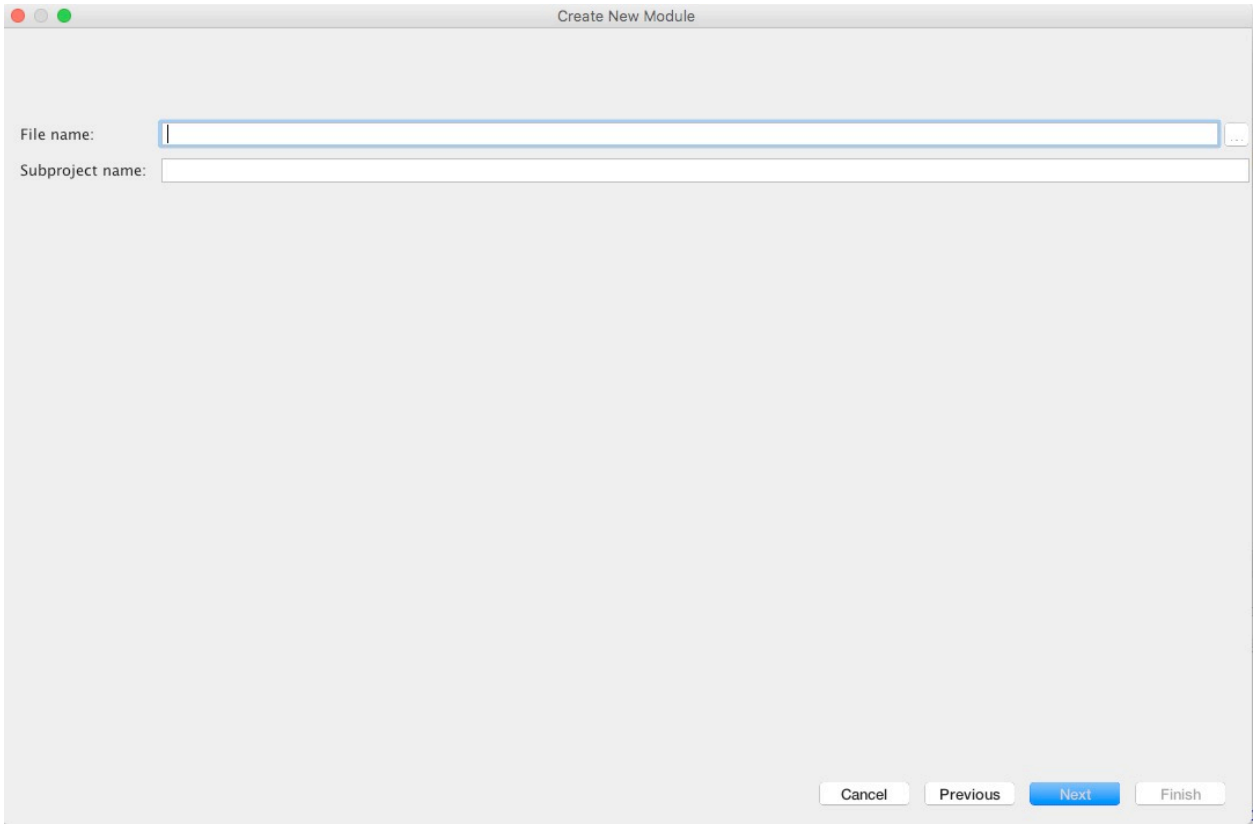
Integrating with Presence SDK

Step 1. Drop Presence sdk aar file in your application project libs folder

Step 2. Import it through "File -> New -> New Module -> Import .JAR / .AAR package". Specify and locate the aar file.

How To Integrate with Presence Sdk





Step 3. Go to your app module build gradle file and set the name of each aar file as compile dependencies as follows:

compile project(':PresenceSDK-release-1.2.0.1')

Step 4. Add the following dependencies in the same place as step #3:

```
compile 'com.android.support:cardview-v7:24.2.1'
compile 'com.android.support:appcompat-v7:24.2.1'
compile 'com.android.support:support-v4:24.2.1'
compile 'com.android.support:recyclerview-v7:24.2.1'
compile 'com.android.support:design:24.2.1'
compile 'com.android.volley:volley:1.0.0'
compile 'com.google.code.gson:gson:2.4'
compile 'com.squareup.picasso:picasso:2.5.2'
compile 'com.romandanylyk:pageindicatorview:0.0.5'
compile 'com.google.zxing:core:3.2.1'
compile 'com.android.support:percent:24.2.1'
```

After adding them, the build gradle dependencies will look similar to the one shown as below:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.3"

    defaultConfig {
        applicationId "com.ticketmaster.tmx.sdk.tmxsdkintegrationapp"
        minSdkVersion 16
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:24.2.1'

