

Adiscope Android Sdk 연동 가이드

버전 히스토리

| 날짜 | 변경 사항 |
|------------|---|
| 2018/03/04 | 기능 추가 <ul style="list-style-type: none">OptionSetter.setUseCloudFrontProxy(boolean useCloudFrontProxy) API 추가 |
| 2018/06/14 | 기능 추가 <ul style="list-style-type: none">오피셜 광고는 Adiscope 서버가 3rd-party 서버로부터 광고 캠페인들을 직접 수집하는 방식으로 변경되었습니다. 따라서 3rd-party 오피셜 sdk들은 제거되었습니다. 인터페이스 변경 <ul style="list-style-type: none">메인 클래스의 클래스 이름이 변경되었습니다. (com.nps.adiscope.Adiscope -> com.nps.adiscope.AdiscopeSdk) |
| 2018/06/25 | 기능 변경 <ul style="list-style-type: none">setUserId API에 사용되는 userId 파라미터의 길이 제한이 20자에서 32자로 변경되었습니다. |
| 2018/08/28 | 기능 추가 <ul style="list-style-type: none">Admob Reward Video Network 추가 |
| 2018/11/08 | 기능 추가 <ul style="list-style-type: none">FAN 5.0.1 (Facebook Audience Network) Reward Video Network 추가Admob 버전 17 지원 |
| 2019/01/04 | 기능 추가 <ul style="list-style-type: none">Applovin network sdk 7.0.3 -> 8.1.4 업데이트Unityads network sdk 2.1.2 -> 3.0.0 업데이트 (proguard 설정 변경) |
| 2019/3/4 | 기능 추가 <ul style="list-style-type: none">Vungle network sdk 4.0.3 -> 6.3.24 업데이트 (AndroidManifest, proguard 설정 및 라이브러리 구조 변경)Youappi Reward Video Network 추가 (v4.3.10)Offerwall 처리 개선 - 설치확인 기능의 예외 처리 추가 |
| 2019/08/09 | 연동가이드 설명 추가 <ul style="list-style-type: none">Mobvista sdk 설정 가이드 및 proguard 설정 변경 |
| 2019/10/24 | 기능 변경 <ul style="list-style-type: none">Applovin 네트워크 제거 및 Applovin 관련 연동가이드 제거Mobvista 네트워크 연동가이드 변경<ul style="list-style-type: none">AndroidManifest.xml의 MTGFileProvider provider 항목 제거mtg_provider_paths.xml xml 파일 생성하는 내용 제거 |
| 2019/11/01 | 기능 변경/연동가이드 변경 <ul style="list-style-type: none">FAN sdk 교체 : 5.0.1 -> 5.5.0FAN 연동가이드 대폭 변경 |
| 2020/01/08 | 기능 변경 <ul style="list-style-type: none">Vungle (v6.4.11) 연동가이드 대폭 변경 (Activity, proguard 설명 변경)Youappi Reward Video Network 제거 |
| 2020/01/31 | 기능 변경/연동가이드 변경 <ul style="list-style-type: none">Unityads sdk 교체 : 3.0.0 -> 3.4.2Unityads AndroidManifest.xml 항목 변경 |
| 2020/02/06 | 연동가이드 변경 <ul style="list-style-type: none">adcolony proguard 설정 변경 |

| | |
|------------|---|
| 2020/02/14 | 연동가이드 변경 <ul style="list-style-type: none"> Adiscope sdk 배포 방식을 maven repository로 변경 및 연동가이드 수정 |
| 2020/02/21 | 연동가이드 변경 <ul style="list-style-type: none"> 구글가속정책 가이드 추가 |
| 2020/03/02 | 기능 변경 <ul style="list-style-type: none"> setUserId API의 userId 문자열 32자 제한 조건이 제거됨 연동가이드 변경 <ul style="list-style-type: none"> setUserId API의 userId 문자열 32자 제한사항에 대한 설명 제거됨 |
| 2020/05/15 | 기능 변경 <ul style="list-style-type: none"> Vungle network sdk 업데이트 : 6.4.11 -> 6.5.3 (use androidx) Mobvista network sdk 업데이트 : 9.12.7 -> 13.1.1 (use androidx) Vungle, Mobvista 연동가이드 설명 변경 <ul style="list-style-type: none"> Vungle, Mobvista 새버전을 사용하기 위해서는 앱내에 androidx 라이브러리가 포함되어야합니다. 앱 환경에 따라서 androidx migration이 필요할 수 있습니다. (https://developer.android.com/jetpack/androidx/migrate) |
| 2020/06/02 | 기능 변경 <ul style="list-style-type: none"> OfferwallAd.show(String unitId) 인터페이스가 OfferwallAd.show(Activity activity, String unitId) 로 변경됨 Admob 네트워크의 Google Play Services Ads 버전이 17.2.0으로 변경됨 |
| 2020/06/23 | 기능 추가 <ul style="list-style-type: none"> 인터스티셜 기능 추가 및 연동가이드 설명 추가 admob 인터스티셜 네트워크 추가 |
| 2020/08/06 | 기능 개선 <ul style="list-style-type: none"> 오퍼월 기능 개선 |
| 2020/08/20 | 기능 수정 <ul style="list-style-type: none"> X509Certificate 경고 수정 |
| 2020/10/22 | 기능 변경 <ul style="list-style-type: none"> target sdk version 30 대응 admob sdk version 변경 : 17.2.0 -> 19.4.0 unityads sdk version 변경 : 3.4.2 -> 3.5.0 vungle sdk version 변경 : 6.5.3 -> 6.8.0 fan sdk version 변경 : 5.9.0 -> 6.1.0 |
| 2020/11/13 | 기능 변경 <ul style="list-style-type: none"> 멀티인스턴스 체계 적용 멀티인스턴스 체계 적용으로 인한 각 네트워크사의 adapter 구조 변경 Adiscope-android-sdk.adiscopeCore 1.5.9 -> 1.6.0 |
| 2020/11/24 | 기능 변경 <ul style="list-style-type: none"> Adiscope-android-sdk.adiscopeCore 1.6.0 -> 1.6.1 target sdk version 30 대응 adcolony sdk version 변경 : 4.1.4 -> 4.3.0 ironsource sdk version 변경 : 6.16.1 -> 7.0.3 기타 사용성 개선 |
| 2020/12/02 | 이슈 수정 <ul style="list-style-type: none"> 오퍼월 최초 진입시 ADID 가져 오지 못하는 이슈 수정 |

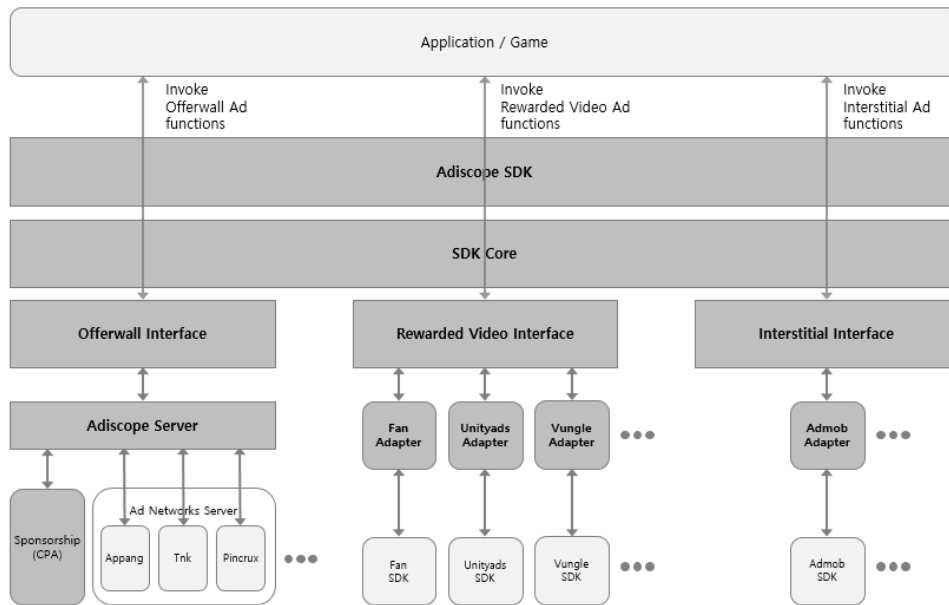
- [Adiscope Android Sdk 구조](#)
 - [Structure Diagram](#)
- [Adiscope Android Sdk 설치](#)
 - [1. gradle 연동 설정](#)
 - [project gradle](#)

