

GroundTruth Display SDK v1.0.4 – Android

Table of Contents

[Overview](#)

[Integration](#)

Stand Alone

- [Banner](#)
- [Interstitial](#)
- [Video](#)

Mediation Networks

- [DFP](#)
- [AdMob](#)
- [MoPub](#)

Overview

The GroundTruth advertising network is powered by patented Location Verification and Blueprints™ technologies. These technologies help us detect quality user location signals that drive the delivery of high-performance ad inventory to the right user at the right time for the purpose of monetizing your Mobile App.

Mobile App publishers can access this ad inventory with the GroundTruth Display SDK. The GroundTruth Display SDK is designed to be embedded within a Mobile App. It provides the necessary modules to serve and render ads and support popular mediation networks.

This guide contains information on how to integrate your application with the GroundTruth Display SDK. It is also designed to work with popular GroundTruth mediation frameworks, namely DFP, Admob, and Mopub.

Capabilities

- Support for Standard Banner Ads
- Support for Interstitial Ads
- Support for MRAID v2.0 (Mobile Rich Media Ad Interface Definition)
- Support for VAST v2.0 Linear (Video Ad Serving Template)
- Mediation support to fall back on the following ad networks:
 - DFP
 - Admob
 - Mopub

Requirements

- Android API 19–25
- Publisher key provided by GroundTruth

Data Collection

The Display SDK requires the most accurate information possible to deliver the right ad to the user. Some information is collected automatically, while other information is passed in by the application.

1) Automatically collected data

- User Agent
- Location (when available)
 - Latitude
 - Longitude
 - Altitude
 - Horizontal accuracy
- Device ID
- IP address
- Application info
 - Name
 - Bundle
 - Version
- Device info
 - Language
 - Operating system
 - Operating system version

2) Application provided data

To prevent advertising specific products or services inside the App, the GroundTruth Display SDK supports blocking per RTB specification.

- Blocked Categories (you can block categories of advertisers by providing a comma-delimited list of IAB codes or the products or services or companies)
- Blocked Advertisers (you can block advertisers by providing a comma-delimited list of top level domain)

You should provide a form to collect the following information:

- Gender
- Age – We do not deliver any ads to users 13 years old or younger, nor do we deliver any alcohol ads to users unless they are 21+
- Location – Alternative location information when you the publisher refuse to enable GPS location acquisition. You can choose to send the user's zip code, the city along with IP address which gets collected automatically.

COPPA Compliance

GroundTruth is COPPA compliant, and as such we do not serve advertisements to children under the age of 13 or those who opt-out using iOS and Android Do-not-track features. Help us by collecting and sending the user age or date of birth.

Integration Instructions

Step 1 – Request an access key

To make Display SDK work, you will need a designated access key. If your partner manager has not yet supplied this please send an email to sdk@groundtruth.com.

Step 2 – Import sdk AAR file

In the build.gradle of the whole project, please add following maven url:

```
allprojects {  
    repositories {  
        jcenter()  
        maven {  
            url "http://groundtruth.bintray.com/GroundTruth"  
        }  
    }  
}
```

Grab the AAR from Maven Central by adding it as a dependency in your build.gradle file:

```
dependencies {  
    ...  
    compile 'com.groundtruth.sdk.displaysdk:displaysdk:1.0.+'  
    ...  
}
```

Step 3 – Add required activities to AndroidManifest.xml

```
<activity android:name="com.xad.sdk.vast.activity.VASTActivity"/>  
<activity android:name="com.xad.sdk.mraid.VideoPlayerActivity"/>
```

Step 4 – Ask for location permission

GroundTruth determines the ad that is most relevant and higher performing with location data like geo-coordinates, horizontal accuracy, and IP address.

For Android API Level 23 and above

You need to ask location permission while the app is running, not when they install the app. Please refer <https://developer.android.com/training/permissions/requesting.html> to request permission.

For Android API Level 22 and below

You're good to go.

Step 5 – Choose to use Display SDK as standalone or integrate with DFP/AdMob/MoPub

Option 1 – Standalone

Banner

Step 1 – Initialize instance of DisplaySDK

You should initialize DisplaySDK's shared instance before you add any ad(i.e. banner, interstitial or video) on your layout (e.g. initialize it in the onCreate method of the activity)

```
DisplaySdk.sharedInstance().init(this);
```

Step 2a – Add the banner view in XML Layout

Insert the following where you wish to place the banner in your activity's layout:

```
<com.xad.sdk.BannerView
    xmlns:gt="http://schemas.android.com/apk/res-auto"
    android:layout_width="320dp"
    android:layout_height="50dp"
    android:id="@+id/banner_view"
    gt:AdSize="BANNER"
    gt:AccessKey="your_access_key"
    gt:AdInterval="Long"/>
```