    // added for Tmx sdk integration
    compile project(':LoginSdk-1.0.0')
    compile project(':TmxSdk-1.0.0.1')
    compile 'com.android.support:cardview-v7:24.2.1'
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile 'com.android.support:support-v4:24.2.1'
    compile 'com.android.support:recyclerview-v7:24.2.1'
    compile 'com.android.support:design:24.2.1'
    compile 'com.android.volley:volley:1.0.0'
    compile 'com.google.code.gson:gson:2.4'
    compile 'com.yqritc:recyclerview-flexibledivider:1.4.0'
    compile 'com.squareup.picasso:picasso:2.5.2'
    compile 'com.romandanylyk:pageindicatorview:0.0.5'
    compile 'com.google.zxing:core:3.2.1'
}
```

Step 5. Create a configurePresenceSDK() method inside your activity class. In this method, the account credentials and branding color will be configured.

```
private void configurePresenceSDK() {

  PresenceSDK.getPresenceSDK(getApplicationContext()).setConfig(consumerKey
/*Consumer key provided on dev portal*/,
displayName /*your team display name */,
useNewAccountManager /*true/false for choosing between new or old account
manager, by default it will choose old accounts manager */);

  // Configure your branding color for the SDK
  //opaque red
  presenceSDK.getPresenceSDK(this).setBrandingColor(Color.parseColor("#ffff0000")
  );
```

```
}
```

NOTE:

1. Be aware that you need to pass application context object (not activity context) to PresenceSDK.getPresenceSDK() method. If it is not an application context, the presence sdk might reject and throw a runtime exception with a message to bring it to developer's attention.

2. To get consumer key please create an account on **https://developer.ticketmaster.com and register your app and it will generate a consumer key that can be used in the above method. Before you can use Presence SDK you will have to provide the generated consumer key together with consumer secret and redirect URI to Presence SDK support team so we can configure your app on our end!**

Step 6. Create launchPresenceSDK() method inside the same activity class. In this method, you will implement a login listener and start the presence sdk

```
private void launchPresenceSDK() {
        PresenceSDK.getPresenceSDK(this).start(this,
                R.id.presenceSDK /*the id of the framelayout defined in
        activity layout where you want to load PreseneSDK UI fragment */
                , new TMLoginListener() {
                @Override
                public void onLoginSuccessful(TMLoginApi.BackendName
                backendName, String accessToken) {
                        Log.i(TAG, "Inside onLoginSuccessful");
                }

                Override
                public void onLoginFailed(TMLoginApi.BackendName backendName,
                String errorMessage) {
                        Log.i(TAG, "Inside onLoginFailed");
                }

                @Override
                public void onLoginCancelled(TMLoginApi.BackendName
                backendName) {
                        Log.i(TAG, "Inside onLoginCancelled");
                }

                @Override
                public void onLoginMethodUsed(TMLoginApi.BackendName
                backendName, TMLoginApi.LoginMethod method) {
                        Log.i(TAG, "Inside onLoginMethodUsed");
                }

                @Override
                public void onLoginForgotPasswordClicked(TMLoginApi.BackendName
                backendName) {
                        Log.i(TAG, "Inside onLoginForgotPasswordClicked");
                }

                @Override
                public void onCacheCleared() {
                        Log.i(TAG, "Inside onCacheCleared");
                }
```

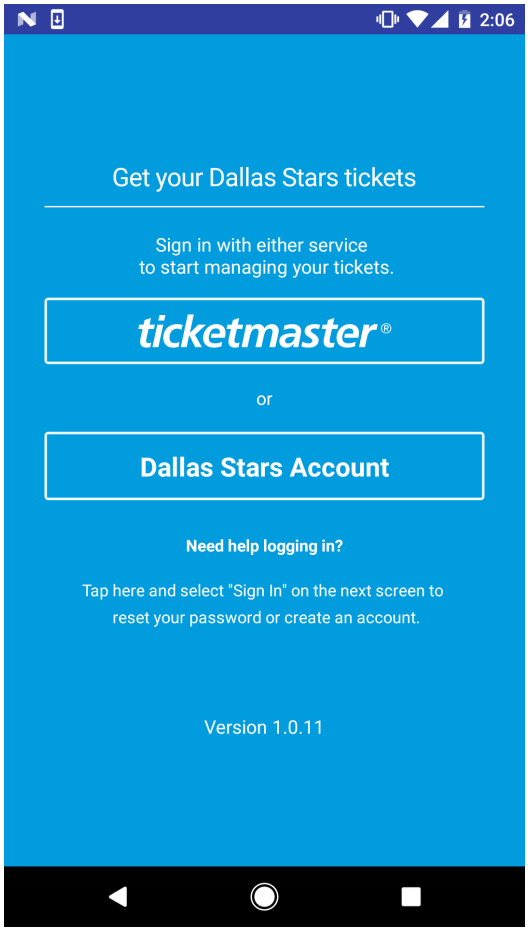
```
                    @Override
                    public void onMemberUpdated(@Nullable TMLoginApi.MemberInfo
                    member) {
                            Log.i(TAG, "Inside onMemberUpdated");
                    }
            });