- Module Gradle
 - 2. Network 별 추가작업
 - Admob – Rewarded Video / Interstitial Network
 - Fan – Rewarded Video Network
 - Mobvista – Rewarded Video Network
 - Vungle – Rewarded Video Network
- Overview
 - Adiscope
 - 초기화
 - 사용자 정보 설정
 - 광고 Instance
 - 광고의 표시
 - OfferwallAd
 - Singleton instance
 - Offerwall Ad 사용 방법
 - RewardedVideoAd
 - Singleton Instance
 - Rewarded Video Ad 사용 방법
 - InterstitialAd
 - Singleton Instance
 - Interstitial Ad 사용 방법
- API Reference
 - Adiscope
 - Class Declaration
 - Methods
 - OfferwallAd
 - Class Declaration
 - Methods
 - RewardedVideoAd
 - Class Declaration
 - Methods
 - InterstitialAd
 - Class Declaration
 - Methods
 - OptionSetter
 - Class Declaration
 - Methods
 - AdiscopeError
 - Class Declaration
 - ErrorCode
 - Example
- Server-to-server Reward 정보 연동
 - 메시지 Flow
 - 설정
 - Callback Request
 - Callback 유효성 검증
 - Response
 - 재시도 정책
 - Sample Code
- Appendix. 구글 가족 정책 (for Android)
 - 1) 구글 가족 정책이란?
 - 2) 파라미터 값 (매체→애디스콥)

Adiscope Android Sdk 구조

1. Structure Diagram

Adiscope Android Sdk 및 Plugin의 structure는 다음의 그림과 같다.



Adiscope Android Sdk 설치

1. gradle 연동 설정

project gradle

project build.gradle

```

allprojects {
    repositories {
        maven {
            url 'https://repository.adiscope.com/repository/adiscope/'
        }
        ...
    }
}

```

Module Gradle

module build.gradle

```

dependencies {

    // [required] adiscope library
    implementation 'com.nps.adiscope:adiscopeCore:1.5.9'
    implementation 'com.nps.adiscope:adiscopeAndroid:1.1.2'

    // adiscope sdk have to extract Google Advertising Id.
    // if com.google.android.gms.ads.identifier.AdvertisingIdClient class is not included in your app, uncomment following code
    // implementation 'com.google.android.gms:play-services-basement:8.3.0'

    // [optional] adiscope rewarded video networks
    implementation 'com.nps.adiscope:adapter.adcolony:4.3.0.0' // adcolony
    implementation 'com.nps.adiscope:adapter.admob:19.4.0.0' // admob (use play-services-ads:19.4.0 dependency)
    implementation 'com.nps.adiscope:adapter.fan:6.1.0.0' // fan
    implementation 'com.nps.adiscope:adapter.ironsource:7.0.3.0' // ironsource
    implementation 'com.nps.adiscope:adapter.mobvista:13.1.1.1' // mobvista (use androidx)
    implementation 'com.nps.adiscope:adapter.unityads:3.5.0.0' // unityads
    implementation 'com.nps.adiscope:adapter.vungle:6.8.0.0' // vungle (use androidx)
}

```

```
// [optional] adiscope interstitial networks
implementation 'com.nps.adiscope:adapter.admob:19.4.0.0' // admob (use play-services-ads:19.4.0 dependency)
}
```

- adiscopeCore, adiscopeAndroid 라이브러리는 필수로 포함시켜야합니다.
- play-services-basement은 옵션으로서 adiscope sdk 내에서 adid 추출에 사용됩니다. 앱에서 이미 google play-services-ads 를 사용중이라면 이것을 포함시킬 필요가 없습니다. 또는 adapter.admob 를 사용하는 경우에도 adapter.admob의 dependency에 google play-services-ads가 포함되었기때문에, play-services-basement를 포함시킬 필요가 없습니다.
- video network adapter들 중에서 앱에서 사용하고자하는 adapter만 implementation으로 추가하시면 됩니다. 모든 video network adapter를 implementation했더라도, adiscope admin 에서 off 설정된 network adapter들은 런타임에 구동되지 않습니다.

2. Network 별 추가작업

Admob - Rewarded Video / Interstitial Network

AndroidManifest.xml 의 application 태그 하위에 아래의 meta-data가 선언되어야합니다.

AndroidManifest.xml 설정

```
<application ..>
// ... (생략)
<meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version"/>

</application>
```

Fan - Rewarded Video Network

디바이스에서 광고를 테스트하기위해서는 Facebook admin에 디바이스의 ADID가 등록되어야합니다.

Mobvista - Rewarded Video Network

13.1.1.0 버전부터 androidx 라이브러리를 사용합니다. 앱 환경에 따라서 androidx migration이 필요할 수 있습니다. (<https://developer.android.com/jetpack/androidx/migrate>)

Vungle - Rewarded Video Network

6.5.3.0 버전부터 androidx 라이브러리를 사용합니다. 앱 환경에 따라서 androidx migration이 필요할 수 있습니다. (<https://developer.android.com/jetpack/androidx/migrate>)

Overview

Adiscope

Adiscope class는 Adiscope Sdk의 초기화 및 설정을 수행한다.

1. 초기화

mediald, mediaSecret이 필요하며 이 값은 admin page를 통해 app을 등록하여 발급받아야 한다.

Sdk Initilization

```
import com.nps.adiscope.AdiscopeSdk;

// initialize AdiscopeSdk, must be called in main thread
AdiscopeSdk.initialize(this, "exampleMediald", "exampleMediaSecret");
```

2. 사용자 정보 설정

application 사용자의 unique user id를 Adiscope Sdk에 전달한다. unique user id는 보상이 지급 될 시 사용자를 구분하기 위해 사용된다.

Setting User Id

```
import com.nps.adiscope.AdiscopeSdk;

// set unique user id to identify the user in reward information
AdiscopeSdk.setUserId("exampleUniqueUserId");
```

3. 광고 Instance

각 Ad의 global singleton instance를 얻을 수 있다.

Get Instance


```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.offerwall.OfferwallAd;
import com.nps.adiscope.reward.RewardedVideoAd;

// get singleton instance of offerwall ad
OfferwallAd offerwallAd = AdiscopeSdk.getOfferwallAdInstance(this);
RewardedVideoAd rewardedVideoAd = AdiscopeSdk.getRewardedVideoAdInstance(this);
```

4. 광고의 표시

실제 광고를 보여주기 위해서는 필요한 상세정보에 대해서는 Overview section의 OfferwallAd와 RewardedVideoAd 항목을 참조할 것

Offerwall Ad는 이후 Show를 하면 광고가 표시된다.

 Rewarded Video Ad의 경우는 이후 위의 과정을 거친후 RewardedVideoAd.Load를 실행해야 OnLoaded callback이 전달 된 후에 Show를 할 수 있다

OfferwallAd

1. Singleton instance

- OfferwallAd Instance는 global singleton instance이므로 여러개의 instance가 생성되지 않는다.

2. Offerwall Ad 사용 방법

Offerwall Ad를 게임 내에서 표시하기 위해서는 다음의 과정을 거쳐야 한다.

- 초기화 및 사용자 정보 설정

Sdk Initilization

```
import com.nps.adiscope.AdiscopeSdk;

AdiscopeSdk.initialize(this, "exampleMediald", "exampleMediaSecret");
AdiscopeSdk.setUserId("exampleUniqueUserId");
```

- Offerwall Ad instance 생성

Get Instance

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.offerwall.OfferwallAd;

// get singleton instance of Offerwall Ad
OfferwallAd offerwallAd = AdiscopeSdk.getOfferwallAdInstance(this);
```

- Callback 등록

Register Callback Event Handlers

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.offerwall.OfferwallAd;
import com.nps.adiscope.offerwall.OfferwallAdListener;

// set callbacks
offerwallAd.setOfferwallAdListener(this);
```

d. Show

- i. Offerwall 광고를 사용자에게 보여준다.
- ii. Show method는 중복하여 호출 할 수 없다.
- iii. unitId는 admin page을 통하여 등록/조회한다.