Step 2b – Add the banner view programmatically

Alternatively, you can also choose to instantiate and position the view programmatically by adding the following code in the onCreate method:

```
BannerView bannerView = new BannerView(this, AdSize.BANNER, "your_access_key");
```

Step 3 – Configure the banner view

```
public class MainActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // Birthday is April 15, 1988
        AdRequest adRequest = new AdRequest.Builder()
            .setBirthday(1988, AdRequest.ARP, 15)
            .setGender(AdRequest.Gender.MALE)
            .build();
        this.bannerView.setAdRequest(adRequest);
        this.bannerView.setRefresh(RefreshInterval.MEDIUM);
        ...
    }
}
```

Step 4 – Set and implement Ad Listener(Only when used as standalone)

To receive event callbacks from the Banner, you must set and implement AdListener.

```

public class MainActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        this.bannerView.setAdListener(new BannerViewListener() {
            @Override
            public void onAdFetchFailed(BannerView bannerView, ErrorCode code)
{
                //Called when error occurs and no ad filled in banner view
                if(code == ErrorCode.NO_INVENTORY) {
                    //No ad match for current request
                } else {
                    //Handle other error codes
                    ...
                }
            }

            @Override
            public void onAdClosed(BannerView bannerView) {
                //Called when the user is about to return to the activity after
clicking on an ad by clicking close button or back button
            }

            @Override
            public void onAdOpened(BannerView bannerView) {
                //Called when an ad opens an overlay that covers the screen
(e.g. landing page or expanded ad).
            }

            @Override
            public void onAdLeftApplication(BannerView bannerView) {
                //Called when an ad leaves the application (e.g. to make a
phone call or go to the native browser).
            }

            @Override
            public void onAdLoaded(BannerView bannerView) {
                //Called when an ad is received.
            }
        });
        ...
    }
}

```

Step 5 – Request an ad

You will need to call the `loadAd()` method to tell the network layer to request an ad from the server.

```
this.bannerView.loadAd();
```

Step 6 – BannerView lifecycle

You should call all lifecycle callbacks(of both bannerView and DisplaySdk) in your activity's lifecycle callbacks as well, as below:

```
public class MainActivity {

    @Override
    public void onPause() {
        ...
        if (bannerView != null) {
            bannerView.pause();
        }
        DisplaySdk.sharedInstance().pause();
        super.onPause();
        ...
    }

    @Override
    public void onResume() {
        ...
        super.onResume();
        DisplaySdk.sharedInstance().resume();
        if (bannerView != null) {
            bannerView.resume();
        }
        ...
    }

    @Override
    public void onDestroy() {
        ...
        if (bannerView != null) {
            bannerView.destroy();
        }
        DisplaySdk.sharedInstance().destroy();
        super.onDestroy();
        ...
    }
}
```

Interstitial Ad

Step 1 – Initialize instance of DisplaySDK

You should initialize DisplaySDK's shared instance before you add any ad(i.e. banner, interstitial or video) on your layout (e.g. initialize it in the onCreate method of the activity)

```
DisplaySdk.sharedInstance().init(this);
```

Step 2 – Add an interstitial view programmatically

Interstitial ads do not require you to modify the XML layout. You will need to instantiate and position the view programmatically by adding the interstitial to your activity.

```
InterstitialAd interstitial = new InterstitialAd(this, "your_access_key");
```

Step 3 – Configure the interstitial

```
public class MainActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // Birthday is April 15, 1988
        AdRequest adRequest = new AdRequest.Builder()
            .setBirthday(1988, AdRequest.ARP, 15)
            .setGender(AdRequest.Gender.MALE)
            .build();
        this.interstitial.setAdRequest(adRequest);
        ...
    }
}
```

Step 4 – Set and implement Ad Listener(Only when used as standalone)

To receive event callbacks from the interstitial, you must set and implement AdListener.

```
public class MainActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        this.interstitial.setAdListener(new InterstitialAdListener() {
            @Override
            public void onAdLoaded(InterstitialAd interstitialAd) {
                //Called when an ad is received.
            }

            @Override
            public void onAdFetchFailed(InterstitialAd interstitialAd,
                ErrorCode code) {
                //Called when error occurs and no ad filled in interstitial
                if(code == ErrorCode.NO_INVENTORY) {
                    //No ad match for current request
                } else {
                    //Handle other error codes
                    ...
                }
            }

            @Override
            public void onInterstitialShown(InterstitialAd interstitialAd) {
```

```

        //Called after interstitial is showed and overlay the full
screen, typically called after interstitial.show() is called
    }

    @Override
    public void onInterstitialFailedToShow(InterstitialAd
interstitialAd) {
        //Called after interstitial has been failed to show when
calling interstitial.show()
    }

    @Override
    public void onAdClosed(InterstitialAd interstitialAd) {
        //Called when the user is about to return to the activity after
clicking close button or back button
    }

    @Override
    public void onAdOpened(InterstitialAd interstitialAd) {
        //Called when an ad opens another overlay that covers the
screen (usually is the landing page).
    }

    @Override
    public void onAdLeftApplication(InterstitialAd interstitialAd) {
        //Called when an ad leaves the application (e.g. to make a
phone call or go to the native browser).
    }
    });
    ...
}
}

