BaaS SDK(Android 版)
利用マニュアル

3.1.0 版

NSW 日本システムウエア株式会社

改版履歴

版数	改版日	改版内容
1.0.0	2014/03/07	新規作成
2.0.0	2014/05/07	「3.8 ビーコンを利用する」「3.9 コンテンツを利用する」を追記
3.0.0	2014/09/08	「3.1.2 SDK 処理化処理の呼び出し」に登録処理完了についての記載を追記
3.1.0	2014/12/08	対応 OS バージョンを Android2.3 へ変更。

はじめに

本マニュアルでは、ポスモバにお申込み頂いた方を対象に、Android 版 SDK の利用方法を説明します。

1. 動作環境

1.1. 対応 OS

本 SDK は Android2.3 以降に対応しています。
※ Android3.0 以降用のスタイル、Bluetooth4.0 の機能を参照しているため、targetSdkVersion は”19”を指定していますが、Android2.3 で動作可能です。
※ GooglePlayServices の Android2.2 サポート終了により、SDKv3.1.0 より対応 OS バージョンを Android2.3 以降と変更しました。

1.2. 必要な外部ライブラリ

本 SDK は以下のライブラリを使用しています。

- google-gson 2.2.4 以降 ：https://code.google.com/p/google-gson/
- Google Play Services：http://developer.android.com/google/play-services/index.html
- Android Support Library - v4：http://developer.android.com/tools/support-library/index.html