    }
```

Step 7. Call the configurePresenceSDK() and launchPresenceSDK() methods in the activity class onCreate() method.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // set layout where you want to load presence sdk login entry view
        setContentView(R.layout.activity_main);
        // call configure presence sdk method
        configurePresenceSDK();
        // call launch presence sdk method
        launchPresenceSDK();
}
```

This will load the entry fragment (UI shown below) where a login screen will be prompted to users to choose Ticketmaster or Team Account to login.



Step 8. Define the "AppTheme" style in styles.xml as follows:

```
<style name="AppTheme"
parent="Theme.AppCompat.Light.NoActionBar">
```

Step 9. Try to build and compile. At this point, it should be compiled without errors.

Logout Methods

You can implement log out functionality on your UI with by calling these API :

```
// TM logout
PresenceSDK.getPresenceSDK(context).logOutHost();
// Team logout
PresenceSDK.getPresenceSDK(context).logOutTeam();
// logout both
PresenceSDK.getPresenceSDK(context).logOut();
```

Check Login Status

Presence SDK also provides some helper methods for checking if user is logged into any of the supported services.

// Method to check if user is logged in any of the service i.e Host or Accounts Manger
PresenceSDK.getPresenceSDK(context).isLoggedIn();
// Method to check if user is logged in Host
PresenceSDK.getPresenceSDK(context).isLoggedIntoHost();
// Method to check if user is logged in Accounts Manager
PresenceSDK.getPresenceSDK(context).isLoggedIntoTeam();

Analytics

For tracking user activity in the Presence SDK a separate class PresenceEventAnalytics is provided that lists all the user actions that are notified via local broadcast manager notifications together with the payload data.

Notification Events – You can observe these notifications to receive udpates from Presence SDK.

```
public static final class Action {

//////////////////////////////////////////////////////////////////////////////////
/
    // UI events (Start)

//////////////////////////////////////////////////////////////////////////////////
/
    public static final String ACTION_MYTICKETSCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.MYTICKETSCREENSHOWED";
    public static final String ACTION_MANAGETICKETSCREENSHOWED =
```

```
"com.ticketmaster.presencesdk.eventanalytic.action.MANAGETICKETSCREENSHOWED";
    public static final String ACTION_ADDPAYMENTINFOSCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.ADDPAYMENTINFOSCREENSHOWED";
    public static final String ACTION_REVIEWPOSTINGSCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.REVIEWPOSTINGSCREENSHOWED";
    public static final String ACTION_POSTINGCONFIRMATIONSCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.POSTINGCONFIRMATIONSCREENSHOWED";
    public static final String ACTION_CANCELPOSTINGSCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.CANCELPOSTINGSCREENSHOWED";
    public static final String ACTION_CANCELPOSTINGCONFIRMSCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.CANCELPOSTINGCONFIRMSCREENSHOWED";
    public static final String ACTION_MYTICKETBARCODESCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.MYTICKETBARCODESCREENSHOWED";
    public static final String ACTION_TICKETDETAILSSCREENSHOWED =
"com.ticketmaster.presencesdk.eventanalytic.action.TICKETDETAILSSCREENSHOWED";
    public static final String ACTION_TICKETSTUBIMAGESHARED =
"com.ticketmaster.presencesdk.eventanalytic.action.TICKETSTUBIMAGESHARED";


////////////////////////////////////////////////////////////////////////////////////////
/
    // UI events (End)

////////////////////////////////////////////////////////////////////////////////////////
/


////////////////////////////////////////////////////////////////////////////////////////
/
    // Business operation events (Start)

////////////////////////////////////////////////////////////////////////////////////////
/
    public static final String ACTION_TRANSFERINITIATED =
"com.ticketmaster.presencesdk.eventanalytic.action.TRANSFERINITIATED";
    public static final String ACTION_TRANSFERCANCELLED =
"com.ticketmaster.presencesdk.eventanalytic.action.TRANSFERCANCELLED";
    public static final String ACTION_TRANSFERACCEPTED =
"com.ticketmaster.presencesdk.eventanalytic.action.TRANSFERACCEPTED";
    public static final String ACTION_RESALEINITIATED =
"com.ticketmaster.presencesdk.eventanalytic.action.RESALEINITIATED";
    public static final String ACTION_RESALECANCELLED =
"com.ticketmaster.presencesdk.eventanalytic.action.RESALECANCELLED";
    public static final String ACTION_RESALEUPDATED =
"com.ticketmaster.presencesdk.eventanalytic.action.RESALEEDITED";

////////////////////////////////////////////////////////////////////////////////////////
/
    // Business operation events (End)

////////////////////////////////////////////////////////////////////////////////////////
/
}
```

