Mopub

You can use Dispply as a `Network` in **Mopub's** Mediation platform.

1. Setup SDKs

- Integrate with Mopub SDK (https://github.com/mopub/mopub-android-sdk/wiki/Getting-Started)

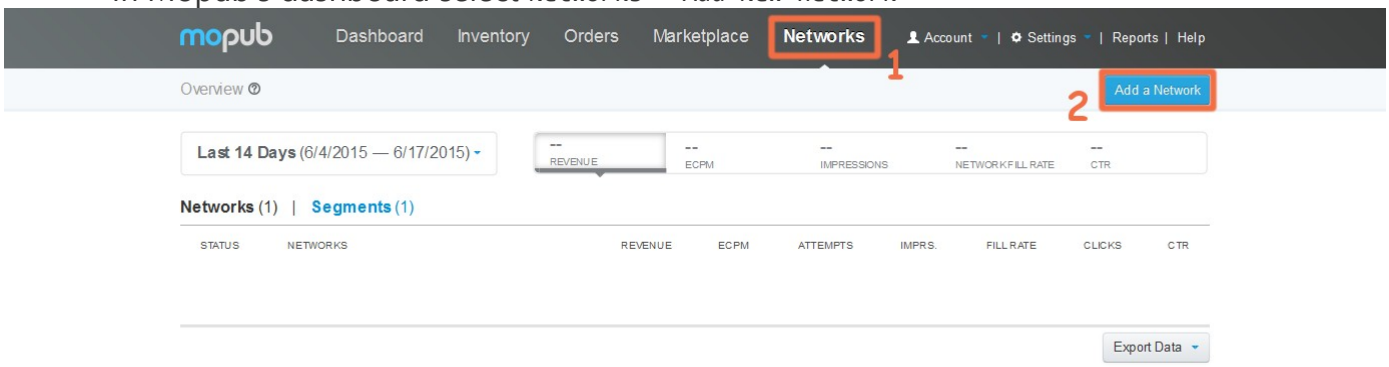
- Install Dispply SDK (Download SDK)

  More info how to install Dispply SDK on Integration and API documentation

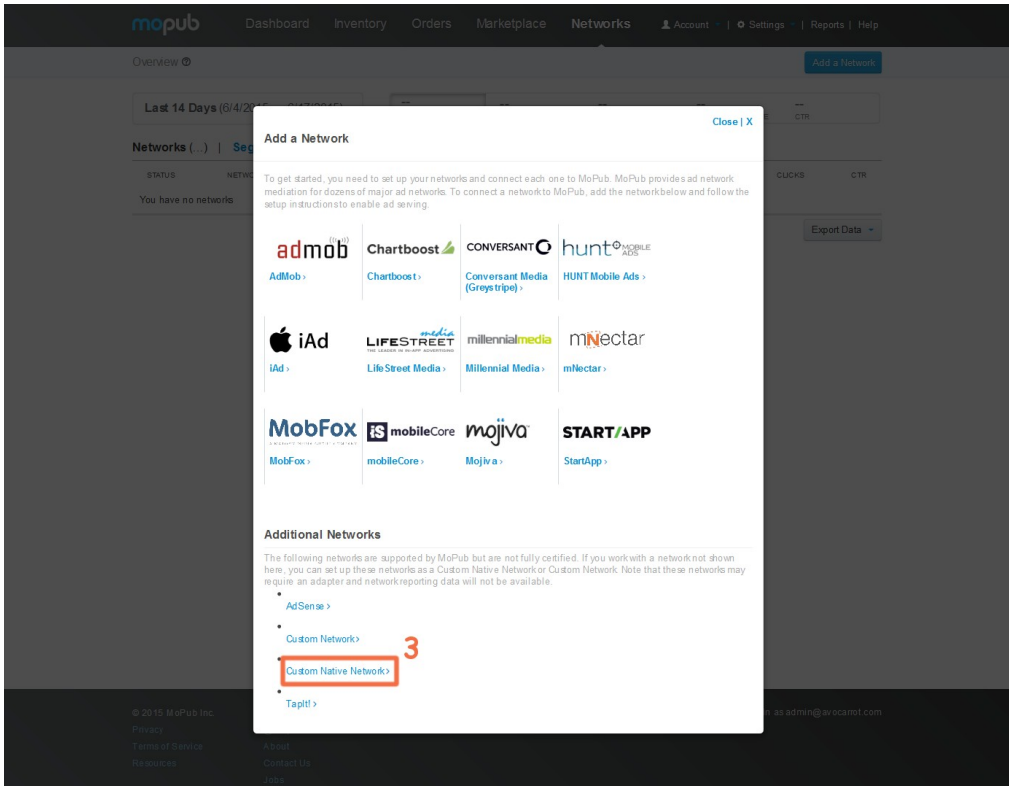
2. Setup Mopub Dashboard

Create an "Dispply" `Network` in Mopub's dashboard and connect it to your Ad Units.