Show Ad

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.offerwall.OfferwallAd;

// show offerwall ad
if (offerwallAd.show(activity, "UNIT_ID"))
{
    // Succeed
}
else
{
    // show is already in progress
}
```

show가 실행되면 (return값이 True일 경우) onOfferwallAdOpened 와 onOfferwallAdFailedToShow 중 하나가 항상 호출되고, onOfferwallAdOpened가 호출되었다면 이후 onOfferwallAdClosed가 항상 호출된다.

e. Callbacks

- i. show 성공 시 onOfferwallAdOpened, onOfferwallAdOpened callback이 순차적으로 호출된다.
- ii. show 실패 시 onOfferwallAdFailedToShow callback이 호출된다.

RewardedVideoAd

1. Singleton Instance

- a. Rewarded Video Ad Instance는 global singleton instance이므로 여러개의 instance가 생성되지 않는다.

2. Rewarded Video Ad 사용 방법

Rewarded Video Ad를 게임 내에서 표시하기 위해서는 다음의 과정을 거쳐야 한다.

- a. 초기화 및 사용자 정보 설정

Sdk Initialization

```
import com.nps.adiscope.AdiscopeSdk;

AdiscopeSdk.initialize(this, "exampleMediaId", "exampleMediaSecret");
AdiscopeSdk.setUserId("exampleUniqueUserId");
```

- b. Rewarded Video Ad instance 생성

Get Instance

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.reward.RewardedVideoAd;

// get singleton instance of Rewarded Video Ad
RewardedVideoAd rewardedVideoAd = AdiscopeSdk.getRewardedVideoAdInstance(this);
```

- c. Callback 등록

Register Callback Event Handlers

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.offerwall.RewardedVideoAd;
import com.nps.adiscope.offerwall.RewardedVideoAdListener;
```

```
// set callbacks
rewardedVideoAd.setRewardedVideoAdListener(this);
```


- d. loadAll
- i. 앱에서 사용되는 모든 Ad 네트워크들의 광고를 load 한다.

Load Ad

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.offerwall.RewardedVideoAd;

// load all rewarded video ad
rewardedVideoAd.loadAll();
```


앱 실행 후에 초기화 과정에서 한 번만 호출하길 권장한다

 show를 호출하기 위해서는 load(unitId)를 다시 호출해야 한다.

- e. load
- i. 특정 유닛에 속한 ad 네트워크들의 광고를 load 한다.
 - ii. onRewardedVideoAdLoaded callback이 호출되면 load가 완료된 것이다.

Load Ad

```
// load a rewarded video ad which belongs to a specific unit
rewardedVideoAd.load("UNIT_ID");
```

load가 실행되면 onRewardedVideoAdLoaded와 onRewardedVideoAdFailedToLoad 중 하나의 callback은 항상 호출된다. Rewarded Video Ad의 load와 show는 pair로 호출되어야 한다. 즉 load를 한 후 show를 하고, 광고를 show한 후에는 다시 load를 하여  다음번 show를 준비하여야 한다.

- f. isLoading
- i. 광고가 load 되었는지 상태를 확인 할 수 있다.

Check Loaded Ad

```
// check video load
if (rewardedVideoAd.isLoading("UNIT_ID"))
{
    // show ad here
}
else
{
    // do something else
}
```

- g. show
- i. 마지막으로 load된 광고를 사용자에게 보여준다.
 - ii. show 호출 후에는 다시 load를 호출해야한다.
 - iii. show method는 중복하여 호출 할 수 없다.
 - iv. unitId는 admin page를 통하여 등록/조회한다.

Show Ad

```
// check rewarded video is loaded
if (rewardedVideoAd.isLoading("UNIT_ID"))
{
    // only one "show" can not be requested at a time
    // if show() returns false, show is in progress somewhere else
    if (rewardedVideoAd.show())
    {
        Debug.Log("AdiscopeExample - RewardedVideoAd.show");
    }
    else
    {
        // show is already in progress
        Debug.Log("AdiscopeExample - this show request is duplicated");
    }
}
else
{
}
```



```
// ad is not loaded
Debug.Log("AdiscopeExample - RewardedVideoAd is not loaded");
}
```

show가 실행되면 (return값이 True일 경우) onRewardedVideoAdOpened 와 onRewardedVideoAdFailedToShow 중 하나가 항상 호출된다. onRewardedVideoAdOpened가 호출되었다면 이후 onRewardedVideoAdClosed 가 항상 호출된다. Rewarded Video Ad의 load와 show는 pair로 호출되어야 한다. 즉 load를 한 후 show를 하고, 광고를 show한 후에는 다시 load를 하여 다음번 show를 준비하여야 한다.

h. Reward

- 보상이 주어져야 할 경우 onRewarded callback이 호출되며 그 parameter로 관련 정보가 전달된다.
- 이 보상 정보를 바탕으로 게임 내에서 보상을 지급할 수 있다.

Reward

```
@Override
public void onRewarded(String unitId, RewardItem rewardItem) {
    /*
     * RewardItem.getType - 보상 type
     * RewardItem.getAmount - 보상의 양
     */
    DoSomethingWithReward();
}
```

onRewarded는 보통 onRewardedVideoAdOpened 와 onRewardedVideoAdClosed 사이에 호출되는 경우가 많으나 광고 System의 상황에 따라 달라 질 수 있다. 또한 onRewarded가 호출되지 않는 경우도 존재할 수 있다.

Mediation network 중 Adcolony의 경우, Reward 설정을 Server-to-server로 하였다면, Adcolony의 Video 시청 후에는 onRewarded가 호출되지 않는다.

Reward 정보는 abusing 방식을 위해서 Server-to-server 방식으로 전달 받는 것을 권장한다.

Server-to-server 방식을 선택하더라도 보상이 전달 될 시에는 onRewarded가 호출된다. 이때는 Server를 통해 전달받은 정보를 기준으로 처리하고, onRewarded를 통해 전달받은 정보는 검증용으로 사용하거나 무시하도록 한다.

i. Callbacks

- load 성공 시 onRewardedVideoAdLoaded, 실패 시 onRewardedVideoAdFailedToLoad 가 호출된다.
- show 성공 시 onRewardedVideoAdOpened, onRewardedVideoAdClosed 가 순차적으로 호출되고, 실패 시 onRewardedVideoAdFailedToShow 가 호출된다.
- 보상이 있을 경우 onRewarded 가 호출된다.

InterstitialAd

1. Singleton Instance

- Interstitial Ad Instance는 global singleton instance이므로 여러개의 instance가 생성되지 않는다.

2. Interstitial Ad 사용 방법

Interstitial Ad를 게임 내에서 표시하기 위해서는 다음의 과정을 거쳐야 한다.

a. 초기화

Sdk Initialization

```
import com.nps.adiscope.AdiscopeSdk;

AdiscopeSdk.initialize(this, "exampleMediaId", "exampleMediaSecret");
```

b. Interstitial Ad instance 생성

Get Instance

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.interstitial.InterstitialAd;

// get singleton instance of Interstitial Ad
InterstitialAd interstitialAd = AdiscopeSdk.getInterstitialAdInstance(this);
```

c. Callback 등록

Register Callback Event Handlers

```
import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.interstitial.InterstitialAd;
import com.nps.adiscope.interstitial.InterstitialAdListener;
```

```
// set callbacks
interstitialAd.setInterstitialAdListener(this);
```

d. load

- 특정 유닛에 속한 ad 네크워크들의 광고를 load 한다.
- onInterstitialAdLoaded callback이 호출되면 load가 완료된 것이다.

Load Ad

```
// load a interstitial ad which belongs to a specific unit
interstitialAd.load("UNIT_ID");
```

load가 실행되면 onInterstitialAdLoaded와 onInterstitialAdFailedToLoad 중 하나의 callback은 항상 호출된다. Interstitial Ad의 load와 show는 pair로 호출되어야 한다. 즉 load를 한 후 show를 하고, 광고를 show한 후에는 다시 load를 하여 다음 번 show를 준비하여야 한다.

e. isLoaded

- 광고가 load 되었는지 상태를 확인 할 수 있다.

Check Loaded Ad