これらのライブラリの導入方法も含め、「2. セットアップ」で説明いたします。

1.3. 開発環境

以降の説明では、Eclipse (Helios 以降) を使用しています。

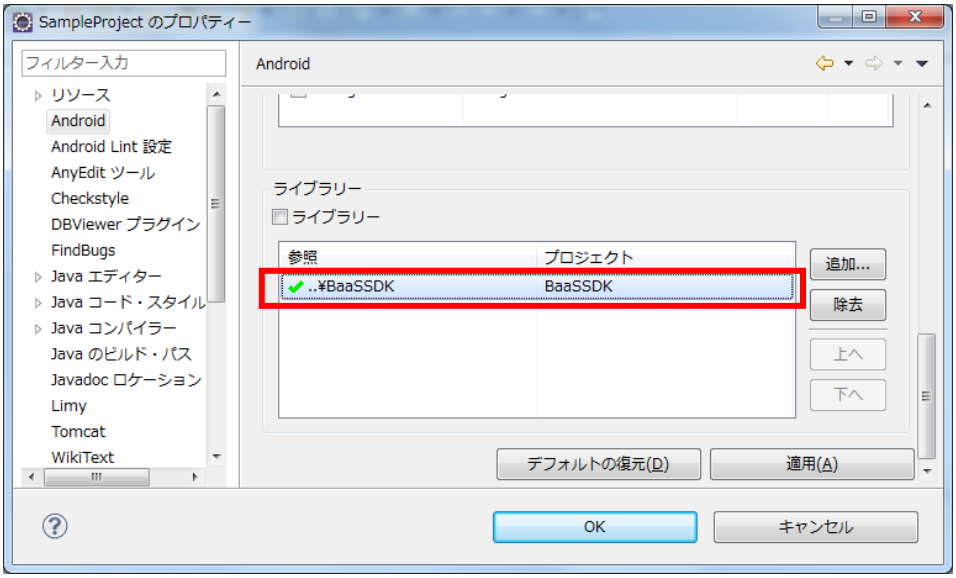
2. セットアップ

2.1. プロジェクトへの導入方法

ここでは、「SampleProject」という名前の Android プロジェクトへの導入方法を説明します。

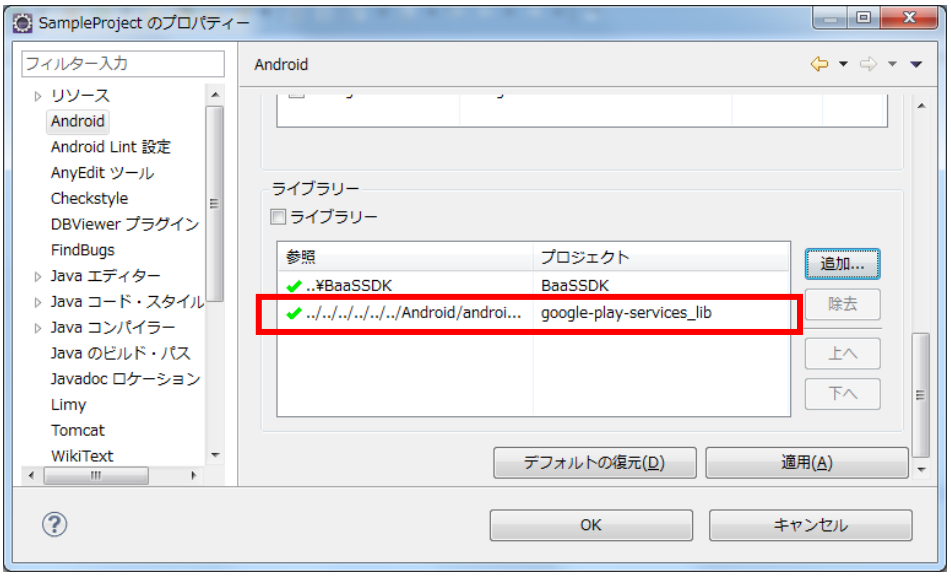
1. BaaS SDK を Android ライブラリとして参照する

ダウンロードした「BaaSSDK_Android_3.1.0.zip」を解凍し、「BaaSSDK」をワークスペースにインポート後、SampleProject のプロパティで Android ライブラリとして「BaaSSDK」を追加します。

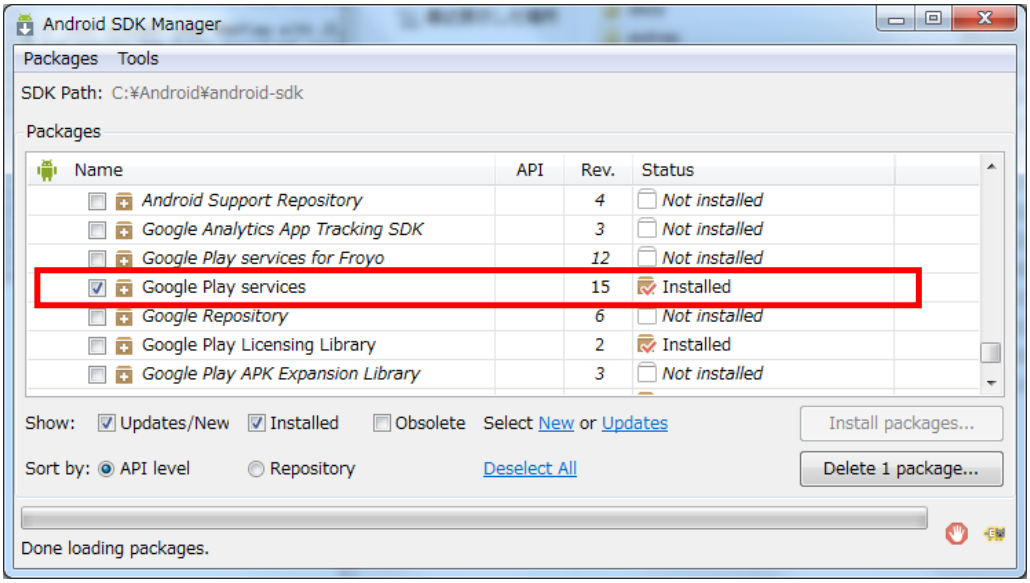


2. Google Play Services を Android ライブラリとして参照する

同様に、GooglePlayServices を Android ライブラリとして追加します。
GooglePlayServices のライブラリプロジェクトは、AndroidSDK インストールフォルダの「¥extras¥google¥google_play_services¥libproject」にあります。



GooglePlayServices のライブラリプロジェクトが存在しない場合は、Android SDK Manager から「Google Play Services」をインストールしてください。

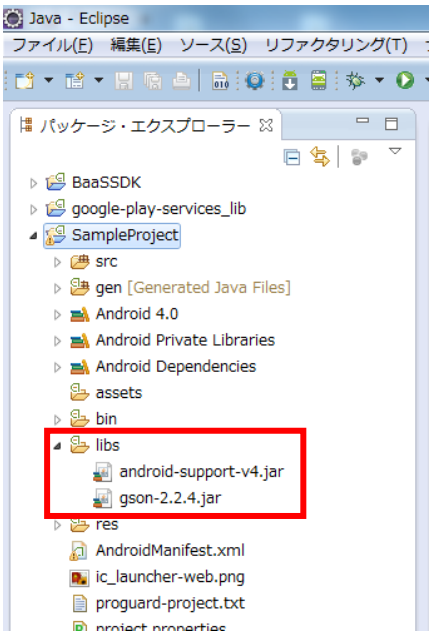


3. google-gson 及び、Android Support Library -v4 をライブラリ参照する

google-gson をダウンロードし、解凍した「gson-x.x.x.jar」を SampleProject の libs フォルダにコピーしてください。

- google-gson：https://code.google.com/p/google-gson/

Android Support Library - v4 のライブラリは、AndroidSDK インストールフォルダの「¥extras¥android¥compatibility¥v4」にあります。



