Wiz-Cart Android SDK

Version 1.5.6

Table of Contents

1. Prerequisites

Minimum Android SDK version: 14

Dependencies:

- com.android.support:appcompat-v7:28.0.0
- com.squareup.retrofit2:retrofit:2.5.0
- com.squareup.retrofit2:converter-gson:2.5.0
- com.google.code.gson:gson:2.8.5

2. Sample App

Namogoo WizCart library is the communication platform which provides SDK, API to add targeted incentives to e-commerce consumers. This Quick Start guide will help you run a sample project powered by Namogoo as fast as possible. To run the sample project, you will need Android Studio installed.

1. Download the sample project from the following location:

   https://github.com/namogoo/wizcart-mobile-sdk/tree/master/Android

2. Open and run in Android Studio

3. Setup and installation

To integrate Namogoo WizCart SDK to your application, you need a Namogoo WizCart account. If you do not have a WizCart account, contact your Customer Success Manager here: killian.buffard@namogoo.com . Namogoo provides you an account ID (account_id) to activate the SDK.

WizCart SDK is built and designed to be used with Android Studio. The following instructions will help you to integrate WizCart into your application:

3.1. Adding AAR

Copy WizCart-version.aar (download from- https://github.com/namogoo/wizcart-mobile-sdk/tree/master/Android) to your local machine.
In Android Studio click:

- File → New → New Module
- Select Import .JAR/.AAR Package
- Select the WizCart ARR file
- Click Finish

  Navigate to build.gradle file at the app level and add the WizCart dependency:

```
defaultConfig {
        minSdkVersion 14
    }
    repositories {
      flatDir {
          dirs 'libs'
      }
    }

    dependencies {
          implementation project(path: ':WizCart')
    }
```

* The SDK uses Gson & Retrofit2 for Rest Api call, make sure your project adds these apis as well.

3.2. AndroidManifest.xml

Set up permissions and references

The Namogoo WizCart SDK requires Internet permission from your app's AndroidManifest.xml file. This permission allows the SDK to communicate with the WizCart backend.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.namogoo.android.wizcartsample">

  <uses-permission android:name="android.permission.INTERNET" />

  <application
        …
        ...
  </application>

</manifest>
```

4. Connecting

4.1 Instantiate

Each application must register with Namogoo for a WizCart Account ID. For example:

```
Integer account_id = 25478;
```

The WizCart SDK has a primary interface WizCartClient for interacting with the WizCart service. You have to initialize the WizCartClient SDK in onCreate() method of your Application. Only one instance of WizCartClient will be instantiated:

```
WizCartClient.newInstance(Context applicationContext,String account_id);
```

* A third parameter (boolean isPreProduction) can be set for Namogoo pre production environment calls. Requires VPN connection.

An instance of the WizCartClient can be accessed via the following method:

```
WizCartClient wizCartClient = WizCartClient.getWizCartClient();
```

4.2 Listener

The WizCart SDK uses the listener pattern to push specific events to your application. You must register a WizCartListener on Activities/Fragments you wish to receive Incentive WizCart events and unregister the listener once the Activity/Fragment has no need to receive these events (Usually “register” will be done in OnResume and “unregister” in OnPause).

```
wizCartClient.registerWizCartListenter(WizCartListener instance);
wizCartClient.unregisterWizCartListenter(WizCartListener instance);
```

4.3 Connect and authenticate

4.3.1 WizCart client ids

Must
In order to calculate incentives for a client, you must pass a unique client id. WizCart will accept a unique String as a Client ID (UUID only), which we call as wizCartClientId. So you can use any new or existing User Management system.

params:

wizCartClientId	Unique identifier for a logged in user. Should be null for non logged in users.
tempWizCartClientId	Unique identifier for a non logged in user. If all users are always logged in. Can be the same as *wizCartClientId*. If null the SDk will generate a temp user ID.
retailVisitor	Unique identifier that remains the same for the user regardless of his log in state.Can be the same as *tempWizCartClientId*. If null will be the same as *wizCartClientId/tempUserId*.