```
// check Interstitial load
if (interstitialAd.isLoaded("UNIT_ID"))
{
    // show ad here
}
else
{
    // do something else
}
```

f. show

- 마지막으로 load된 광고를 사용자에게 보여준다.
- show 호출 후에는 다시 load를 호출해야한다.
- show method는 중복하여 호출 할 수 없다.
- unitId는 admin page을 통하여 등록/조회한다.

Show Ad

```
// check Interstitial Ad is loaded
if (interstitialAd.isLoaded("UNIT_ID"))
{
    // only one "show" can not be requested at a time
    // if show() returns false, show is in progress somewhere else
    if (interstitialAd.show())
    {
        Debug.Log("AdiscopeExample - InterstitialAd.show");
    }
    else
    {
        // show is already in progress
        Debug.Log("AdiscopeExample - this show request is duplicated");
    }
}
else
{
    // ad is not loaded
    Debug.Log("AdiscopeExample - InterstitialAd is not loaded");
}
```

show가 실행되면 (return값이 True일 경우) onInterstitialAdOpened 와 onInterstitialAdFailedToShow 중 하나가 항상 호출되고, onInterstitialAdOpened가 호출되었다면 이후 onInterstitialAdClosed 가 항상 호출된다. Interstitial Ad의 load와 show는 pair로 호출되어야 한다. 즉 load를 한 후 show를 하고, 광고를 show한 후에는 다시 load를 하여 다음 번 show를 준비하여야 한다.

g. Callbacks

- load 성공 시 onInterstitialAdLoaded, 실패 시 onInterstitialAdFailedToLoad 가 호출된다.
- show 성공 시 onInterstitialAdOpened, onInterstitialAdClosed 가 순차적으로 호출되고, 실패 시 onInterstitialAdFailedToShow 가 호출된다.

API Reference

Adiscope

1. Class Declaration

Declaration

```
package com.nps.adiscope;

public class AdiscopeSdk

public static void initialize(Activity activity, String mediald, String mediaSecret)
public static void initialize(final Activity activity, String mediald, String mediaSecret, String callbackTag)
public static boolean setUserId(String userId)
public static OfferwallAd getOfferwallAdInstance(Activity activity)
public static RewardedVideoAd getRewardedVideoAdInstance(Activity activity)
public static void getUnitStatus(String unitId, IUnitStatus callback)
```

2. Methods

a. initialize

Sdk 기능을 초기화 한다.

Definition

```
public static void Initialize(Activity activity, String mediald, String mediaSecret)
public static void Initialize(Activity activity, String mediald, String mediaSecret, String callbackTag)
```

Parameters

| | |
|-------------|--|
| activity | Activity |
| mediald | Amdin page에서 등록한 해당 application에 대한 Id |
| mediaSecret | mediald에 대응되는 secret key |
| callbackTag | 보상 콜백을 복수 개로 등록해서 사용할시에 어떤 보상 콜백을 사용할지 지정한다. 지정하지 않을시에는 기본 보상콜백이 사용된다. |

Return

b. setUserId

Application 사용자의 Unique Id를 설정한다. 이 정보는 reward 지급 등에 있어 사용자를 구분하는데 사용된다.

Definition

```
public static boolean setUserId(String userId)
```

Parameters

| | |
|--------|------------------------------|
| userId | application user's unique id |
|--------|------------------------------|

Return

c. GetOfferwallAdInstance

Offerwall Ad의 전역 Singleton 객체를 생성한다.
Initialize()가 먼저 호출 되어야 함

Definition

```
public static OfferwallAd getOfferwallAdInstance(Activity activity)
```

Parameters

| | |
|----------|----------|
| activity | Activity |
|----------|----------|

| Return | |
|-------------|----------------------------------|
| OfferwallAd | Offerwall 광고의 Singleton Instance |

- d. **GetRewardedVideoAdInstance**
 Rewarded Video Ad의 전역 Singleton 객체를 생성한다.
 Initialize()가 먼저 호출 되어야 함

| Definition |
|---|
| public static RewardedVideoAd getRewardedVideoAdInstance(Activity activity) |

| Parameters | |
|-----------------|---------------------------------------|
| activity | Activity |
| Return | |
| RewardedVideoAd | Rewarded Video 광고의 Singleton Instance |

- e. **getUnitStatus**
 유닛의 상태 정보를 구한다.

| Definition |
|---|
| public static void getUnitStatus(String unitId, IUnitStatus callback) |

| Parameters | | | |
|-------------|--|------|--------|
| unitId | 유닛 id | | |
| callback | 결과를 리턴받을 콜백 객체 | | |
| Return | | | |
| IUnitStatus | Interface | | |
| | com.nps.adiscopemodel.IUnitStatus { void onResult(AdiscopeError error, UnitStatus unitStatus); } | | |
| | Properties | | |
| | UnitStatus.isLive | bool | 유닛 수익화 |
| | UnitStatus.isActive | bool | 유닛 활성화 |

| Example |
|---|
| <pre>AdiscopeSdk.getUnitStatus("UNIT_ID", new IUnitStatus() { @Override public void setResult(AdiscopeError error, UnitStatus unitStatus) { if (error == null) { // use unitStatus.isLive() // use unitStatus.isActive() } else { // error occurred } } });</pre> |

OfferwallAd

1. Class Declaration

| Declaration |
|--|
| <pre>package com.nps.adiscope.offerwall; public interface OfferwallAd</pre> |

```
boolean show(Activity activity, String unitId)
void setOfferwallAdListener(OfferwallAdListener offerwallAdListener)

public interface OfferwallAdListener

void onOfferwallAdOpened(String unitId)
void onOfferwallAdFailedToShow(String unitId, AdiscopeError error)
void onOfferwallAdClosed(String unitId)
```

2. Methods

a. show

Offerwall 광고를 사용자에게 표시한다.

Definition

```
boolean show(Activity activity, String unitId)
```

Parameters

| | |
|----------|---|
| activity | 상위 액티비티 |
| unitId | 사용자에게 표시할 광고의 unit id. Admin page에 등록된 id와 동일해야 한다. |

Return

| | |
|------|---|
| bool | show가 정상적으로 시작되면 True, 만약 이미 다른 show가 처리되는 중이라면 False |
|------|---|

show가 실행되면 (return값이 True일 경우) onOfferwallAdOpened 와 onOfferwallAdFailedToShow 중 하나가 항상 호출되고, onOfferwallAdOpened가 호출되었다면 이후 onOfferwallAdClosed가 항상 호출된다.

Callback

| onOfferwallAdOpened | Offerwall 광고창이 열릴 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr></table> | Parameters | | unitId | 유닛 Id | | |
|---------------------------|----------------------------|---|------------|--|--------|-------|-------|------|
| Parameters | | | | | | | | |
| unitId | 유닛 Id | | | | | | | |
| onOfferwallAdClosed | Offerwall 광고창이 닫혔을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr></table> | Parameters | | unitId | 유닛 Id | | |
| Parameters | | | | | | | | |
| unitId | 유닛 Id | | | | | | | |
| onOfferwallAdFailedToShow | Offerwall 광고창을 보여 줄 수 없을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr><tr><td>error</td><td>에러코드</td></tr></table> | Parameters | | unitId | 유닛 Id | error | 에러코드 |
| Parameters | | | | | | | | |
| unitId | 유닛 Id | | | | | | | |
| error | 에러코드 | | | | | | | |

Example

```
package com.test.myapplication;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.AdiscopeError;
import com.nps.adiscope.offerwall.OfferwallAd;
import com.nps.adiscope.offerwall.OfferwallAdListener;

public class SampleActivity extends Activity implements OfferwallAdListener {

    OfferwallAd offerwallAd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AdiscopeSdk.initialize(this, "exampleMediaId", "exampleMediaSecret");
        AdiscopeSdk.setUserid("exampleUniqueUserId");

        offerwallAd = AdiscopeSdk.getOfferwallAdInstance(this);
        offerwallAd.setOfferwallAdListener(this);
    }
}
```