以上でセットアップの手順は終了です。

3. 利用方法

3.1. プッシュ通知を利用する

本 SDK を使用し、プッシュ通知を利用するために必要な内容を説明いたします。

3.1.1 AndroidManifest.xml への追記

AndroidManifest.xml に、本 SDK を使用するうえで必要となるパーミッションの設定を行います。
<application>タグの前に下記のようにパーミッションを指定してください。

```
<!-- パーミッション指定。com.example.sampleproject の部分はアプリのパッケージ名に合わせて変更してください。 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<permission
    android:name="com.example.sampleproject.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />
<uses-permission android:name="com.example.sampleproject.permission.C2D_MESSAGE" />

<!-- バイブレートを使用しない場合は必要ありません。-->
<uses-permission android:name="android.permission.VIBRATE"/>
```

「com.example.sampleproject」の部分は、実際のアプリケーションのパッケージ名に変更してください。

続いて、Service, Activity, Receiver の設定を行います。
<application>タグの中に、下記のように設定してください。

```
<!-- Service、Activity、Receiver の指定。
com.example.sampleproject の部分はアプリのパッケージ名に合わせて変更してください。-->
<meta-data  android:name="com.google.android.gms.version"
       android:value="@integer/google_play_services_version"/>
<receiver
    android:name="jp.co.nsw.baassdk.PushReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.example.sampleproject" />
    </intent-filter>
</receiver>
<activity
    android:name="jp.co.nsw.baassdk.NotifyActivity"
    android:excludeFromRecents="true"
    android:exported="true"
    android:launchMode="singleTop"
    android:theme="@style/Theme.Nbsdk.NotifyDialog" >
</activity>
<service android:name="jp.co.nsw.baassdk.PushIntentService" android:exported="false">
    <intent-filter>
        <action android:name="nbsdk.intent.action.PUSH"/>
        <action android:name="nbsdk.intent.action.DIRECT"/>
    </intent-filter>
</service>
```

こちらも「com.example.sampleproject」の部分は、実際のアプリケーションのパッケージ名に変更してください。

以上で AndroidManifest.xml への追記は終了です。

3.1.2 SDK 初期化処理の呼び出し

Activity の onCreate()にて、本 SDK の初期化処理を呼び出します。
また、onResume(), onPause()にて、SDK の resume(), pause()をそれぞれ呼び出します。

```
package com.example.sampleproject;

import jp.co.nsw.baassdk.NswBaaSManager;
import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

    private NswBaaSManager mManager = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // NswBaaSManager インスタンスの取得
        mManager = NswBaaSManager.getInstanse(getApplicationContext());

        // 初期化を実行
        mManager.init("[Google Developers Console で取得した Project Number を指定する]",
         "[ポスモバ Web で発行された SDK ID を指定する]");

        …省略…
    }

    @Override
    protected void onPause() {
        super.onPause();
        // pause()をコールする
        mManager.pause();
    }
    @Override
    protected void onResume() {
        super.onResume();
        // resume()をコールする
        mManager.resume();
    }
```

本 SDK は、resume()メソッドが呼び出されてから pause()メソッドが呼び出されるまでの間、通信エラーがあった場合のリトライを実施しています。そのため、スプラッシュ画面のようなすぐに終了してしまう Activity ではなく、アプリケーションのメイン画面の Activity から上記呼び出しを行うようにしてください。

mManager.init()にてポスモバ Web への登録処理が実行されますが、本登録処理が完了した後にその他の API が利用可能となります。そのため、ランドマークやビーコンの利用をアプリ起動直後に実施したいような場合は、NswBaaSManager.isRegistered() と NswBaaSManager.init (java.lang.String project_number, java.lang.String sdk_id, RegisterCallback callback) を利用して登録処理が完了しているかを判定します。詳細は API リファレンスおよびサンプルコードをご確認ください。

以上で初期化処理の呼び出しは終了です。

3.1.3 SDK パーミッションの設定

本 SDK では、「プッシュ通知を受信したくない」、「位置情報検索を使用したくない」ユーザ向けに、
それぞれの有効／無効を切り替える機能があります。

```
// SDK のパーミッションを指定。以下の指定では、「Push 通知を受信する」「位置情報検索を使用する」。
mManager.setPermission(true, true);
```