```

Step 5 – Request an ad

You will need to call the `loadAd()` method to tell the network layer to request an ad from the server.

```
this.interstitial.loadAd();
```

Step 6 – Show the interstitial

You will need to call the `show()` method to present the interstitial ad to users. If you want to show the interstitial once it is ready, you can make the call in `AdListener`'s `onAdLoaded(InterstitialAd interstitialAd)` callback, see detail in Step 3.

```
this.interstitial.show();
```

Step 7 – Interstitial lifecycle

You should call all lifecycle callbacks(of both interstitial and `DisplaySdk`) in your activity's lifecycle callbacks as well, as below:


```

public class MainActivity {

    @Override
    public void onPause() {
        ...
        if (interstitial != null) {
            interstitial.pause();
        }
        DisplaySdk.sharedInstance().pause();
        super.onPause();
        ...
    }

    @Override
    public void onResume() {
        ...
        super.onResume();
        DisplaySdk.sharedInstance().resume();
        if (interstitial != null) {
            interstitial.resume();
        }
        ...
    }

    @Override
    public void onDestroy() {
        ...
        if (interstitial != null) {
            interstitial.destroy();
        }
        DisplaySdk.sharedInstance().destroy();
        super.onDestroy();
        ...
    }
}

```

Video Ad

Step 1 – Initialize instance of DisplaySDK

You should initialize DisplaySDK's shared instance before you add any ad(i.e. banner, interstitial or video) on your layout (e.g. initialize it in the onCreate method of the activity)

```
DisplaySdk.sharedInstance().init(this);
```

Step 2 – Add a video ad programmatically

Video ads do not require you to modify the XML layout. You will need to instantiate and position the view programmatically by adding the video ad to your activity.

Note: you should put video min/max durations to indicate the server to return video ad with the

proper length.

```
VideoAd video = new VideoAd(this, 10, 30, "your_access_key");
```

Step 3 – Configure the video ad

```
public class MainActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // Birthday is April 15, 1988
        AdRequest adRequest = new AdRequest.Builder()
            .setBirthday(1988, AdRequest.ARP, 15)
            .setGender(AdRequest.Gender.MALE)
            .build();
        this.video.setAdRequest(adRequest);
        ...
    }
}
```

Step 4 – Set and implement Ad Listener(Only when used as standalone)

To receive event callbacks from the video ad, you must set and implement AdListener.

```
public class MainActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        this.video.setAdListener(new VideoAdListener() {
            @Override
            public void onVideoFailedToLoad(VideoAd videoAd, ErrorCode
errorCode) {
                //Called when error occurs and no ad filled in video ad
                if(code == ErrorCode.NO_INVENTORY) {
                    //No ad match for current request
                } else {
                    //Handle other error codes
                    ...
                }
            }

            @Override
            public void onVideoStarted(VideoAd videoAd) {
                //Called when the video starts to play
            }

            @Override
            public void onPlaybackError(VideoAd videoAd) {
                //Called when the playback is interrupted or stopped by any
errors
            }
        })
    }
}
```

```

        @Override
        public void onVideoClosed(VideoAd videoAd) {
            //Called when user close the video ad by clicking close button
or back button
        }

        @Override
        public void onVideoClicked(VideoAd videoAd) {
            //Called when user click anywhere on the video and show the
landing page
        }

        @Override
        public void onVideoCompleted(VideoAd videoAd) {
            //Called when playback is completed
        }

        @Override
        public void onVideoLoadSuccess(VideoAd videoAd) {
            //Called when an ad is received.
        }
    });
    ...
}
}

```

Step 5 – Request an ad

You will need to call the `loadAd()` method to tell the network layer to request an ad from the server.

```
this.video.loadAd();
```

Step 6 – Play the video

You will need to call the `play()` method to start to play the ad. If you want to play the ad once it is ready, you can make the call in `AdListener's onVideoLoadSuccess(VideoAd videoAd)` callback, see detail in Step 3.

```
this.video.play();
```

Step 7 – Video ad lifecycle

You should call all lifecycle callbacks (of both video ad and `DisplaySdk`) in your activity's lifecycle callbacks as well, as below:

```

public class MainActivity {

    @Override
    public void onPause() {
        ...
        if (video != null) {
            video.pause();
        }
        DisplaySdk.sharedInstance().pause();
        super.onPause();
        ...
    }

    @Override
    public void onResume() {
        ...
        super.onResume();
        DisplaySdk.sharedInstance().resume();
        if (video != null) {
            video.resume();
        }
        ...
    }

    @Override
    public void onDestroy() {
        ...
        if (video != null) {
            video.destroy();
        }
        DisplaySdk.sharedInstance().destroy();
        super.onDestroy();
        ...
    }
}

