AIQ.AWARE

Android SDK Developer's Guide

Ver.06

Release date: May 15, 2020

Skelter Labs

Overview

스켈터랩스의 AIQ.AWARE 서비스는 초개인화(Hyper-personalization) 솔루션을 위한 API를 제공합니다. 수집한 고객의 데이터를 바탕으로 초개인화 기능을 쉽고 올바르게 사용하기 위해서는 스켈터랩스에서 제공하는 AIQ.AWARE Android SDK가 필요합니다.

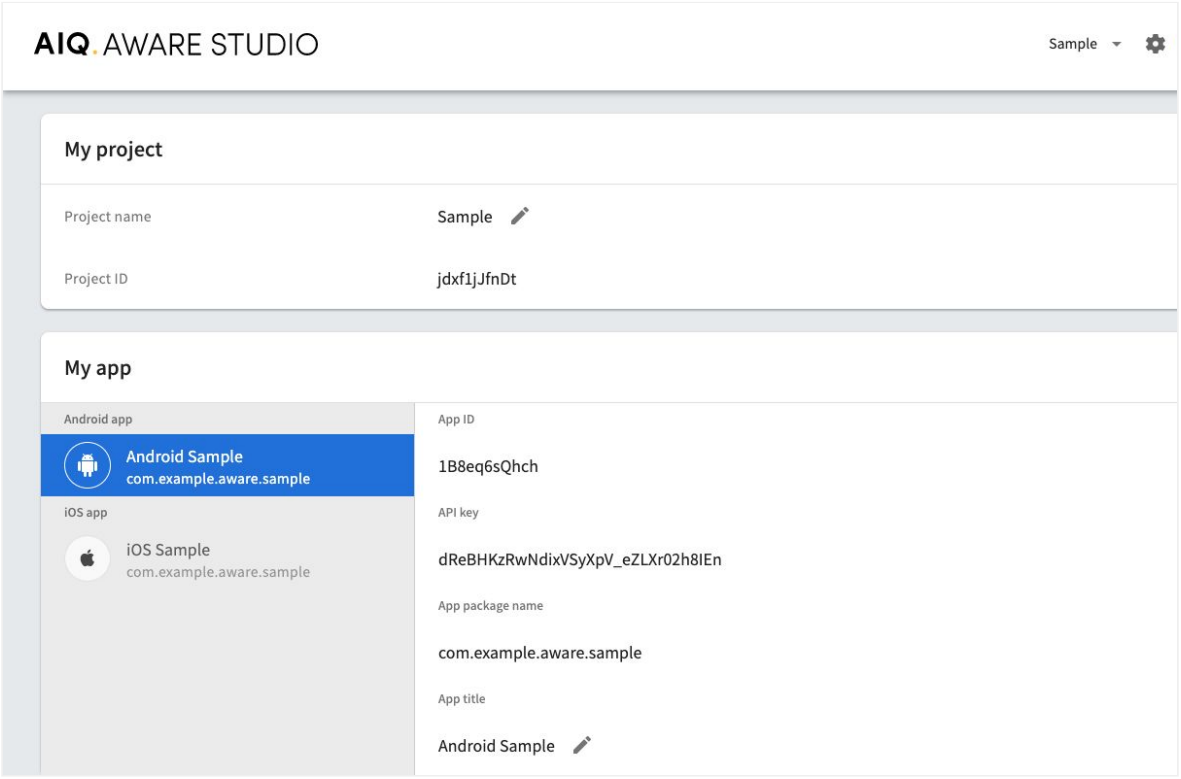
본 문서는 AIQ.AWARE 서비스를 사용하기 위한 기본 개발환경 구성 및 Android 애플리케이션에서 필요한 SDK 사용법 뿐 아니라 AIQ.AWARE 서비스를 통해 제공되는 정보의 상세 설명을 포함하고 있습니다.

프로젝트 및 앱 생성

프로젝트 생성 요청

AIQ.AWARE Android SDK를 사용하기 위해선 프로젝트 및 Android 앱이 생성되어야 합니다.

프로젝트를 사용하려면 https://aware.skelterlabs.com/studio 에서 회원가입후, 신청 서베이를 통해 앱 신청을 요청하실 수 있습니다.