Payload Data for the Notifications – Only relevant information is sent out with the notification.

```
public static final class Data {

    // general data for event details, and ticket details
    public static final String EVENT_ID = "event_id";
    public static final String EVENT_NAME = "event_name";
    public static final String EVENT_DATE = "event_date";
    public static final String EVENT_IMAGE_URL = "event_image_url";
    public static final String EVENT_ORDER_ID = "event_order_id";
    public static final String VENUE_NAME = "venue_name";
    public static final String VENUE_ID = "venu_id";
    public static final String CURRENT_TICKET_COUNT = "current_ticket_count";
    public static final String EVENT_ARTIST_NAME = "artist_name";
    public static final String EVENT_ARTIST_ID = "artist_id";

    // data for transfer initiate event
    public static final String INITIATE_TRANSFER_TICKET_COUNT =
"initiate_transfer_ticket_count";
    public static final String INITIATE_TRANSFER_TICKET_FACEVALUE =
"initiate_transfer_ticket_facevalue";
    public static final String INITIATE_TRANSFER_TICKET_SERIALIZABLE =
"initiate_transfer_ticket_serializable";

    // data for transfer cancel event
    public static final String CANCEL_TRANSFER_ID = "cancel_transfer_id";
    public static final String CANCEL_TRANSFER_ORDER_ID = "cancel_transfer_order_id";


    // data for resale initiate event
    public static final String INITIATE_RESALE_TICKET_COUNT =
"initiate_resale_ticket_count";
    public static final String INITIATE_RESALE_PRICE = "initiate_resale_price";
    public static final String INITIATE_RESALE_TICKET_SERIALIZABLE =
"initiate_resale_ticket_serializable";

    // data for resale update event
    public static final String UPDATE_RESALE_PRICE = "update_resale_price";
    public static final String UPDATE_RESALE_POSTING_ID = "update_resale_posting_id";

    // data for resale initiate and update events
    public static final String RESALE_BUYER_FEES = "resale_buyer_fees";
    public static final String RESALE_ORIGINAL_FACE_VALUE =
"resale_original_face_value";
    public static final String RESALE_SELLER_PAYOUT = "resale_seller_payout";
    public static final String RESALE_SELLER_FEES = "resale_seller_fees";

    // data for resale cancel event
    public static final String CANCEL_RESALE_POSTING_ID = "cancel_resale_posting_id";

    //data for sharing image
    public static final String SHARE_TICKET_IMAGE_DIR = "share_ticket_image_path";
    public static final String SHARE_TICKET_IMAGE_FILENAME =
"share_ticket_image_filename";
    public static final String SHARE_TICKET_EVENT_ID = "share_ticket_event_id";
    public static final String SHARE_TICKET_EVENT_NAME = "share_ticket_event_name";
}
```

Analytics Usage

If you want to track ACTION_MANAGETICKETSCREENSHOWED event, you should register a local broadcast listener as below:

```
IntentFilter analyticEventFilter = new IntentFilter();
analyticEventFilter.addAction(PresenceEventAnalytics.Action.ACTION_MYTICKETSCREENSHOWED)
;
LocalBroadcastManager.getInstance(MainActivity.this).registerReceiver(mAnalyticEventRece
iver, analyticEventFilter);
```

You can implement receiver mAnalyticsEventReceiver as follows:

```
private BroadcastReceiver mAnalyticEventReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {

        if
(PresenceEventAnalytics.Action.ACTION_MYTICKETSCREENSHOWED.equals(intent.getAction())) {
            Toast.makeText(MainActivity.this, "Analytic Event: My tickets screen
showed.", Toast.LENGTH_LONG).show();
        }
    }
};
```

Setting Branding Color In Compile Time

Presence sdk clients can set their own branding theme color by defining this color value in their application resource “colors.xml” file:

```
<color name="tmx_color_branding">#FFAA81</color>
```

The defined color will be displayed on all action buttons, action bars and ticket details page. If the above color variable is not defined in the client’s apk project, Tmx sdk will use a default color.