本 SDK のデフォルトでは「プッシュ通知を受信する」「位置情報検索を使用する」
(= setPermission(true,true))
にて動作するようになっているため、ユーザの切り替えではなく常に位置情報検索を使用したくないような
場合は、初期化処理 init()の呼び出し直後に下記のような呼び出しを行ってください。

```
// プッシュ通知は設定値、位置情報検索は常に false を設定。
mManager.setPermission(mManager.getPushPermission(), false);
```

以上で終了です。

3.1.4 アプリケーションのパッケージ名について

本 SDK を使用するアプリケーションのパッケージ名は、ポスモバ Web で登録したアプリケーションのパッ
ケージ名と一致するようにしてください。

3.2. 位置連動配信を利用する

ポスモバでは、位置連動配信を利用できます。
位置連動配信とは、ユーザが特定のランドマークに近づいた時にメッセージを表示する機能です。
「3.1. プッシュ通知を利用する」の手順を実施したうえで、位置連動配信用の追加の設定を行うことで利用可能となります。

また、位置連動配信は「3.1.3. SDK パーミッションの設定」にて、「位置情報検索を使用する」を設定した場合のみ動作します。

3.2.1 動作条件

本機能は、端末設定で Wi-Fi が ON になっていることが動作条件となります。
Android4.3 以降の場合は Wi-Fi 詳細設定の「スキャンを常に実行する」が ON であれば、Wi-Fi が OFF でも動作可能です。

※上記設定が行われていない場合、位置情報検索の精度が落ちてしまうため正確なメッセージ配信ができなくなってしまいます。

3.2.2 AndroidManifest.xml への追記

AndroidManifest.xml に、パーミッションの追加を行います。
<application>タグの前に下記のようにパーミッションを指定してください。

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

続いて、Service, Receiver の設定を行います。
<application>タグの中に、下記のように設定してください。

```
<receiver
    android:name="jp.co.nsw.baassdk.RestartReceiver"
    android:permission="android.permission.RECEIVE_BOOT_COMPLETED">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.PACKAGE_REPLACED"/>
    </intent-filter>
</receiver>
<service android:name="jp.co.nsw.baassdk.LocationIntentService" android:exported="true"/>
<service android:name="jp.co.nsw.baassdk.GeofenceIntentService" android:exported="true"/>
```

3.3. エリア配信を利用する

エリア配信とは、現在特定のランドマークの範囲内にいるユーザに対してプッシュ通知を行う機能です。

動作条件や、利用のための設定は「3.2. 位置連動配信を利用する」と同じですので、そちらを参照してください。

3.4. ランドマークを利用する

ポスモバ Web のランドマーク管理にて登録したランドマーク情報を本 SDK から取得することが可能です。

3.4.1 ランドマークの取得

ランドマーク情報の取得は、下記のように行います。

```
mManager.getLandmarks(new GetLandmarksCallback() {
    @Override
    public void onLandmarkCallback(Set<Landmark> set) {
        if (set != null) {
            for (Landmark landmark : set) {
                Log.i("Sample",  landmark.toString());
            }
        }
    }
});
```

通信状況により、コールバックの引数に null が返ることがありますので、null チェックを実施してください。

以上で終了です。

3.5. その他 API を利用する

ここまでの説明にあった基本的な実装に使用する API に加え、本 SDK では様々な API を用意しています。
※API の詳細は API リファレンスを参照してください。

3.5.1 属性条件の設定

ポスモバでは、本 SDK 利用アプリケーションから設定した属性条件に従い、プッシュ通知の対象者を限定する機能があります。

属性条件の設定は以下のように行います。

```
// 属性条件の条件 1 に「男性」、条件 2 に「東京都」を設定。
mManager.setUserDatas("男性", "東京都", "", "", "");
```

このように属性条件を設定しておくことで、ポスモバ Web からのプッシュ通知実行の際に、
「男性のみに送る」といった限定が可能となります。

以上で終了です。

3.5.2 ダイアログから起動する Intent の設定

後述の「４．SDK 自動処理」に記載のダイアログから起動する Intent は、以下のように設定します。

```
Intent intent = new Intent();
ComponentName comp = new ComponentName(getPackageName(), MainActivity.class.getName());
intent.setComponent(comp);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
mManager.setDefaultStartIntent(intent);
```

setDefaultStartIntent()の引数に null を指定することで、ダイアログのアプリ起動ボタンを非表示にできます。

以上で終了です。

3.6. プッシュ通知受信時の動作を独自に処理する

本 SDK では、後述の「４．SDK 自動処理」に記載の動作を自動的に実施するようになっています。
ここでは、プッシュ通知受信時に独自の動作を行いたい場合の実装方法を説明します。