[프로젝트 설정 화면]

개발환경 구성

Gradle 설정

AIQ.AWARE SDK는 gradle을 통해 환경을 구성할 수 있습니다. 현재 가장 최신 버전의 AWARE_SDK_VERSION 은 0.2.7 입니다.

```
// app build.gradle

repositories {
  ...
  maven { url 'https://jitpack.io' }
  jcenter()
}

dependencies {
  // AIQ.AWARE 서비스를 사용하기 위해 필요
  implementation 'com.skelterlabs:aware:0.2.7'

  // (Optional) RxJava를 이용한 register/unregister 함수를 사용하는 경우 필요
  implementation 'io.reactivex.rxjava2:rxandroid:2.1.0'
}
```

권한 설정

AndroidManifest.xml파일에 정보 수집에 필요한 AIQ.AWARE SDK 권한을 설정해야 합니다.

```
<!-- SDK를 사용하기 위한 필수 권한 -->
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<!-- targetSDKVersion이 Android API Level 28 이상인 경우 -->
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
<!-- targetSDKVersion이 Android API Level 29 이상인 경우 -->
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />

<!-- 그 외 설정할 권한은 ‘권한 별 수집 가능 Signals’ 섹션을 확인하시기 바랍니다. -->
```

프로가드 설정

```
# proguard-rules.pro

# Keep AIQ.Aware classes
-keep class com.skelterlabs.aware.** { *; }

### Gradle 3.4.0 미만을 사용하는 경우 아래 rules를 같이 추가해주어야 합니다.
# Joda Time 2.3
-dontwarn org.joda.convert.**
-dontwarn org.joda.time.**
-keep class org.joda.time.** { *; }
-keep interface org.joda.time.** { *; }

# Compile fails w/o these.
-dontwarn org.apache.**
-keep class javax.**
```

앱 설정

src/main/assets에 aware-client-config.json 파일을 생성합니다. 생성된 JSON 파일에는 다음과 같이 앱 설정을 합니다. AIQ.AWARE Android SDK를 통해 고객의 정보를 수집하려면 제공되는 앱 정보를 사용해야 합니다. 앱 정보는 생성된 Studio 프로젝트 페이지를 확인하시기 바랍니다.

```
{
  “version”: 1,
  “project_id”: “AAAAAAAAAA”,
  “app_id”: “AAAAAAAAAA”,
  “api_key”: “AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA”
}
```

AIQ.AWARE 서비스 시작

SDK 초기화

AIQ.AWARE SDK를 초기화 후 사용 등록을 해야 고객 정보 수집을 시작할 수 있습니다.

```
// Java
public class SampleApp extends Application {
  @Override
  public void onCreate() {
    final AIQAware aiqAware = AIQAwareApp.getInstance(this);
    if (aiqAware.init()) {
      Log.d(TAG, "Initialized");
    }
  }
```

```
}
```

```
// Kotlin
class SampleApp: Application() {
  override fun onCreate() {
    val aiqAware = AIQAwareApp.getInstance(application)
    if (aiqAware.init()) {
      Log.d(TAG, "Initialized")
    }
  }
}
```

SDK 사용 등록

사용 등록을 한 시점부터 고객 정보 수집이 시작됩니다.

```
// Java
final AIQAware aiqAware = AIQAwareApp.getInstance(getApplication());
aiqAware.register(null, null, new AIQAware.Callback<String>() {
  @Override
  public void onSuccess(String token) {
    Log.d(TAG, "Registered: " + token);
  }

  @Override
  public void onError(AIQAwareException e) {
    Log.e(TAG, "Error: " + e.getMessage());
  }
});
```

```
// Kotlin
val aiqAware = AIQAwareApp.getInstance(application)
aware.register(null, null, object : AIQAware.Callback<String> {
  override fun onSuccess(token: String) {
    Log.d(TAG, "Registered: $token")
  }

  override fun onError(e: AIQAwareException) {
    Log.e(TAG, "Error: ${e.message}")
  }
})
```

SDK 사용 해제

사용 해제를 하면 SDK에서 더 이상 고객의 정보 수집을 하지 않습니다. 사용 등록을 하는 시점부터는 항상 새로운 고객으로 처리하기 때문에 정확한 정보 수집 및 분석을 위해서라면 꼭 필요한 경우 외에는 SDK 사용 해제는 하지 말아야 합니다.

```
// Java
AIQAware aiqAware = AIQAware.getInstance(getApplication());
```

```
aiqAware.unregister(new AIQAware.Callback<Unit>() {
  @Override
  public void onSuccess(Unit result) {
    Log.d(TAG, "Unregistered");
  }

  @Override
  public void onError(AIQAwareException e) {
    Log.d(TAG, "Error: " + e.getMessage());
  }
});
```

```
// Kotlin
val aiqAware = AIQAware.getInstance(application)
aiqAware.unregister(object : AIQAware.Callback<Unit> {
  override fun onSuccess(result: Unit) {
    Log.d(TAG, "Unregistered")
  }

  override fun onError(e: AIQAwareException) {
    Log.d(TAG, "Error: " + e.message)
  }
})
```

권한 별 수집 가능 Signals

고객으로부터 다음의 권한을 획득하면 더 정확한 고객 프로파일링이 가능합니다.