```

@Override
public void onOfferwallAdOpened(String unitId) {
    Log.d("adiscope", "onOfferwallAdOpened " + unitId);
}

@Override
public void onOfferwallAdFailedToShow(String unitId, AdiscopeError error) {
    Log.d("adiscope", "onOfferwallAdFailedToShow " + unitId + ", error " + error);
}

@Override
public void onOfferwallAdClosed(String unitId) {
    Log.d("adiscope", "onOfferwallAdClosed " + unitId);
}

public void showOfferwallAd() {
    offerwallAd.show(this, "UNIT_ID");
}
}

```

RewardedVideoAd

1. Class Declaration

Declaration

```

package com.nps.adiscope.reward;

public interface RewardedVideoAd

void loadAll()
void load(String unitId)
boolean isLoaded(String unitId)
boolean show()
void setRewardedVideoAdListener(RewardedVideoAdListener rewardedVideoAdListener)

public interface RewardedVideoAdListener

void onRewardedVideoAdLoaded()
void onRewardedVideoAdFailedToLoad(AdiscopeError error)
void onRewardedVideoAdOpened(String unitId)
void onRewardedVideoAdClosed(String unitId)
void onRewarded(String unitId, RewardItem rewardItem)
void onRewardedVideoAdFailedToShow(String unitId, AdiscopeError error)

```

2. Methods

a. loadAll

앱에서 사용되는 모든 Rewarded Video 광고를 load 한다.
 show()를 사용하기 위해서는 loadAll() API가 아닌 load(unitId) API의 호출이 선행되어야 한다.
 앱 실행시에 앱 초기화과정에서 loadAll API를 호출하여, 앱에서 사용되는 모든 Rewarded Video 광고가 load 되도록하기를 권장한다.

Definition

```
void loadAll()
```

Parameters

| | |
|--|--|
| | |
|--|--|

Return

| | |
|--|--|
| | |
|--|--|

loadAll이 실행되면 onRewardedVideoAdLoaded 와 onRewardedVideoAdFailedToLoad 중 하나의 callback은 항상 호출된다.

Callback

onRewardedVideoAdLoaded

Rewarded Video 를 load하였을 때

Parameters

| | | |
|--|--|--|
| | | |
|--|--|--|

| | | | |
|-------------------------------|------------------------------|------------|------|
| | | | |
| onRewardedVideoAdFailedToLoad | Rewarded Video load가 실패하였을 때 | Parameters | |
| | | error | 에러코드 |

b. load

특정 유닛에 속한 ad 네트워크들의 Rewarded Video 광고를 load 한다.

Rewarded Video Ad의 load와 show는 pair로 호출되어야 한다. 즉 load를 한 후 show를 하고, 광고를 show한 후에는 다시 load를 하여 다음번 show를 준비하여야 한다.

load - show 후 다시 load를 하려 할 때 load 는 show 이후 언제든 호출가능하다. 광고가 show되는 동안 다음 광고를 load를 할 수도 있지만 이는 사용하는 mediation ad network company의 종류에 따라 달라질 수 있으므로 항상 보장되는 동작은 아니다. 그러므로 show의 callback 인 onRewardedVideoAdClosed 에서 다시 load를 하는 것을 권장한다.

이는 Abusing 방식을 위해 Rewarded Video Ad를 연속으로 보여주는 것을 제한하여 한번 광고를 보고 나면 일정 시간이 지난 후에 다시 Show를 할 수 있도록 Admin page에서 서비스 설정을 할 수 있기 때문이다. (일정 시간 이내에 show를 할 경우 onRewardedVideoAdFailedToShow callback이 호출된다)

| Definition |
|--------------------------|
| void load(String unitId) |

| Parameters |
|---|
| unitId load 할 광고의 unit id. Admin page에 등록된 id와 동일해야 한다. |
| Return |
| |

load가 실행되면 onRewardedVideoAdLoaded 와 onRewardedVideoAdFailedToLoad 중 하나의 callback은 항상 호출된다.

| Callback | | | | | | |
|-------------------------------|------------------------------|---|------------|--|-------|------|
| onRewardedVideoAdLoaded | Rewarded Video 를 load하였을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td></td><td></td></tr></table> | Parameters | | | |
| Parameters | | | | | | |
| | | | | | | |
| onRewardedVideoAdFailedToLoad | Rewarded Video load가 실패하였을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>error</td><td>에러코드</td></tr></table> | Parameters | | error | 에러코드 |
| Parameters | | | | | | |
| error | 에러코드 | | | | | |

c. isLoaded

특정 유닛의 Rewarded Video 광고가 load 되었는 지를 확인한다.

| Definition |
|---------------------------------|
| boolean isLoaded(String unitId) |

| Parameters |
|--|
| unitId load 여부를 체크할 광고의 unit id |
| Return |
| bool load된 광고가 있을 시 True, load된 광고가 없을 시 False |

d. Show

최근에 load된 Rewarded Video 광고 유닛에 속한 광고를 사용자에게 보여준다.

| Definition |
|----------------|
| boolean show() |

| Parameters |
|---|
| |
| Return |
| bool show가 정상적으로 시작되면 True, 만약 이미 다른 show가 진행중이라면 False |

show가 실행되면 (return값이 True일 경우) onRewardedVideoAdOpened 와 onRewardedVideoAdFailedToShow 중 하나가 항상 호출된다. onRewardedVideoAdOpened가 호출되었다면 이후 onRewardedVideoAdClosed가 항상 호출된다.



OnRewarded는 보통 onRewardedVideoAdOpened 와 onRewardedVideoAdClosed 사이에 호출되는 경우가 많으나 광고 System의 상황에 따라 달라 질 수 있다.

Reward 정보는 abusing 방지를 위해서 Server-to-server 방식으로 전달 받는 것을 권장한다.

Server-to-server 방식을 선택하더라도 보상이 전달 될 시에는 OnRewarded가 호출된다. 이때는 Server를 통해 전달받은 정보를 기준으로 처리하고, OnRewarded를 통해 전달받은 정보는 검증용으로 사용하거나 무시하도록 한다.

| Call back | | | | | | | | | | | | | | | |
|-------------------------------|---|---|------------|-------|-----------|--------|-------|------|------------|---|--|---------|-------|-----------|-------|
| onRewardedVideoAdOpened | Rewarded Video 광고창이 열릴 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr></table> | Parameters | | unitId | 유닛 Id | | | | | | | | | |
| Parameters | | | | | | | | | | | | | | | |
| unitId | 유닛 Id | | | | | | | | | | | | | | |
| onRewardedVideoAdClosed | Rewarded Video 광고창이 닫혔을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr></table> | Parameters | | unitId | 유닛 Id | | | | | | | | | |
| Parameters | | | | | | | | | | | | | | | |
| unitId | 유닛 Id | | | | | | | | | | | | | | |
| onRewarded | Rewarded Video 시청 후 보상이 있을 시 | <table><tr><th colspan="3">Parameters</th></tr><tr><td>unitId</td><td colspan="2">유닛 Id</td></tr><tr><td>RewardItem</td><td colspan="2"><table><tr><td>getType</td><td>보상 타입</td></tr><tr><td>getAmount</td><td>보상 수량</td></tr></table></td></tr></table> | Parameters | | | unitId | 유닛 Id | | RewardItem | <table><tr><td>getType</td><td>보상 타입</td></tr><tr><td>getAmount</td><td>보상 수량</td></tr></table> | | getType | 보상 타입 | getAmount | 보상 수량 |
| Parameters | | | | | | | | | | | | | | | |
| unitId | 유닛 Id | | | | | | | | | | | | | | |
| RewardItem | <table><tr><td>getType</td><td>보상 타입</td></tr><tr><td>getAmount</td><td>보상 수량</td></tr></table> | | getType | 보상 타입 | getAmount | 보상 수량 | | | | | | | | | |
| getType | 보상 타입 | | | | | | | | | | | | | | |
| getAmount | 보상 수량 | | | | | | | | | | | | | | |
| onRewardedVideoAdFailedToShow | Rewarded Video 광고창을 보여 줄 수 없을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr><tr><td>error</td><td>에러코드</td></tr></table> | Parameters | | unitId | 유닛 Id | error | 에러코드 | | | | | | | |
| Parameters | | | | | | | | | | | | | | | |
| unitId | 유닛 Id | | | | | | | | | | | | | | |
| error | 에러코드 | | | | | | | | | | | | | | |

Example