```

Option 2 – DFP

Step 1 – Import sdk AAR file

In the build.gradle of the whole project, please add following maven url:

```

allprojects {
    repositories {
        jcenter()
        maven {
            url "http://groundtruth.bintray.com/GroundTruth"
        }
    }
}

```

Grab the AAR from Maven Central by adding it as a dependency in your build.gradle file:

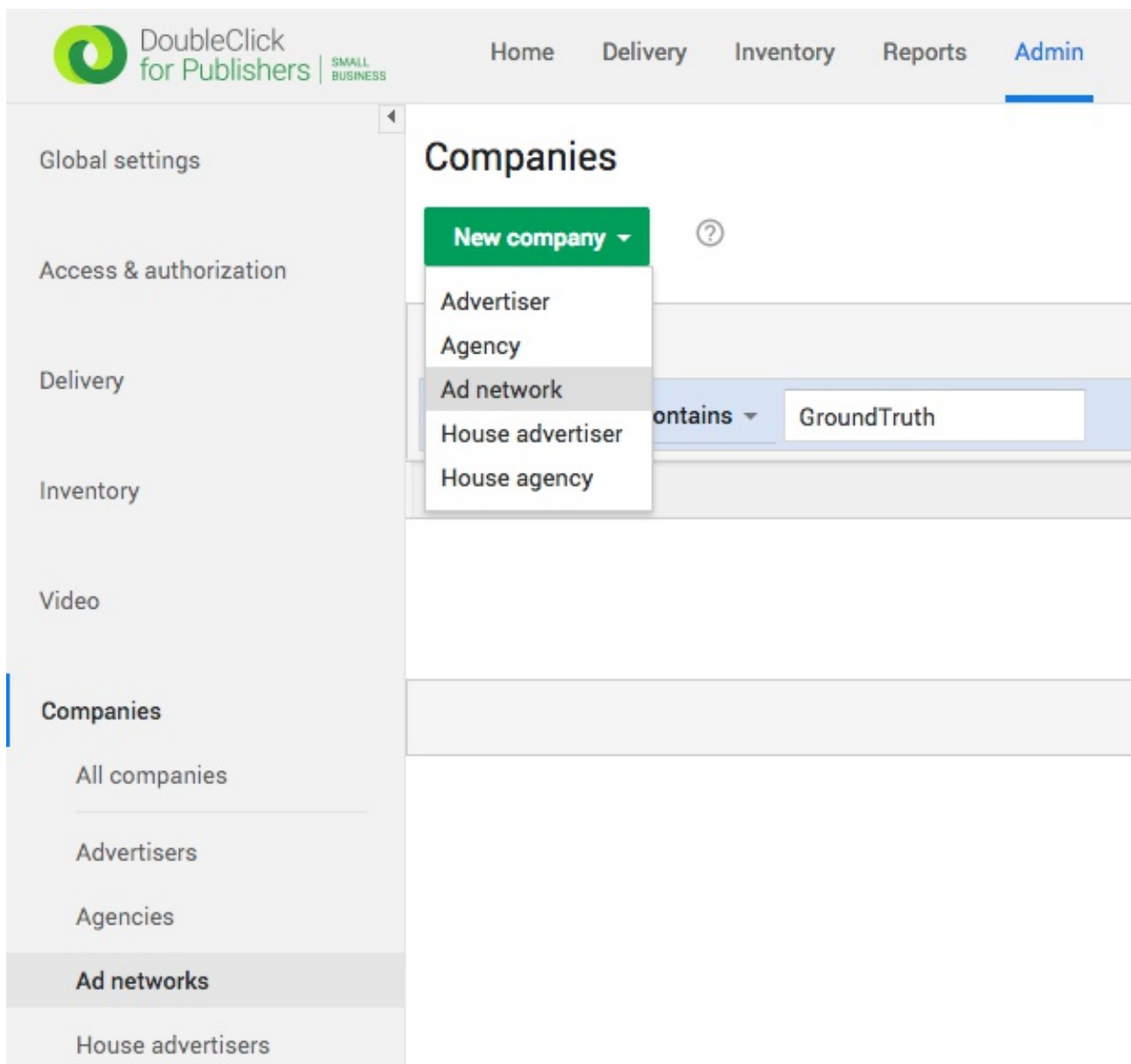
```
dependencies {  
    ...  
    compile 'com.groundtruth.sdk.displaysdk:customeventgooglemobileads:1.0.+'  
    ...  
}
```

Config in the DoubleClick for Publisher console

Step 2 – Add GroundTruth as an Ad Network

Under the Admin section click All Companies.