Signals	Permission
Notification	Settings > Search > Notification access 에서 앱 선택

샘플앱 실행

AIQ.AWARE github에서 샘플 앱을 찾아보실 수 있습니다.
https://github.com/SkelterLabsInc/aware-android-sdk

문제 해결

Android SDK 버전이 맞지 않는 경우

AIQ.AWARE는 현재 minSdkVersion이 19 미만이거나 targetSdkVersion이 29 미만인 서비스는 지원하지 않습니다. 모듈 레벨의 build.gradle를 확인하여 minSdkVersion를 19 이상으로, targetSdkVersion를 29 이상으로 설정해 주시기 바랍니다.

AndroidX 이전 버전을 사용 중인 경우

Migrating to AndroidX 문서를 통해 AndroidX Migration을 진행해주시기 바랍니다.

Protobuf 라이브러리를 사용 중인 경우

protobuf-lite로 Migration을 진행해 주시기 바랍니다. Android에서 protobuf 보다 protobuf-lite가 권장되는 이유에 대해서는 이 문서를 참고하시기 바랍니다.

Firebase Cloud Messaging 메시지 사용 시

AIQ.AWARE Android SDK는 AIQ.AWARE 서비스를 통해 생성된 정보를 개발된 애플리케이션에 전달하기 위해 Firebase Cloud Messaging 서비스를 이용하고 있습니다.

중첩된 Firebase Cloud Messaging 서비스의 사용으로 인한 메시지 소실을 막고 고객의 데이터 수집을 위해서는 추가 설정이 필요합니다.

직접 Firebase Cloud Messaging 서비스의 메시지 처리 등을 위해 FirebaseMessagingService를 확장한 경우에만 아래 문서를 참고하시기 바랍니다.

FirebaseMessagingService

Firebase Cloud Messaging 서비스를 사용하고 수신된 메시지를 직접 처리하는 경우 FirebaseMessagingService를 확장해야 합니다. FirebaseMessagingService의 자세한 내용은 Firebase 문서를 확인하시기 바랍니다.

확장한 FirebaseMessagingService에서 onMessageReceived의 함수에서는 AIQ.AWARE의 handleAiqAwareFcmMessage 함수를 수신된 메시지와 함께 호출해야 하고, onNewToken에서는 updateAiqAwareFcmToken를 호출해야합니다.

```
// Java
public class FcmMessagingService extends FirebaseMessagingService {

  private AIQAware aiqAware;

  @Override
  public void onCreate() {
    super.onCreate();
    aiqAware = AIQAware.getInstance(getApplication());
  }

  @Override
  public void onNewToken(token String) {
    aiqAware.updateAiqAwareFcmToken()
  }

  @Override
  public void onMessageReceived(RemoteMessage remoteMessage) {
    if (aiqAware.handleAiqAwareFcmMessage(remoteMessage)) {
      return;
    }
    // Handle client application logic.
  }
}
```

```
// Kotlin
class FcmMessagingService : FirebaseMessagingService() {

  lateinit var aiqAware: AIQAware

  override fun onCreate() {
    super.onCreate()
    aiqAware = AIQAware.getInstance(application)
  }

  override fun onNewToken(token: String) {
    aiqAware.updateAiqAwareFcmToken()
  }

  override fun onMessageReceived(remoteMessage: RemoteMessage) {
    if (aiqAware.handleAiqAwareFcmMessage(remoteMessage)) {
      return
    }
    // Handle client application logic.
  }
}
```

AIQ.AWARE SDK API

AIQAwareApp

AIQAware Instance를 access하고 constant variable을 담고있는 class입니다.

Name	Return	Description
`static getInstance(` `  Application app)`	`AIQAware`	AIQ.AWARE SDK사용을 위해 Singleton instance를 생성합니다.

AIQAware

AIQ.AWARE SDK의 기능 사용을 위한 interface입니다. 자세한 정보는 API 문서를 참고 바랍니다.

Name	Return	Description
`init()`	`boolean`	AIQ.AWARE SDK를 초기화합니다. 만약 필요한 설정이 되어있지 않으면 false를 반환합니다.
`isRegistered()`	`boolean`	AIQ.AWARE SDK 사용 등록이 되었는지 여부를 반환합니다.

`register(`
`  String userHint,`
`  String ctoken,`
`  Callback<String> callback)`
		사용자 기기를 등록합니다. callback을 통해 AIQ.AWARE API에 사용될 토큰이 반환됩니다. userHint에는 사용자를 구별하기 위한 ID나 유니크한 값을 지정할 수 있습니다. 관리를 원하지 않는 경우 null 값을 입력하면 됩니다.

`register(`
`  String userHint,`
`  String ctoken)`
	`Single<String>`	Callback대신 RxJava를 이용한 register 함수 입니다. subscribe를 해야만 동작합니다.