```
package com.test.myapplication;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.AdiscopeError;
import com.nps.adiscope.reward.RewardItem;
import com.nps.adiscope.reward.RewardedVideoAd;
import com.nps.adiscope.reward.RewardedVideoAdListener;

public class SampleActivity extends Activity implements RewardedVideoAdListener {

    RewardedVideoAd rewardedVideoAd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AdiscopeSdk.initialize(this, "exampleMediaId", "exampleMediaSecret");
        AdiscopeSdk.setUserId("exampleUniqueUserId");

        rewardedVideoAd = AdiscopeSdk.getRewardedVideoAdInstance(this);
        rewardedVideoAd.setRewardedVideoAdListener(this);
    }

    @Override
    public void onRewardedVideoAdLoaded() {
        Log.d("adiscope", "onRewardedVideoAdLoaded ");
    }

    @Override
    public void onRewardedVideoAdFailedToLoad(AdiscopeError error) {
        Log.d("adiscope", "onRewardedVideoAdFailedToLoad " + error);
    }

    @Override
```



```

public void onRewardedVideoAdOpened(String unitId) {
    Log.d("adiscopescope", "onRewardedVideoAdOpened " + unitId);
}

@Override
public void onRewardedVideoAdClosed(String unitId) {
    Log.d("adiscopescope", "onRewardedVideoAdClosed " + unitId);
}

@Override
public void onRewarded(String unitId, RewardItem rewardItem) {
    Log.d("adiscopescope", "onRewarded " + unitId);
}

@Override
public void onRewardedVideoAdFailedToShow(String unitId, AdiscopescopeError error) {
    Log.d("adiscopescope", "onRewardedVideoAdFailedToShow " + unitId + ", error " + error);
}

public void loadAllRewardedVideoAd() {
    rewardedVideoAd.loadAll();
}

public void loadRewardedVideoAd() {
    rewardedVideoAd.load("UNIT_ID");
}

public void showRewardedVideoAd() {
    if (rewardedVideoAd.isLoaded("UNIT_ID")) {
        rewardedVideoAd.show();
    }
}
}

```

InterstitialAd

1. Class Declaration

| Declaration |
|---|
| <pre> package com.nps.adiscopescope.interstitial; public interface InterstitialAd void load(String unitId) boolean isLoaded(String unitId) boolean show() void setInterstitialAdListener(InterstitialAdListener interstitialAdListener) public interface InterstitialAdListener void onInterstitialAdLoaded() void onInterstitialAdFailedToLoad(AdiscopescopeError error) void onInterstitialAdOpened(String unitId) void onInterstitialAdClosed(String unitId) void onInterstitialAdFailedToShow(String unitId, AdiscopescopeError error) </pre> |

2. Methods

a. load

특정 유닛에 속한 ad 네트워크들의 Interstitial 광고를 load 한다. Interstitial Ad의 load와 show는 pair로 호출되어야 한다. 즉 load를 한 후 show를 하고, 광고를 show한 후에는 다시 load를 하여 다음 번 show를 준비하여야 한다.

| Definition |
|---------------------------------------|
| <pre> void load(String unitId) </pre> |
| Parameters |
| |

| | |
|---------------|--|
| unitId | load 할 광고의 unit id. Admin page에 등록된 id와 동일해야 한다. |
| Return | |
| | |

load가 실행되면 onInterstitialAdLoaded 와 onInterstitialAdFailedToLoad 중 하나의 callback은 항상 호출된다.

| Call back | | | |
|------------------------------|----------------------------|------------|------|
| onInterstitialAdLoaded | Interstitial를 load하였을 때 | Parameters | |
| | | | |
| onInterstitialAdFailedToLoad | Interstitial load가 실패하였을 때 | Parameters | |
| | | error | 에러코드 |

b. isLoaded

특정 유닛의 Interstitial 광고가 load 되었는 지를 확인한다.

| Definition |
|---------------------------------|
| boolean isLoaded(String unitId) |

| Parameters | |
|------------|---|
| unitId | load 여부를 체크할 광고의 unit id |
| Return | |
| bool | load된 광고가 있을 시 True, load된 광고가 없을 시 False |

c. Show

최근에 load된 Interstitial 광고 유닛에 속한 광고를 사용자에게 보여준다.

| Definition |
|----------------|
| boolean show() |

| Parameters | |
|------------|--|
| | |
| Return | |
| bool | show가 정상적으로 시작되면 True, 만약 이미 다른 show가 진행중이라면 False |

show가 실행되면 (return값이 True일 경우) onInterstitialAdOpened 와 onInterstitialAdFailedToShow 중 하나가 항상 호출되고, onInterstitialAdOpened가 호출되었다면 이후 onInterstitialAdClosed가 항상 호출된다.

| Call back | | | | | | | | |
|------------------------------|-------------------------------|---|------------|--|--------|-------|-------|------|
| onInterstitialAdOpened | Interstitial 광고창이 열릴 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr></table> | Parameters | | unitId | 유닛 Id | | |
| Parameters | | | | | | | | |
| unitId | 유닛 Id | | | | | | | |
| onInterstitialAdClosed | Interstitial 광고창이 닫혔을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr></table> | Parameters | | unitId | 유닛 Id | | |
| Parameters | | | | | | | | |
| unitId | 유닛 Id | | | | | | | |
| onInterstitialAdFailedToShow | Interstitial 광고창을 보여 줄 수 없을 때 | <table><tr><th colspan="2">Parameters</th></tr><tr><td>unitId</td><td>유닛 Id</td></tr><tr><td>error</td><td>에러코드</td></tr></table> | Parameters | | unitId | 유닛 Id | error | 에러코드 |
| Parameters | | | | | | | | |
| unitId | 유닛 Id | | | | | | | |
| error | 에러코드 | | | | | | | |

Example

```
package com.test.myapplication;

import android.app.Activity;
import android.os.Bundle;
```

```

import android.util.Log;

import com.nps.adiscope.AdiscopeSdk;
import com.nps.adiscope.AdiscopeError;
import com.nps.adiscope.interstitial.InterstitialAd;
import com.nps.adiscope.interstitial.InterstitialAdListener;

public class SampleActivity extends Activity implements InterstitialAdListener {

    InterstitialAd interstitialAd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AdiscopeSdk.initialize(this, "exampleMediaId", "exampleMediaSecret");
        AdiscopeSdk.setUserId("exampleUniqueUserId");

        interstitialAd = AdiscopeSdk.getInterstitialAdInstance(this);
        interstitialAd.setInterstitialAdListener(this);
    }

    @Override
    public void onInterstitialAdLoaded() {
        Log.d("adiscope", "onInterstitialAdLoaded ");
    }

    @Override
    public void onInterstitialAdFailedToLoad(AdiscopeError error) {
        Log.d("adiscope", "onInterstitialAdFailedToLoad " + error);
    }

    @Override
    public void onInterstitialAdOpened(String unitId) {
        Log.d("adiscope", "onInterstitialAdOpened " + unitId);
    }

    @Override
    public void onInterstitialAdClosed(String unitId) {
        Log.d("adiscope", "onInterstitialAdClosed " + unitId);
    }

    @Override
    public void onInterstitialAdFailedToShow(String unitId, AdiscopeError error) {
        Log.d("adiscope", "onInterstitialAdFailedToShow " + unitId + ", error " + error);
    }

    public void loadInterstitialAd() {
        interstitialAd.load("UNIT_ID");
    }

    public void showInterstitialAd() {
        if (interstitialAd.isLoaded("UNIT_ID")) {
            interstitialAd.show();
        }
    }
}

```

OptionSetter

1. Class Declaration

Declaration

```

package com.nps.adiscope;

public interface OptionSetter

public void setUseCloudFrontProxy(boolean useCloudFrontProxy)

```

2. Methods

a. **setUseCloudFrontProxy**

AWS Cloud Front Proxy를 사용할지 말지 설정한다.



이 옵션을 사용하게 되면, 북미, 유럽에서 게임을 서비스할시에 Adiscope API의 응답속도가 향상된다.

Definition