Click New Company and select Ad network.



Create a new Ad network called GroundTruth.

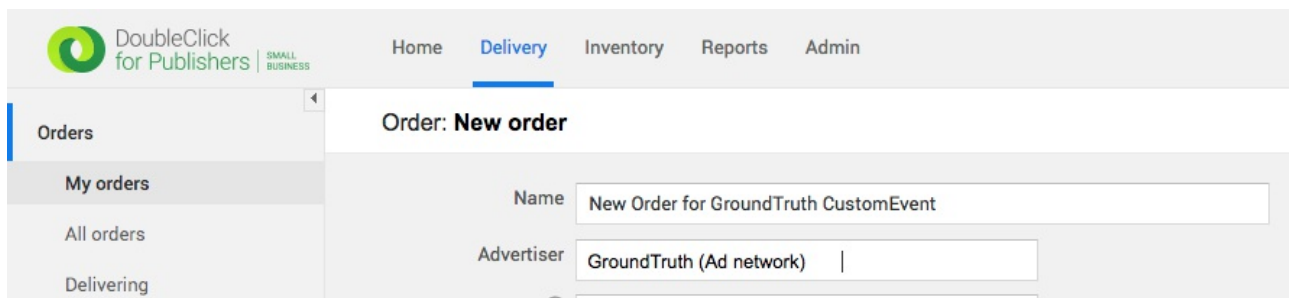
The screenshot shows the 'Admin' section of the DoubleClick for Publishers interface. The left sidebar contains a menu with 'Companies' selected. The main content area is titled 'Company: New ad network'. It includes a form with the following fields: 'Ad network' (a dropdown menu currently showing 'Other company'), 'Name' (a text input field containing 'GroundTruth'), and 'Enable for mediation' (a checked checkbox). Below these fields is a section for 'Additional settings'. At the bottom of the form are 'SAVE' and 'CANCEL' buttons. The footer of the interface shows the copyright notice '© 2017 Google'.

Step 3 – Create a New Order

Under the Delivery tab, select My Orders and create a New Order.

The screenshot shows the 'Delivery' section of the DoubleClick for Publishers interface. The left sidebar contains a menu with 'Orders' selected. The main content area is titled 'Orders'. It features a green 'NEW ORDER' button and a grey 'CHECK INVENTORY' button. Below these buttons, it indicates '5 orders'. At the bottom, there are buttons for 'PAUSE', 'RESUME', and a 'More actions' dropdown menu.

Set Advertiser as GroundTruth in the order.



The screenshot shows the DoubleClick for Publishers Small Business interface. The top navigation bar includes links for Home, Delivery (which is highlighted), Inventory, Reports, and Admin. On the left, a sidebar menu lists 'Orders' (highlighted), 'My orders', 'All orders', and 'Delivering'. The main content area is titled 'Order: New order' and contains two input fields: 'Name' with the value 'New Order for GroundTruth CustomEvent' and 'Advertiser' with the value 'GroundTruth (Ad network)'.

We recommend that you refer to Google's type and priorities and network documents to better understand how to configure your line items. GroundTruth custom events will be set up as a network line item and it will be in consideration along with AdSense and Ad Exchange.

Make sure you set the type as NETWORK and the end time is set to UNLIMITED to keep this maintain this configuration. Set the bid price in this section with the value that was agreed upon with the GroundTruth team.

Line item: **New line item**

Name

Custom Event Banner

Inventory sizes

?

☒ Standard

☐ Video VAST

?

320x50

×

Enter one or more sizes separated by a comma
[Target creatives and help forecast available inventory.](#)

Labels

?

optional

Add a label

?

☐ Allow same advertiser exception.

Comments

optional

Custom fields

?

optional

Type to find items

Settings

Type

?

Network

▼

12

Start time

Immediately

▼

End time

Unlimited

▼

Goal

100

% of remaining impressions

Rate

?

\$5.00

CPM

▼

USD

[set value CPM](#)

Total value

--

Select ad units with the one that you intend to use for the current campaign in the Add targeting section.

Add targeting

Targeting preset [?] optional

Inventory	Browse	Search	Selected criteria	Reset
	BACK Home > Ad units >		Inventory	
	Inventory filtered based on sizes (320x50). Show all		Ad units >	
	<input type="text" value="Type to filter items"/>		Custom Android	×
Key-values	Custom Android included			
Geography	Custom iOS include ▼			
Devices				
Connection				
Mobile application				
2 items		Include all	1 item	
			Save selected criteria as a preset [?]	
			<input type="text" value="Give this preset a name"/>	
			SAVE	

Configure the rest of the order to your specifications.

Step 4 – Create a New Creative

Under the line item, click New Creative.