```
wizCartClient.setWizCartClientIds(String wizCartClientId, String
tempWizCartClientId, String retailerVisitor);
```

4.3.2 Client traffic sources

Must
In order to determine if the client is part of the WizCart incentive program and which

“bucket” of incentives should be used for the end client, traffic sources should be passed to the wizCartClient.

params:

TrafficSources	List of strings that determine the user group and type, for example: “Personali”, “User40”

```
wizCartClient.setTrafficSources(ArrayList<String> trafficSources);
```

4.4 WizCartListener

WizCartListener should be registered within the WizCartClient object. It's used to get response objects when calling WizCart’s GetIncentive & Tracking API calls.

```
wizCartListener = new WizCartListener() {
   @Override
   public void incentiveResponse(WizCartIncentiveResponse
wizCartIncentiveResponse, Exception exception) {
       if (wizCartIncentiveResponse != null) {
           //display incentive according to the wizCartIncentiveResponse;
       } else {
           //something went wrong, check the Exception object
       }
   }

   @Override
   public void trackingResponse(WizCartTrackingResponse trackingResponse) {
       //WizCartTrackingResponse can be OK or Error
```

```
};
wizCart.registerWizCartListener(wizCartListener);
```

5. API

5.1 Track client journey

The WizCart ML requires information about the client journey in order to calculate the WizCart incentive. On application load at least one GetIncentive api call should be performed before calling the tracking apis.

The ‘Cart’ and ‘Page type’ information should be sent with each of the events below:

Cart	The current client cart state.
Page type	The current PageType the client is viewing.

5.1.1 Page View

When to call:
Each time a client changes a PageType in his journey or a view is resumed.

params:

displayedWizCartIncentive	boolean if the WizCart incentive was visible for the client
wizCartCouponCode	What is the current wizCart coupon code that is eligible by the client. If the client is not eligible for a WizCart coupon set to null
Referrer	What application started the host app * First PageView only
pageReferrer	What was the Uri used to start the host application * First PageView only

```
wizCartClient.trackPageView(PageType pageType, Cart cart, boolean
displayedWizCartIncentive, String wizCartCouponCode, String referrer, String
pageReferrer);

wizCartClient.trackPageView(PageType pageType, Cart cart, boolean
displayedWizCartIncentive, String wizCartCouponCode);
```

5.1.2 Incentive view Minimized/Maximized

When to call:
For the relevant users (who received incentive), when the incentive is first seen and additionally every time a client engages with the incentive view (banner) minimize/maximize capability.

params:

wizCartCouponCode	What is the current wizCart coupon code that is eligible by the client. If the client is not eligible for a WizCart coupon set to null

```
wizCartClient.trackMinimizeIncentive(PageType pageType, Cart cart, String
wizCartCouponCode);

wizCartClient.trackMaximizeIncentive(PageType pageType, Cart cart, String
wizCartCouponCode);
```

5.1.3 WizCart coupon copied

When to call:
When the coupon is copied to the clipboard by the client.

params:

wizCartCouponCode	What is the current wizCart coupon code that is eligible by the client. If the client is not eligible for a WizCart coupon set to null

```
wizCartClient.trackCopyCoupon(PageType pageType, Cart cart, String
wizCartCouponCode);
```

5.1.4 WizCart coupon added to cart

When to call:
When the coupon is used by the client during the purchase process.

params:

wizCartCouponCode	What is the current wizCart coupon code that is eligible by the client. If the client is not eligible for a WizCart coupon set to null

```
wizCartClient.trackInsertCouponToCart(PageType pageType, Cart cart, String
wizCartCouponCode);
```

5.1.5 Client Cart updated

When to call:
Every time the client adds or removes items from the cart in the host application.

params:

product	The information about the product that was modified in the cart including the new quantity (0 if removed from cart)

```
wizCartClient.trackAddedItemToCart(PageType pageType, Cart cart, Product product);
wizCartClient.trackRemovedItemFromCart(PageType pageType, Cart cart, Product
product);
```

5.1.6 Purchase completed

When to call:
When the user completed a purchase.

params:

cartDiscountValueOfWizCart	If a WizCart incentive was used, insert the currency value of it
wizCartCouponCode	What is the current wizCart coupon code that is eligible by the client. If the client is not eligible for a WizCart coupon set to null
addedOtherIncentivetypes	True if other incentives were used in this purchase, free shipping for example.
appliedCoupons	A list of coupons used in the purchase process. If WizCart Coupon was used it should be in the list.
trackingId	The tracking id for this purchase