```
void setUseCloudFrontProxy(boolean useCloudFrontProxy)
```

Parameters

| useCloudFrontProxy | AWS Cloud Front Proxy를 사용할지 말지 여부 |
|--------------------|-----------------------------------|
| | |

Return

AdiscopeError

1. Class Declaration

Declaration

```
package com.nps.adiscope;  
  
public enum AdiscopeError
```

2. ErrorCode

| Code | Value | Description | Cause | Instruction |
|----------------------------------|-------|---|---|--|
| INTERNAL_ERROR | 0 | "Internal error" | Adiscope Sdk 내부 오류 혹은 Adiscope Server 오류 | 지속적으로 발생 시 Adiscope 개발팀에 문의 |
| | | | RewardedVideoAd.Show()를 Load() 없이 실행하였을 경우 | RewardedVideoAd.Show()하기 전 Load()를 호출하고, IsLoaded 값을 확인한 후 Show를 호출 |
| MEDIATION_ERROR | 1 | "3rd party mediation network error" | Mediation 광고 Network의 3rd party sdk 혹은 server 오류 | 지속적으로 발생 시 Adiscope 개발팀에 문의 |
| INITIALIZE_ERROR | 2 | "mediald /mediaSecret must be valid" | Adiscope.Sdk.Initialize 시 mediald /mediaSecret이 유효하지 않음 | Adiscope admin page에서 등록된 media (application)의 id와 secret을 확인 |
| SERVER_SETTINGS_ERROR | 3 | "Server settings are incorrect" | 광고를 보여주기 위해 필요한 내부 설정값 오류. AndroidManifest에 설정된 값이거나 Runtime 시 server로 부터 전달 받은 값이 정확하지 않음 | Adiscope admin page의 설정을 다시 확인하고 service.json을 다운로드하여 Adiscope Unity Sdk Bundle Importer를 이용하여 AndroidManifest를 다시 생성 |
| | | | Adiscope admin 설정의 수직화, 유닛 활성화가 OFF인 경우 | Adiscope admin page의 설정 확인 |
| INVALID_REQUEST | 4 | "The request is invalid" | Show() 시 입력한 unitId 오류 | Adiscope admin page에 정의된 각 unitId를 다시 확인 후 Show()에 입력 |
| NETWORK_ERROR | 5 | "There is a network problem" | Network read/write timed out 혹은 Network connection 오류 | Device의 network 연결 상태를 확인 |
| NO_FILL | 6 | "No more ads to show" | 하루에 볼 수 있는 Rewarded Video 광고의 횟수를 모두 소진 하였을 경우 | Adiscope admin page의 media (application)에 설정된 기준 시각이 지나면 광고 횟수가 다시 초기화 되므로 기준 시각 이후(next day)에 다시 시도 |
| TIME_LIMIT | 7 | "It was time-limited" | Rewarded Video 광고를 한번 보여주고 난 후 일정 시간 (30초~60초, Adiscope admin page에서 설정된 시간 간격)이 지나기 전에 다시 Show를 시도할 경우 | Adiscope admin page에서 설정된 시간 간격 만큼 간격을 두고 다시 시도 |
| NOT_EXIST_IDFA (Only iOS) | 8 | IDFA value is invalid | iOS 디바이스에서 추출된 IDFA 값이 "00000000-0000-0000-0000-000000000000" 인 경우 | iOS 디바이스 설정에서 "광고 추적 제한" 설정이 ON 일 경우에 발생되며, 이 경우, SDK에서는 사용자가 "광고 추적 제한" 설정을 OFF 하도록 유도하는 안내 문구를 System Alert으로 띄운다. 사용자가 "광고 추적 제한" 설정을 OFF 후 게임을 재실행하면 광고 참여가 가능하다 |
| GOOGLE_FAMILY_ERROR | 9 | "It is not available because of Google Family Policy" | 구글 가족정책에 의거, 사용할 수 없는 기능이 호출되었음을 의미 | 구글 가족정책 가이드 참고 |

| | | | | |
|-------------------------------|----|----|-----------|-----------------------------|
| (Only Android) | | | | |
| UNKNOWN_ERROR (Only Unity) | -1 | "" | 알 수 없는 오류 | 지속적으로 발생 시 Adiscope 개발팀에 문의 |

3. Example

Example for Error Handling

```
private void onRewardedVideoAdFailedToShow(String unitId, AdiscopeError error)
{
    switch (error)
    {
        case INTERNAL_ERROR:
            // adiscope core/server error
            // in case of RewardedVideoAd.show(), check load() is called before show()
            break;
        case MEDIATION_ERROR:
            // 3rd party mediation network sdk/service error
            break;
        case INITIALIZE_ERROR:
            // mediald & mediaSecret is not valid. check them again from admin page
            break;
        case SERVER_SETTING_ERROR:
            // settings from admin page error.
            // check the settings in admin page, download service.json and
            // update adiscope sdk using bundle importer again
            break;
        case INVALID_REQUEST:
            // unitId parameter error. check unitId from admin page
            break;
        case NETWORK_ERROR:
            // network timed out or connection error
            break;
        case NO_FILL:
            // there's no more ads to show today. do not retry until the next day.
            break;
        case TIME_LIMIT:
            // wait for 30~60 seconds (based on the setting in admin page) and try again to show
            break;
        default:
            break;
    }
}
```

- [Server-to-server Reward 정보 연동](#)
 - [메시지 Flow](#)
 - [설정](#)
 - [Callback Request](#)
 - [Callback 유효성 검증](#)
 - [Response](#)
 - [재시도 정책](#)
 - [Sample Code](#)

Server-to-server Reward 정보 연동

1. 메시지 Flow

- 광고를 통해 발생한 리워드는 3rd-party 서버로 부터 Adiscope 의 리워드 서버로 전달된다.
- 해당 내용은 Adiscope server내에서 적절히 가공된다.
- 마지막으로 Adiscope admin page에 등록된 Application/Game 서버로 callback request를 통해 전달되게 된다.

2. 설정

- Server-to-server callback 에 대한 정보를 Adiscope admin page 에 입력

- b. Adiscope admin page 에서 발급받은 secretKey 와 Callback 의 request param에 포함된 signature 값을 통해 검증

3. Callback Request

- a. Methods : GET
b. URL 내 Parameter의 Key는 자유롭게 설정할 수 있다. Adiscope service는 Value Placeholder의 이름을 기준으로 실제 data를 URL에 삽입하여 호출한다.
c. URL format :

example #1

https://yourserver.com/anypath/callback_url.php?transactionId=[TRANSACTION_ID]&signature=[SIGNATURE]&unitId=[UNIT_ID]&userId=[USER_ID]&ad=[ADID]&rewardUnit=[REWARD_UNIT]&rewardAmount=[REWARD_AMOUNT]"
(위 URL은 예시임. "transactionId", "signature", "unitId", "userId", "adid", "currency", "amount" 등의 key 값은 다른 이름으로 대체할 수 있음)

example #2

https://yourserver.com/anypath/another_callback_url.php?tid=[TRANSACTION_ID]&sign=[SIGNATURE]&unit=[UNIT_ID]&user=[USER_ID]&ad=[ADID]&rUnit=[REWARD_UNIT]&rAmount=[REWARD_AMOUNT]"
(위 URL은 예시임. "tid", "sign", "unit", "user", "ad", "rUnit", "rAmount" 등의 key 값은 다른 이름으로 대체할 수 있음)

- d. Request Parameters

| Sample Key | Value Placeholder | Description | Remark |
|---------------|-------------------|--|--------|
| transactionId | [TRANSACTION_ID] | transaction id. 중복호출여부 확인용. (각각의 트랜잭션 요청 단위를 구분하는 유니크 값으로서 앱에서는 이 값으로 같은 요청이 중복해서 유입된것지를 체크할 수 있다) | |
| signature | [SIGNATURE] | Callback 유효성 검증에 사용되는 MD5 hash값 | |
| unitId | [UNIT_ID] | 광고 유닛 ID | |
| userId | [USER_ID] | 클라이언트에서 설정한 custom user ID | |
| adid | [ADID] | 구글 adid / iOS는 idfa | |
| currency | [REWARD_UNIT] | 보상 화폐 단위 | |
| amount | [REWARD_AMOUNT] | 보상 지급 수량 | |

4. Callback 유효성 검증

- a. Adiscope admin 에서 발급받은 secretKey 와 Callback 의 request param에 포함된 signature 값을 통해 검증
b. "signature" is the message authentication code with MD5 hash

Pseudo Code Example