DoubleClick
for Publishers

SMALL BUSINESS

[Home](#) [Delivery](#) [Inventory](#) [Reports](#) [Admin](#)

Orders

My orders

All orders

Delivering

Starting soon

Ending soon

Starred

xAd > My Order >

☆ Line items: **My Line Item** ▼

ID: 135284045 Status: Draft / Unreserved Type: Standard Time: Apr 12, 2017 10:43 PM PDT

ADD CREATIVES

More actions ▼

RUN REPORT

Impressions [?]

Clicks [?]

CTR [?]

Progress [?]

0

0

0.00%

N/A

Select SDK Mediation.

The screenshot shows the DoubleClick for Publishers interface. The top navigation bar includes links for Home, Delivery (selected), Inventory, Reports, and Admin. The left sidebar menu lists various sections: Orders, My orders, All orders, Delivering, Starting soon, Ending soon, Starred, Recently viewed, Ready, Draft, Inactive, Needs creatives, Pending approval, Disapproved, Completed, Paused, Archived, Video, Line items, and Creatives (selected). The main content area displays a list of creative templates with descriptions:

- Third party**: A highly customizable, interactive creative. You can use any custom HTML and JavaScript snippets as well as tags from a third-party ad server or DoubleClick for Advertisers.
- HTML5**: A creative built from HTML and supporting assets that enables rich user experiences.
- Image**: A basic image creative. Requires a GIF, JPG, or PNG file.
- Native (Not currently available due to size restrictions)**: A component-based native creative that is styled by the publisher.
- Custom**: Supply your own custom creative code.
- DoubleClick tag**: The recommended way to traffic a creative hosted by DoubleClick Campaign Manager or another DFP network. Traffic the creative with a URL rather than a creative snippet, streamlining creative trafficking and reducing reporting discrepancies.
- Image animation**: An animated HTML5 creative created from a set of images with timed transitions. Requires GIF, JPG or PNG files.
- Mobile video interstitial (Not currently available due to size restrictions)**: A full-screen interactive video ad that can include action buttons. Only for iOS 4+ and Android 2+.
- SDK mediation**: A creative that uses SDKs in your mobile application to send impressions to ad networks.
- User-defined template**: A creative based on a user-defined creative template.
- System-defined template**: A creative based on a standard DFP creative template.
- Existing creative**: Use a creative (of a compatible size) that already exists.

At the bottom of the creative list, there is a button labeled "Choose a size" with a dropdown arrow.

Select Custom Event under the Select Network drop-down.

Set Class Name to `com.xad.sdk.customevent.googlemobileads.CustomEventForDFP` for both banner interstitial ads.

Set `serverParameter` to your GroundTruth access key.

Set Location Data to Active.

DoubleClick for Publishers | SMALL BUSINESS

Home Delivery Inventory Reports Admin

Orders

My orders

All orders

Delivering

Starting soon

Ending soon

Starred

Recently viewed

Ready

Draft

Inactive

Needs creatives

Pending approval

Disapproved

Completed

Paused

Archived

Video

GroundTruth > New Order for GroundTruth CustomEvent > Custom Event Banner >

Creative: New creative

Type: SDK mediation creative Advertiser: GroundTruth

Name Custom Creative

Target ad unit size 320x50

Location data Active

Network timeout 5000 milliseconds

Ad networks 1

Select network Custom Event

Parameter your_access_key

Label

Class Name com.xad.sdk.customever

Add network Add a maximum of 5 ad networks

Orientation (mobile-only) optional Any

Custom fields Type to find items

SAVE CANCEL

Step 5 Integrating DFP

<https://developers.google.com/mobile-ads-sdk/docs/dfp/android/banner>

Note: If you are trying to implement Banner View Delegate methods to be notified when an event happens like `onAdLoaded..` or `onAdFailedToLoad..`, you should use `com.google.android.gms.ads.AdListener` instead of using `com.xad.sdk.listeners.BannerViewListener`.

Option 3 – AdMob

Step 1 – Import sdk AAR file

In the build.gradle of the whole project, please add following maven url:

```
allprojects {
    repositories {
        jcenter()
        maven {
            url "http://groundtruth.bintray.com/GroundTruth"
        }
    }
}
```

Grab the AAR from Maven Central by adding it as a dependency in your build.gradle file:

```
dependencies {
    ...
    compile 'com.groundtruth.sdk.displaysdk:customeventgooglemobileads:1.0.+'
    ...
}
```

Step 2 – Configure GroundTruth as an Ad Source

Click on Ad Sources under Mediation to view the ad source editor.

AD UNITS (1)

ALLOW & BLOCK ADS

SETTINGS

+ NEW AD UNIT

MOVE

ARCHIVE

View filtered report

<input type="checkbox"/>	↑ Ad unit	Ad format	Mediation	Frequency capping (per user) ?
<input type="checkbox"/>	ad_unit Ad unit ID: ca-app-pub-5709149378349090/2858123563	Banner	1 ad source	Not applicable

1-1 of 1

Click + New Add Network to open a list of ad networks.