```
wizCartClient.trackCompletedPurchase(PageType pageType, Cart cart, float
cartDiscountValueOfWizCart, String wizCartCouponCode, boolean
addedOtherIncentiveTypes, ArrayList<String> appliedCoupons, String trackingId);
```

5.1.7 Client cart cleared

When to call:
Every time the client adds or removes items from the cart in the host application.

params:

wizCartCouponCode	What is the current wizCart coupon code that is eligible by the client. If the client is not eligible for a WizCart coupon set to null

```
wizCartClient.trackClearedCart(PageType pageType, Cart cart, String
wizCartCouponCode);
```

5.2 Get client incentive

getIncentive is called In order to receive information if the client is eligible for an incentive at the current journey snapshot.

The host application should call this method each time the client changes view or changes the content of his cart.
The getIncentive call is done asynchronously and the result will be returned via the WizCartListener.

params:

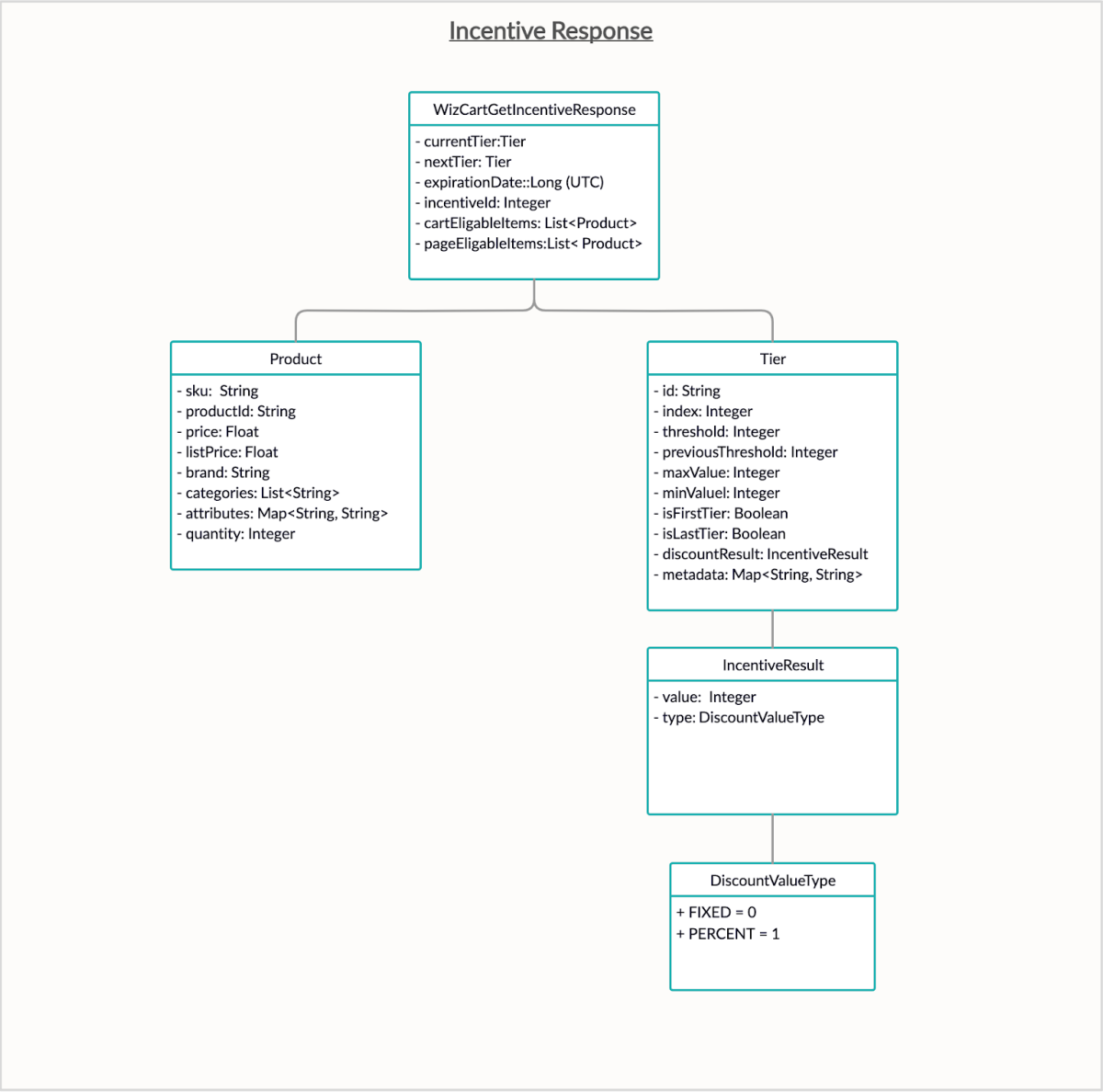
Cart	The current client cart state.

Page type	The current PageType the client is viewing.

