

**Presence SDK**

**Android Integration**

September 29, 2017

**Change log**

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 three aar files:

* PresenceSdk

**Supported API levels**

* API level 16 ~ 25

**Release Notes**

**Requirements**

* Supported API level 16 ~ 25

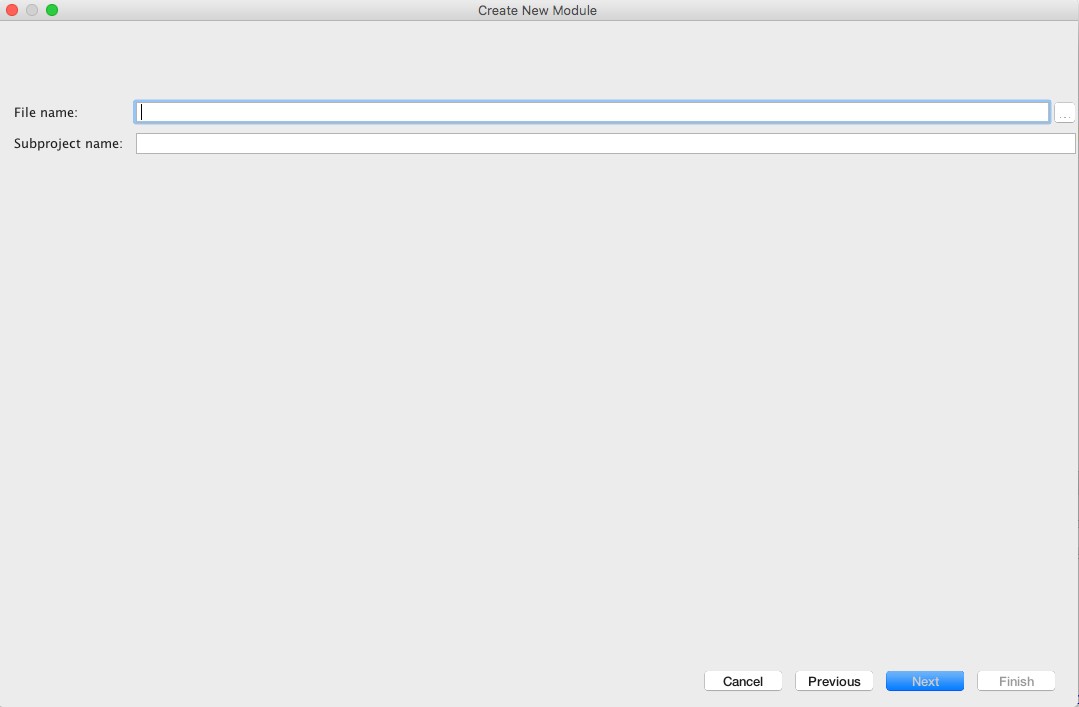
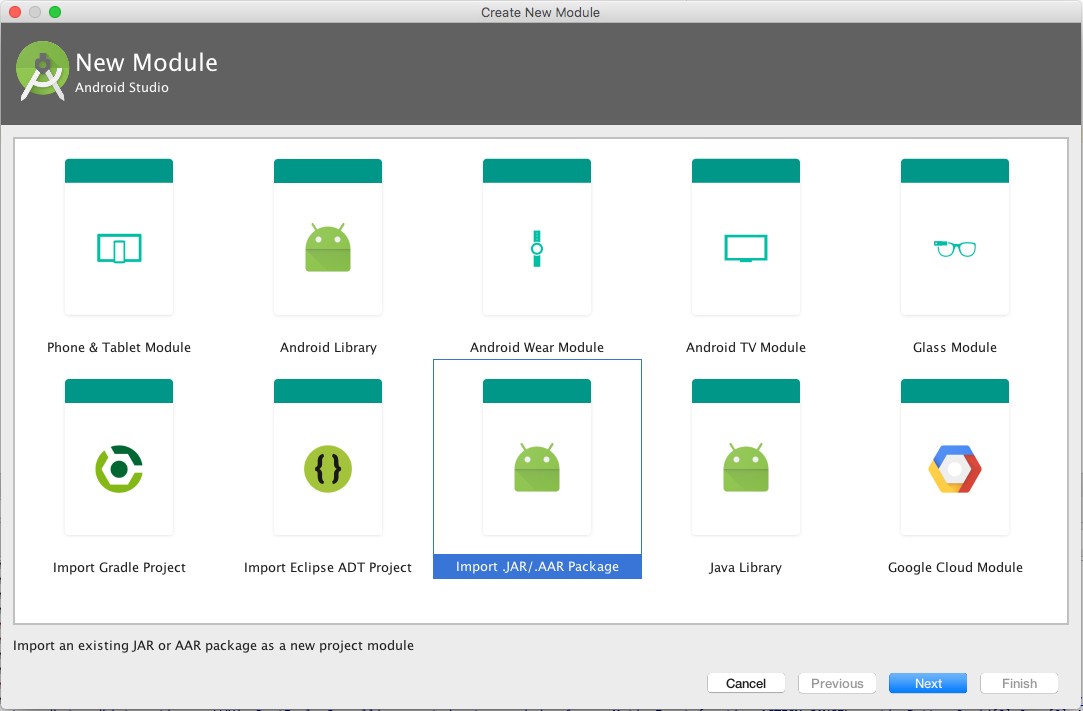
**What’s New?**