`unregister(` `  Callback<Unit> callback)`		사용자 기기 등록을 해제합니다. 성공 실패 여부가 callback을 통해 반환됩니다.
`unregister()`	`Completable`	Callback대신 RxJava를 이용한 unregister 함수 입니다. subscribe를 해야만 동작합니다.
`isServiceEnabled()`	`boolean`	서비스 사용이 활성화 되었는지 여부를 반환합니다. 활성화되어 있지 않으면 권한이 있더라도 고객의 정보를 수집하지 않습니다.
`setEnableService(` `  Boolean enabled)`		서비스의 사용/비활성을 설정합니다.

`updateAiqAwareFcmToken()`		AIQ.AWARE SDK의 FCM 메시지 토큰을 업데이트 합니다.
`handleAiqAwareFcmMessage(` `  RemoteMessage message)`	`boolean`	AIQ.AWARE SDK의 FCM 메시지를 처리합니다. 메시지 처리가 완료되면 true를 반환합니다.
`getProfile(` `  Callback<String> callback)`		고객의 프로필 정보를 callback을 통해 JSON string 형태로 반환합니다.

상태 변경 이벤트

BroadcastReceiver를 통해 상태 변경 이벤트를 받을 수 있습니다.
전달 받은 이벤트는 JSON string 형태로 되어있어서 원하는대로 parsing해서 사용할 수 있습니다.

```
// AndroidManifest.xml
...
<application ...>
  ...
  <receiver android:name="your.package.ActivityEventReceiver">
    <intent-filter>
      <action android:name="com.skelterlabs.aware.action.ACTIVITY_EVENT"/>
    </intent-filter>
  </receiver>
  ...
</application>
```

```
// Java: your.package.ActivityEventReceiver.java
public class ActivityEventReceiver extends BroadcastReceiver {

  @Override
  public void onReceive(Context context, Intent intent) {
    if (AIQAwareApp.ACTION_ACTIVITY_EVENT.equals(intent.getAction())) {
      // JSON format.
      String activityEvent = intent.getStringExtra(AIQAwareApp.EXTRA_ACTIVITY);
      if (activityEvent != null) {
        Log.d("TAG", "ActivityEvent Received: " + activityEvent);
      }
    }
  }
}
```

```
// Kotlin: your.package.ActivityEventReceiver.kt
class ActivityEventReceiver : BroadcastReceiver() {

  override fun onReceive(context: Context, intent: Intent) {
    if (intent.action == AIQAwareApp.ACTION_ACTIVITY_EVENT) {
      // JSON format.
      val activityEvent = intent.getStringExtra(AIQAwareApp.EXTRA_ACTIVITY)
      if (activityEvent != null) {
        Log.d("TAG", "ActivityEvent Received: $activityEvent")
      }
```

```
    }
  }
}
```

상태 변경 이벤트 데이터의 자세한 형식은 AIQ.AWARE API document를 참고하시기 바랍니다.

AIQ.AWARE 알림 UI 설정

주기적으로 데이터를 수집할 때, 나타나는 알림(Notification)에 대한 UI를 AndroidManifest.xml파일의 application meta-data를 통해 설정할 수 있습니다.

```
// AndroidManifest.xml
...
<application ...>
  ...
  <meta-data
    android:name="com.skelterlabs.aware.sdk.noti_channel_id"
    android:resource="@string/default_noti_channel_id" />
  <meta-data
    android:name="com.skelterlabs.aware.sdk.noti_color"
    android:resource="@color/colorAccent" />
  <meta-data
    android:name="com.skelterlabs.aware.sdk.noti_icon"
    android:resource="@drawable/ic_whatshot" />
  <meta-data
    android:name="com.skelterlabs.aware.sdk.service_channel_name"
    android:resource="@string/aware_service_channel_name" />
  <meta-data
    android:name="com.skelterlabs.aware.sdk.service_message"
    android:resource="@string/aware_service_message" />
  ...
</application>
```

name	resource	description
com.skelterlabs.aware.sdk.noti_channel_id	@string	기본 알림 채널 ID (기존에 생성된 채널 필요)
com.skelterlabs.aware.sdk.noti_color	@color	기본 알림의 색상
com.skelterlabs.aware.sdk.noti_icon	@drawable	기본 알림의 아이콘
com.skelterlabs.aware.sdk.service_channel_name	@string	AIQ.AWARE 서비스의 이름
com.skelterlabs.aware.sdk.service_message	@string	AIQ.AWARE 수집 메시지

기타 문의

AIQ.AWARE Android SDK에 대해 추가적으로 문의사항이 있다면 aware-support@skelterlabs.com 으로 문의해 주시기 바랍니다.