```
wizCartClient.getIncentive(Cart cart, PageType pageType);
```

6. Models

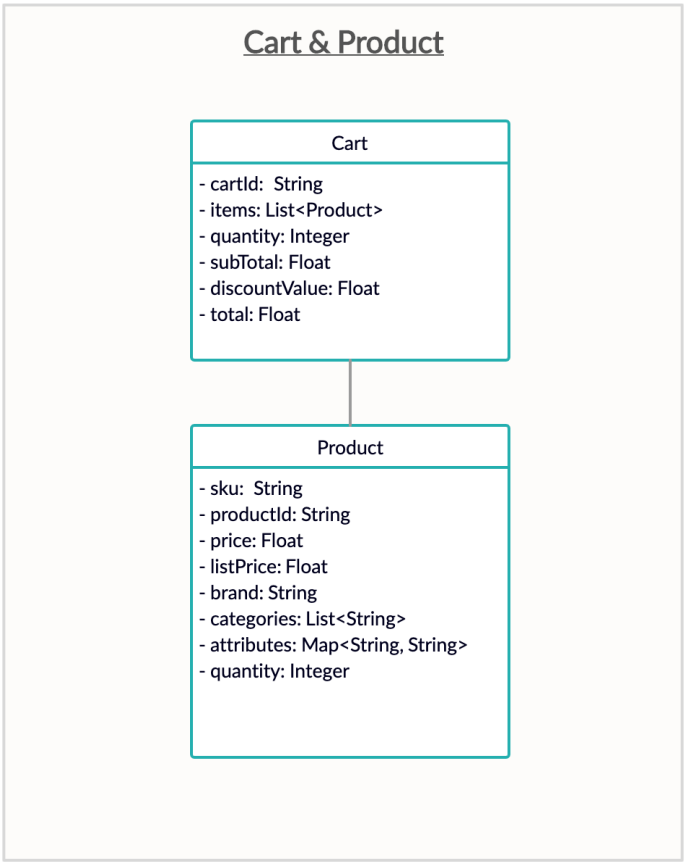
6.1 WizCartIncentiveResponse



6.2 Cart & Product

Cart

1. cartId - The host app cart Id
2. items - list of all products in the cart
3. quantity - the total number quantities of all items
4. subtotal - the price of the cart before tax/discount/shipping...
5. discountValue - the total discount the client receives for this cart from all incentives
6. total - the final amount the client will spend on the cart



Product

1. sku - sku of the product
2. productId - Product identifier of the product / can be the same as sku

3. price - The price of a single unit of the product after discounts (or same as list price)
4. listPrice - The price of a single unit of the product before discounts
5. brand - the brand name of the product
6. categories - A list of categories of the product (Baked goods / Meat / Clothing...)
7. attributes - A Map of descriptive attributes of the product (color / size ...)
8. quantity - The quantity of the product the client requests

7. Enums

7.1 PageType

The various pages that the WizCart should be called (Tracking & Incentive):

```
public enum PageType {
  PRODUCT_PAGE,
  HOME_PAGE,
  LISTING_PAGE,
  CART_PAGE,
  SEARCH_PAGE,
  REGISTRATION_PAGE,
  PAYMENT_PAGE,
  CONFIRMATION_PAGE,
  QUICK_VIEW,
  PRODUCT_SET_PAGE,
  PRODUCT_SET_QUICK_VIEW,
  CATEGORY_PAGE,
  SUB_CATEGORY_PAGE
}
```