* 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.**



**Step 3. Go to your app module build gradle file and set the name**

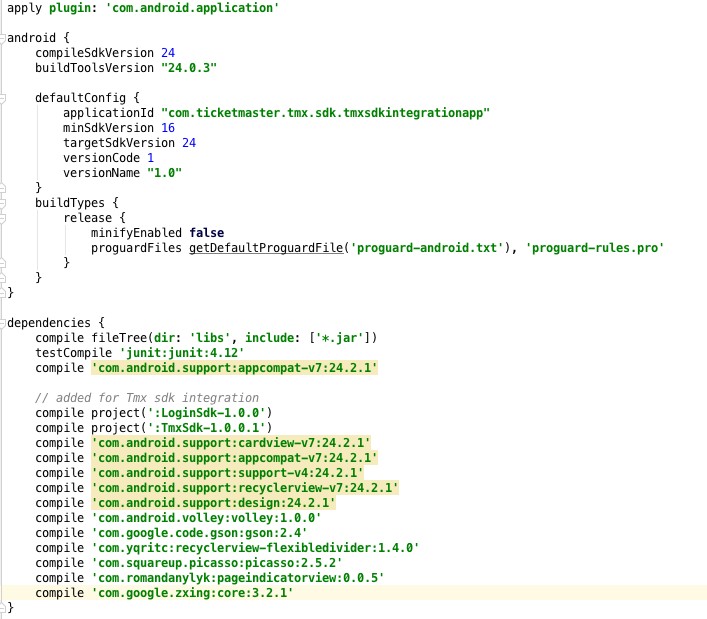
**of each aar file as compile dependencies as follows:**

compile project(‘:PresenceSDK-release-1.1.0.1’)

**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:



**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 will 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

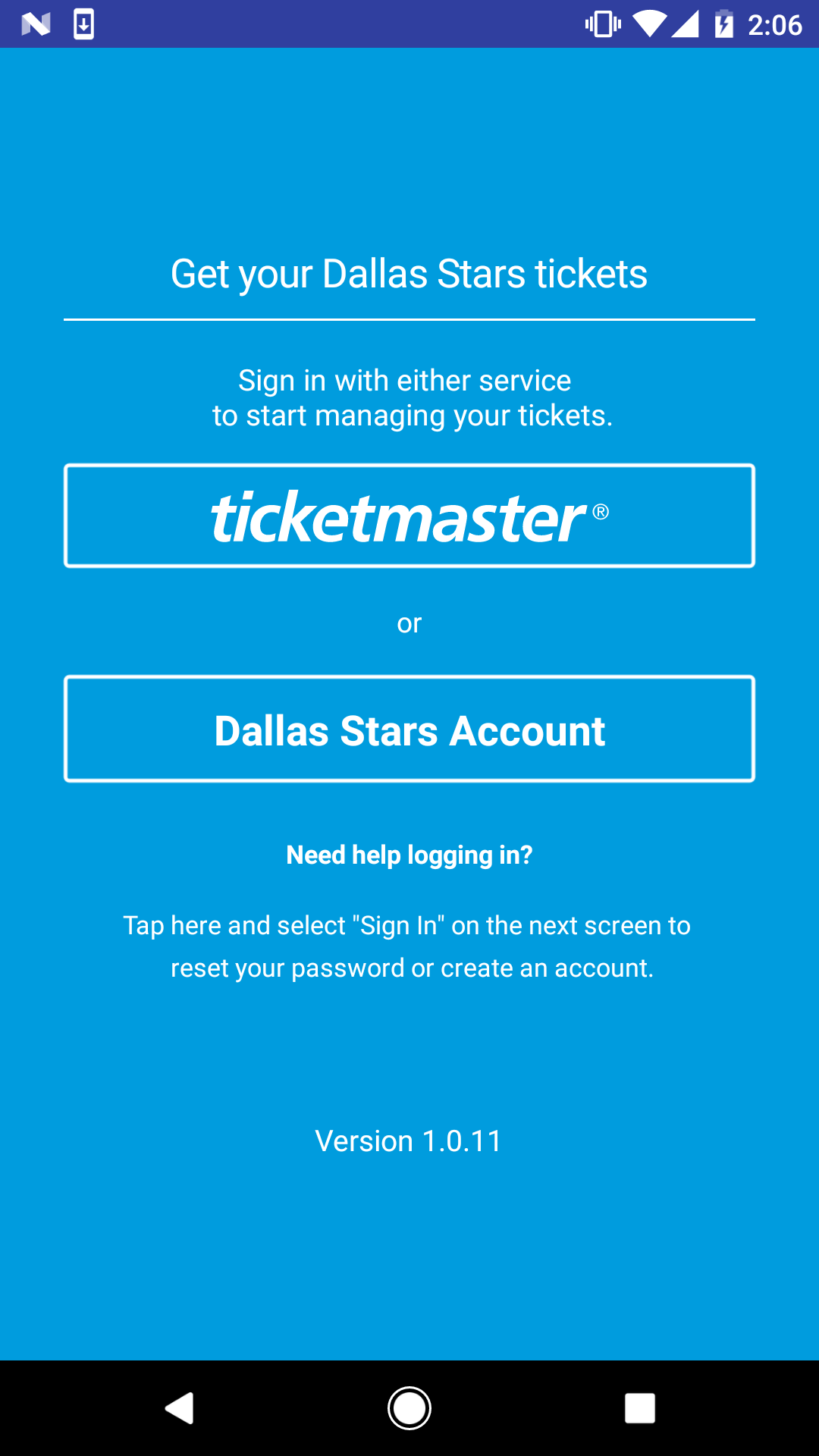
configurePresenceSDK();

// 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(**mAnalyticEventReceiver**, 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.