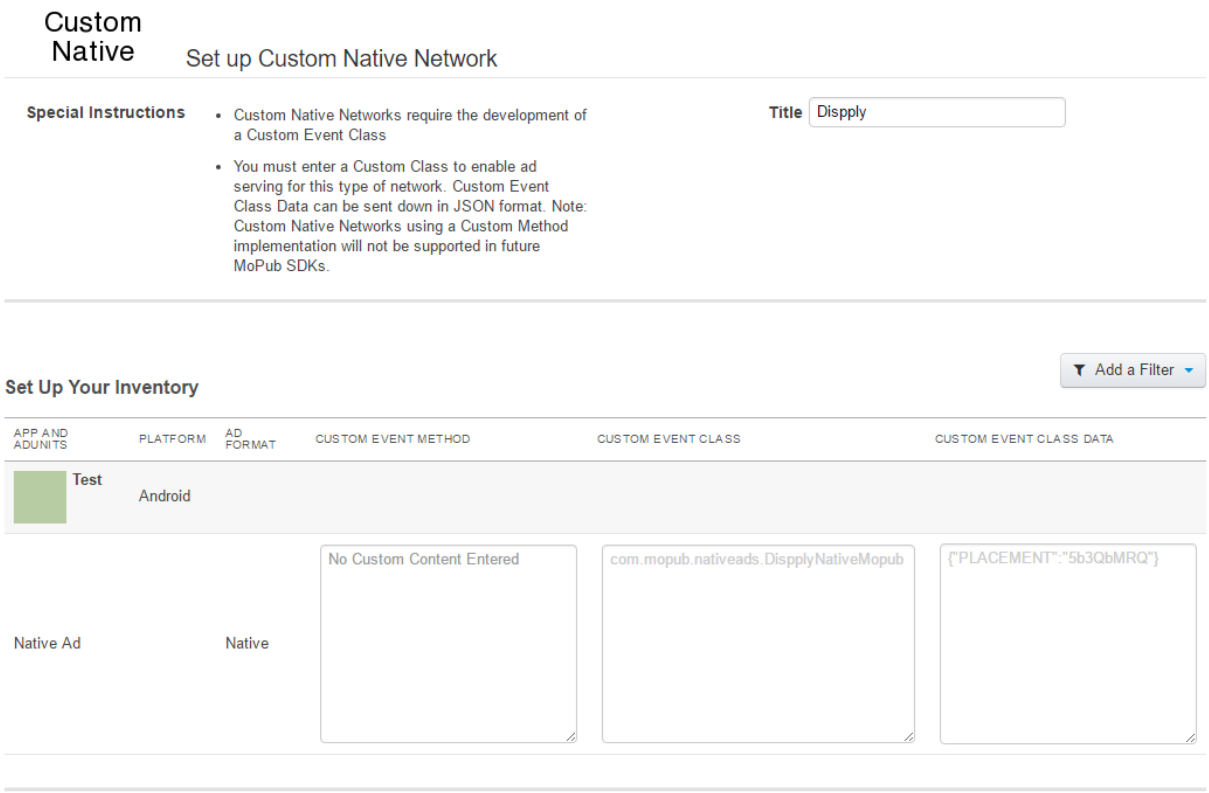
- In Mopub's dashboard select `Networks > Add New network`



- Then select `Custom Native Network`



•Complete the fields accordingly to the Ad Unit that you want to use



Custom Event Class

`com.mopub.nativeads.DispplyNativeMopub`

Custom Event Class Data

`{"PLACEMENT":"<YOUR PLACEMENT>"}`

You can use the test placement "5b3QbMRQ"

2. Add adapter in your project

Create package "`com.mopub.nativeads`" in your project and put this class in there:

`DispplyNativeMopub.java`

```
package com.mopub.nativeads;

import android.app.Activity;
import android.os.AsyncTask;
import android.support.annotation.NonNull;
import android.util.Log;
import android.view.View;
import com.dispply.lib.sdk.nativeads.Ad;
import com.dispply.lib.sdk.nativeads.AdListener;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
public class DispplyNativeMopub extends CustomEventNative {
    private static final String PLACEMENT_KEY = "PLACEMENT";
    com.dispply.lib.sdk.nativeads.NativeAd nativeAd;
    @Override
    protected void loadNativeAd(final @NonNull Activity activity, final @NonNull
CustomEventNativeListener customEventNativeListener, @NonNull Map<String,
Object> localExtras, @NonNull Map<String, String> serverExtras) {
        final String placement;
        if ((serverExtras != null) && serverExtras.containsKey(PLACEMENT_KEY)) {
            placement = serverExtras.get(PLACEMENT_KEY);
        } else {

customEventNativeListener.onNativeAdFailed(NativeErrorCode.NATIVE_ADAPTER_CONFIG
URATION_ERROR);
            return;
        }
        nativeAd = new com.dispply.lib.sdk.nativeads.NativeAd(activity,
placement); //Native AD constructor
        nativeAd.setAdListener(new AdListener() { // Add Listeners
            @Override
            public void onAdLoaded(Ad ad) {
                customEventNativeListener.onNativeAdLoaded(new
DispplyNativeAd(ad, nativeAd, activity));
            }
            @Override
            public void onError(Ad nativeAd, String error) { // Called when load
is fail

customEventNativeListener.onNativeAdFailed(NativeErrorCode.EMPTY_AD_RESPONSE);
            }
            @Override
            public void onAdClicked() { // Called when user click on AD
                Log.wtf("TAG", "AD Clicked");
            }
        });
        nativeAd.loadAd();
    }
    class DispplyNativeAd extends StaticNativeAd {
```

```
        final Ad dispplyModel;
        final com.dispply.lib.sdk.nativeads.NativeAd nativeAd;
        final ImpressionTracker impressionTracker;
        final NativeClickHandler nativeClickHandler;
        final Activity activity;
        public DispplyNativeAd(@NonNull Ad customModel,
com.dispply.lib.sdk.nativeads.NativeAd nativeAd, Activity activity) {
            dispplyModel = customModel;
            this.nativeAd = nativeAd;
            this.activity = activity;
            impressionTracker = new ImpressionTracker(activity);
            nativeClickHandler = new NativeClickHandler(activity);
            setIconImageUrl(dispplyModel.getIcon_url());
            setMainImageUrl(dispplyModel.getImage_url());
            setTitle(dispplyModel.getTitle());
            setText(dispplyModel.getDescription());
            setClickDestinationUrl(dispplyModel.getClickUrl());
            for (Ad.Tracker tracker : dispplyModel.getTrackers())
                if (tracker.getType().equals("impression")) {
                    addImpressionTracker(tracker.getUrl());
                }
        }
        @Override
        public void prepare(final View view) {
            impressionTracker.addView(view, this);
            nativeClickHandler.setOnClickListener(view, this);
        }
        @Override
        public void recordImpression(final View view) {
            notifyAdImpressed();
            for (Ad.Tracker tracker : dispplyModel.getTrackers())
                if (tracker.getType().equals("impression")) {
                    new LoadUrlTask().execute(tracker.getUrl());
                }
        }
        @Override
        public void handleClick(final View view) {
            notifyAdClicked();
            nativeClickHandler.openClickDestinationUrl(getClickDestinationUrl(),
view);
            if (dispplyModel.getClickUrl() != null)
                new LoadUrlTask().execute(dispplyModel.getClickUrl());
        }
        private class LoadUrlTask extends AsyncTask<String, Void, String> {
            String userAgent;
            public LoadUrlTask() {
                userAgent =
com.dispply.lib.sdk.Util.getDefaultUserAgentString(activity);
            }
            @Override
            protected String doInBackground(String... urls) {
                String loadingUrl = urls[0];
                URL url = null;
                try {
                    url = new URL(loadingUrl);
                } catch (MalformedURLException e) {
                    return (loadingUrl != null) ? loadingUrl : "";
                }
                com.dispply.lib.sdk.Log.d("Checking URL redirect:" +
loadingUrl);
                int statusCode = -1;
                HttpURLConnection connection = null;
                String nextLocation = url.toString();
                Set<String> redirectLocations = new HashSet<String>();
```

```
                redirectLocations.add(nextLocation);
                try {
                    do {
                        connection = (HttpURLConnection) url.openConnection();
                        connection.setRequestProperty("User-Agent",
                                userAgent);
                        connection.setInstanceFollowRedirects(false);
                        statusCode = connection.getResponseCode();
                        if (statusCode == HttpURLConnection.HTTP_OK) {
                            connection.disconnect();
                            break;
                        } else {
                            nextLocation =
connection.getHeaderField("location");
                            connection.disconnect();
                            if (!redirectLocations.add(nextLocation)) {
                                com.dispply.lib.sdk.Log.d("URL redirect cycle
detected");
                                return "";
                            }
                            url = new URL(nextLocation);
                        }
                    }
                    while (statusCode == HttpURLConnection.HTTP_MOVED_TEMP ||
statusCode == HttpURLConnection.HTTP_MOVED_PERM
                            || statusCode == HttpURLConnection.HTTP_UNAVAILABLE
                            || statusCode == HttpURLConnection.HTTP_SEE_OTHER);
                } catch (IOException e) {
                    return (nextLocation != null) ? nextLocation : "";
                } finally {
                    if (connection != null)
                        connection.disconnect();
                }
                return nextLocation;
            }
            @Override
            protected void onPostExecute(String url) {
            }
        }
    }
}
```

This is your adapter. Now you can use Mopub as usual.