+ NEW AD NETWORK EDIT DELETE			
<input type="checkbox"/>	Ad source	eCPM - USD (\$)	Country-specific settings
	AdMob Network (Optimized)	Real-time eCPM	Real-time eCPM

1-1 of 1 < >

Click + Custom Event to open the custom event editor.

Available ad networks	+ CUSTOM EVENT	Selected ad networks: 0
Aarki (SDK-less)	»	
AdColony	»	
AdFalcon	»	
AdFonic	»	
AdMob Network - added	»	
AdRally	»	

Set Class Name to `com.xad.sdk.customevent.googlemobileads.CustomEventForAdmob` for both banner interstitial ads.

Set `serverParameter` to your GroundTruth access key

Available ad networks
+ CUSTOM EVENT

Selected ad networks: 1
CLEAR ALL

Aarki (SDK-less) »
AdColony »
AdFalcon »
AdFonic »
AdMob Network - added »
AdRally »
ADResult »
AMoAd »
Amobee »

Custom Event

Class Name ⓘ

com.xad.sdk.customevent.googlemobiles

Label ⓘ

GroundTruth Network

Parameter ⓘ

Optional

your_access_key

Finally, click Continue and then Save to finalize the addition of GroundTruth as an ad source.

+ NEW AD NETWORK EDIT DELETE

<input type="checkbox"/>	Ad source	eCPM - USD (\$)	Country-specific settings
	AdMob Network (Optimized)	Real-time eCPM	Real-time eCPM
<input type="checkbox"/>	GroundTruth Network <small>Class Name: com.xad.sdk.customevent.googlemobiles.CustomEventForAdmob Parameter: your_access_key</small>	\$ 0.01	None

1-2 of 2 < >

Step 3 – Configure eCPM prices

Click on the AdMob Network and uncheck Optimize AdMob Network to allow GroundTruth's custom event to participate in the auction.

AdMob Network settings

☐ Optimize AdMob Network (recommended) ⓘ

Select **Optimize AdMob Network** to maximize total revenue. The real-time eCPM will be used to automatically position the AdMob Network in the mediation stack.

☐ Enable eCPM floor for the AdMob Network ⓘ \$

Define a minimum eCPM to run in auction.
New ad units may take up to a week to accurately apply floors. [Learn more](#)

CONTINUE

CANCEL

Enter the agreed eCPM price, per negotiation with GroundTruth, into the eCPM column.

<input type="checkbox"/>	Ad source	eCPM - USD (\$)	Country-specific settings
	AdMob Network (Not maximizing revenue. Optimize now)	\$ 4.00	None
<input type="checkbox"/>	xAd Network Class Name: XADCustomEventForGoogleMobileAd.XADCustomEventForAdmob Parameter: <access_key>	\$ 5.00	None

For this to work, the GroundTruth eCPM should ideally be higher than your default AdMob eCPM.

Step 4 – Integrating Admob

<https://developers.google.com/admob/android/banner>

Note: If you are trying to implement Banner View Delegate methods to be notified when an event happens like `onAdLoaded..` or `onAdFailedToLoad..`, you should use `com.google.android.gms.ads.AdListener` instead of using `com.xad.sdk.listeners.BannerViewListener`.

Option 4 – MoPub

Step 1 – Import sdk AAR file

In the `build.gradle` of the whole project, please add following maven url:

```
allprojects {
    repositories {
        jcenter()
        maven {
            url "http://groundtruth.bintray.com/GroundTruth"
        }
    }
}
```

Grab the AAR from Maven Central by adding it as a dependency in your `build.gradle` file:

```
dependencies {
    ...
    compile 'com.groundtruth.sdk.displaysdk:customeventmopub:1.0.+'
    ...
}
```

Step 2 Configure your app for multidex

Total methods in mopub base library has exceeded 64k, in order to solve this issue you need to enable multidex.

Set `multiDexEnabled = true` in the `defaultConfig` part and add multidex library in dependencies. Refer to <https://developer.android.com/studio/build/multidex.html#mdex-gradle> for more info about multidex.

```
defaultConfig {
    ...
    minSdkVersion 19
    targetSdkVersion 25
    versionCode 1
    versionName "1.0"
    multiDexEnabled = true
    ...
}
```

&

```
dependencies {
    ...
    compile 'com.android.support:multidex:1.0.1'
    ...
}
```

Step 3 – Add GroundTruth as a Native Network

mopub Apps Orders Marketplace **Networks** Segments Reports jacob.zelek@xad.com Help

Overview [?](#) [Add a Network](#)

Last 14 Days (3/30/2017 — 4/12/2017) ▾

REVENUE ECPM IMPRESSIONS NETWORK FILL RATE CTR

Networks

STATUS	NETWORKS	REVENUE	ECPM	ATTEMPTS	IMPRS.	FILL RATE	CLICKS	CTR
You have no networks								

[Export Data](#) ▾

Add a Network

To get started, you need to set up your networks and connect each one to MoPub. MoPub provides ad network mediation for dozens of major ad networks. To connect a network to MoPub, add the network below and follow the setup instructions to enable ad serving.