```
// Pseudo code
if (HMACMD5(concat([USER_ID], [REWARD_UNIT], [REWARD_AMOUNT], [TRANSACTION_ID]), secretKey) == signature)
{
    // do task
}
else
{
    // verification failed
}
```

5. Response

- a. Adiscope server가 Application/Game 서버의 callback URL을 호출하여 reward 정보를 전달하였을 때 Application/Game 서버는 다음과 같이 HTTP response를 생성해야 한다.

| Case | HTTP Status Code | HTTP Response Body |
|-----------------|------------------|--------------------|
| 성공 시 | 200 (Success) | "OK" |
| 유효성 검증 실패일 경우 | | "SIGNATURE_ERROR" |
| 중복된 reward 일 경우 | | "DUPLICATED" |

6. 재시도 정책

- a. HTTP Status code가 200이 아닌 경우 Adiscope server는 adiscope내의 정해진 재시도 시간의 간격으로 Application/Game Server의 callback URL을 지속적으로 다시 호출 한다.
- adiscope내의 정해진 재시도 시간의 간격은
1번째 시도 시 대기시간 10초후에 재시도
2번째 시도 시 대기시간 1분후에 재시도
3번째 시도 시 대기시간 10분후에 재시도
4번째 시도 시 대기시간 30분후에 재시도

5번째 시도 시 대기시간 1시간후에 재시도
6번째 시도 시 대기시간 6시간후에 재시도
7번째 시도 시 대기시간 6시간후에 재시도
8번째 시도 시 대기시간 12시간후에 재시도
9번째 시도 시 대기시간 12시간후에 재시도
10번째 시도 시 대기시간 12시간후에 재시도
그 이후 실패 로그를 DB에 저장하며 재시도 하지 않는다.

7. Sample Code

Node.js Example

```
var express = require('express');
var crypto = require('crypto')
var secret = "YOUR_SECRET_KEY";
app.listen(process.env.PORT || 3412);
app.get('/', function (req, res) {
  var userId = req.query.userId;
  var rewardUnit = req.query.rewardUnit;
  var rewardAmount = req.query.rewardAmount;
  var transactionId = req.query.transactionId;
  var signature = req.query.signature;
  var plainText = makePlainText(userId, rewardUnit, rewardAmount, transactionId);
  var hmac = getHMAC(plainText, secret);
  console.log(hmac);
  console.log(signature);

  // Signatures checking
  if (hmac === signature) {

    // Check for duplicated transaction id here (whether player already has received the reward)
    if (!isDuplicatedReward(transactionId)) {
      // If there's no duplicate – give virtual goods to player
      // and store the transaction id for duplicate checking.
      giveReward(transactionId);

      // On success, return 200 and include 'OK' in the HTTP body
      res.status(200).send('OK');
    } else {
      // reward already received by user
      res.status(200).send('DUPLICATED');
    }
  } else {
    // signature error
    res.status(200).send('SIGNATURE_ERROR');
  }
});

function getHMAC(plainText, secret) {
  return crypto.createHmac('md5', secret).update(plainText).digest('hex');
}

function makePlainText(userId, rewardUnit, rewardAmount, transactionId) {
  return userId.concat(rewardUnit, rewardAmount, transactionId);
}

function isDuplicatedReward(transactionId) {
  // check transaction id is duplicated or not
  return false;
}

function giveReward(transactionId) {
  // give reward to user
  // store transaction id for future checking
}
```

- Adiscope SDK Version 1.0.804.191016 (aOS, unity)
해당 릴리즈는 구글 가족 정책을 준수할 수 있도록 지원해주는 기능을 포함하고 있습니다.

- Google Play Console에서 설정한 대상 연령대가 13세 미만에 속하는 경우(0~5세, 6~8세, 9~12세) 구글 가족 정책의 대상이 됩니다. 정책을 준수하지 않을 경우, 앱이 삭제 또는 일시 정지될 수 있습니다.

해당 경우, 애디스콧 측으로 2가지 정보 전달이 필요합니다. (상세 내용은 하단 1),2) 내용 참고부탁드립니다.)

- **Google Play Console에서 설정한 대상 연령대 (필수)**
: 애디스콧 매체 등록 시 활용되며, 해당 값을 기준으로 구글 가족 정책의 적용을 받습니다. 애디스콧 담당자에게 Google Play console에서 설정한 대상 연령대 정보를 필히 전달해주셔야 합니다.
- **어린이 여부(childYN) 파라미터 값 (선택)**
: 사용자의 어린이 여부 전달이 필요합니다. 값을 전달하지 않는 경우, 어린이/비어린이 여부에 관계없이 광고에 제한이 생기기에(광고 물량이 축소될 수 있음) 선택사항이지만 해당 파라미터 값을 전달하는 것을 권장합니다.
(보상형영상광고 : 구글 플레이 자체 인증 광고 네트워크로만 송출, 오퍼월 : 진입불가)

1) 구글 가족 정책이란?

구글 가족 정책은 어린이가 Google Play를 안전하게 사용할 수 있도록 하는 정책으로, 2019년 9월 1일부터 시행되었습니다.

어린이 연령대를 타겟층으로 한 어플리케이션인 경우 가족 정책을 필수로 준수해야하며, 광고 게재 시 Google Play 자체 인증 광고 네트워크 프로그램에 속한 광고만 표시됩니다.

- 대상 : Google Play Console의 타겟층에 13세 미만에 속하는 연령대(또는 거주 국가에서 적용되는 적정 연령)를 선택한 경우 (13세 미만 연령대가 포함되지 않는다고 선언한 경우 별도의 조치를 취하지 않아도 됩니다.)
- 내용 : 13세 미만의 사용자에게는 Google Play 자체 인증 광고 네트워크 프로그램에 속한 광고만 표시됨

2) 파라미터 값 (매체→애디스콧)

구글 가족 정책 대상인 어플리케이션인지 Google 앱 타겟층 설정을 확인 부탁드립니다. 해당 경우, 애디스콧으로 어린이 여부(childYN) 파라미터 전달이 필요합니다.

어린이 여부 파라미터 값은 아래 3가지 중 하나로 전달이 필요하며, 값에 따라 광고 표시 조건이 달라집니다.

- YES: 어린이인 경우
- NO: 어린이가 아닌 경우
- (null) : 어린이 여부 파라미터를 전달 주지 않는 경우

어린이 여부 파라미터 값이 YES 또는 (null)인 경우, 구글 가족 정책에 따라 Google Play 자체 인증 광고 네트워크 프로그램에 속한 광고만 표시되며, 오퍼월은 진입 불가합니다.**

**** 오퍼월 진입 시 어린이 여부 파라미터 값이 YES 또는 (null)인 경우 아래 에러코드가 전달됩니다.**
GOOGLE_FAMILY_ERROR(9, "It is not available because of Google Family Policy");
해당 경우 “어린이인 경우 무료 충전소에 참여할 수 없습니다” 문구로 팝업 표시가 필요합니다.

[적용 예시] (구글 가족정책 준수를 위해 어린이 여부를 설정하는 샘플코드)

Android

| setting childYN |
|--|
| String childYN = "YES" // "YES" or "NO" |
| AdiscopeSdk.getOptionSetterInstance(this).setChildYN(childYN); |

Unity-Android

| setting childYN |
|---|
| string childYN = "YES" // "YES" or "NO" |
| Adiscope.Sdk.GetOptionSetter().SetChildYN(childYN) |

React Native-Android

| setting childYN |
|---|
| var childYN = "YES" // "YES" or "NO" |
| Adiscope.OptionSetter.setChildYN(childYN) |