[AdColony >](#)[AdMob >](#)[Chartboost >](#)[Conversant Media >](#)[Facebook >](#)[HUNT Mobile Ads >](#)[LifeStreet Media >](#)[ONE by AOL:
Mobile \(Millennial\) >](#)[MobFox >](#)[mobileCore >](#)[Mojiva >](#)

Additional Networks

The following networks are supported by MoPub but are not fully certified. If you work with a network not shown here, you can set up these networks as a Custom Native Network or Custom Network. Note that these networks may require an adapter and network reporting data will not be available.

- [AdSense >](#)
- [Custom Network >](#)
- [Custom Native Network >](#)
- [TapIt! >](#)

Custom Native

Set up Custom Native Network

Special Instructions

- Custom Native Networks require the development of a Custom Event Class
- You must enter a Custom Class to enable ad serving for this type of network. Custom Event Class Data can be sent down in JSON format. Note: Custom Native Networks using a Custom Method implementation will not be supported in future MoPub SDKs.

Title

GroundTruth

Step 4 – Create a New Order

mopub

Apps**Orders**MarketplaceNetworksSegmentsReports

jacob.zelek@xad.comHelp

Orders

Create order

Find orders or advertisers

All except archived

Last 14 days

Order name	Advertiser	Enabled	Status	Line Items	Impressions	Clicks	CTR
No orders							
Try changing your filters to see more results							

Step 1 - Create a New Order

An Order is a set of Line Items that is usually created for a campaign and targets one or more Ad Units.

Order Name*:My Order

Advertiser*:GroundTruth

Description:What is this order going to accomplish? Who are you targeting?

Step 5 – Create a New Line Item



Step 2 - Add a Line Item

Each Order has one or more Line Items. A Line Item is a set of creatives and targeting parameters that runs with a specific budget.

Type & Priority*	Network	12	Name*: <input type="text" value="Line Item Name"/>
Line items set to Priority 12 will compete with MoPub Marketplace and Ad Networks.			
Network:	Custom Native Network		
Class:	<input "="" type="text" value="com.xad.sdk.customevent.mopub.{"/>		
Data (Optional):	<div><pre>{ "accessKey": "your_access_key", "test_mode": "false" }</pre></div>		
Method:	<input type="text" value="ex. myCustomEventMethod"/>		

Set Type & Priority to Network.

Set Class to `com.xad.sdk.customevent.mopub.CustomEventForMopub` for banner ads or `com.xad.sdk.customevent.mopub.CustomEventInterstitialForMopub` for interstitial ads.

Set Data to your GroundTruth access key.

Step 6 – Set eCPM

Set the duration, the agreed budget, and the negotiated eCPM.

Select the previously created Ad Unit which utilizes the GroundTruth custom ad network.

Start Time*:

01/26/2017

12:00 AM

Stop Time:

07/26/2017

11:59 PM

Please enter Start and Stop Times in Pacific Time.

Budget:

1000

impressions/day

eCPM

\$ 5.00

USD

Delivery Speed:

☐ Spread Evenly
 ☒ All at once

Day Part Targeting

Run this line item on the following days:

☒ All
 ☒ Sun
 ☒ Mon
 ☒ Tues
 ☒ Weds
 ☒ Thu
 ☒ Fri
 ☒ Sat

from: 01:00 PM to 04:00 PM

from: 01:00 PM to 04:00 PM

Step 3 - Ad Unit Targeting

Add a Filter

Select which apps and/or adunits you'd like this Line Item to target. This area can be completed or edited later.

APP NAME	AD UNITS	FORMAT
<div>Custom Event Demo</div> <div>(Android)</div>	<input type="checkbox"/> Banner Ad	320x50
<div>Custom Event Demo iOS</div> <div>(iOS)</div>	<input checked="" type="checkbox"/> Banner Ad	320x50
<div>MopubTester</div> <div>(iOS)</div>	<input type="checkbox"/> Interstitial <input type="checkbox"/> Banner Ad	Full 320x50

Continue setting all additional parameters and save.

Step 7 – Integrating Mopub

<https://www.mopub.com/resources/docs/android-sdk-integration/integrating-banner-ads-android/>

Note: If you are trying to implement Banner View Delegate methods to be notified when an event happens like `onBannerLoaded..` or `onBannerFailed..`, you should use `com.mopub.mobileads.MoPubView.BannerAdListener` instead of using `com.xad.sdk.listeners.BannerViewListener`.