3.6.1 カスタム動作用 Service の作成

以下の例のように PushIntentService クラスを継承したクラスを作成し、処理を切り分けます。

```
package com.example.sampleproject;
import jp.co.nsw.baassdk.PushIntentService;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
public class CustomPushIntentService extends PushIntentService {
    @Override
    protected boolean onPushReceived(Intent intent) {
        boolean ret = false;
        Bundle extras = intent.getExtras();
        /*********************************************************************************************
         * プッシュ受信時のカスタム動作
         * ポスモバのプッシュ配信メッセージ画面の「オプション-JSON データ」に以下のように入力し、
         * 動作を切り分けることが可能です。
         * ・アクション１を実行するためのプッシュ配信を実施する場合
         *   {"data" : {"no_display_action" : "action1"}}
         * ・アクション２を実行するためのプッシュ配信を実施する場合
         *   {"data" : {"no_display_action" : "action2"}}
         *********************************************************************************************/
        String action = extras.getString("no_display_action");
        if (action != null) {
            if (action.equals("action1")) {
                // アクション１の処理を実施
                Log.i("Sample", "アクション１を受信しました。");
                // true を返し、SDK 自動処理を停止する
                ret = true;
            } else if (action.equals("action2")) {
                // アクション２の処理を実施
                Log.i("Sample", "アクション２を受信しました。");
                // true を返し、SDK 自動処理を停止する
                ret = true;
            }
        }
        return ret;
    }
}
```

onPushReceived()の戻り値に true を指定すると、SDK 自動処理を停止できるため、ユーザの見た目には見えないプッシュ通知を送信することも可能です。

上記クラスの作成に合わせて、AndroidManifest.xml の PushIntentService の定義に代わり、
下記のように作成したクラスを定義してください。

```
<service android:name="com.example.sampleproject.CustomPushIntentService" android:exported="false">
    <intent-filter>
        <action android:name="nbsdk.intent.action.PUSH"/>
    </intent-filter>
</service>
```

以上で終了です。

3.7. Proguard について

Proguard を使用する場合は、Proguard の除外設定を追加する必要があります。
本 SDK 用の除外設定に加え、google-gson 用の除外設定が必要です。

3.7.1 除外設定

Proguard の除外設定(proguard-project.txt)に、以下の追記を実施してください。

```
-keepattributes *Annotation*
-keepattributes Signature
-keepattributes InnerClasses
-keep class com.google.gson.** {
    <fields>;
    <methods>;
}

-keep class jp.co.nsw.baassdk.** { *; }
```

以上で終了です。

3.8. ビーコンを利用する

ポスモバでは、Bluetooth4.0(BLE,BluetoothSmart)を利用したビーコンの検出が可能です。
「3.1. プッシュ通知を利用する」の手順を実施したうえで、ビーコン検出用の追加の設定を行うことで利用可能となります。

3.8.1 動作条件

本機能は、Android4.3 以降であること及び、端末設定で Bluetooth が ON になっていることが動作条件となります。

3.8.2 AndroidManifest.xml への追記

AndroidManifest.xml に、パーミッションの追加を行います。
<application>タグの前に下記のようにパーミッションを指定してください。

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

続いて、Service の設定を行います。
<application>タグの中に、下記のように設定してください。

```
<service android:name="jp.co.nsw.baassdk.BeaconService" />
```

3.8.3 ビーコンの取得

ポスモバ Web のビーコン管理にて登録したビーコン情報を本 SDK から取得することが可能です。
ビーコン情報の取得は、下記のように行います。

```
mManager.getBeacons(new GetBeaconsCallback() {
    @Override
    public void onBeaconCallback(Set<Beacon> set) {
        if (set != null) {
            for (Beacon  beacon : set) {
                Log.i("Sample",  beacon.toString());
            }
        }
    }
});
```

3.8.4 ビーコンの検出

ビーコン検出は、下記のように行います。

```
        mBeacon = mManager.getBeaconManager();
        // Android4.3 より前の OS バージョンの場合、getBeaconManager()の戻り値が null となる(ビーコン利用不可)
        if(mBeacon != null){
            int ret = mBeacon.startBeaconSearch();
            if(ret == BeaconManager.START_SUCCESS){
                mBeacon.setTransitionCallback(new TransitionCallback() {
                    @Override
                    public void onTransitionExit(final Beacon beacon) {
                        // ビーコン検出範囲から出た
                        Log.v("BEACON", "onTransitionExit: " + beacon);

                    }

                    @Override
                    public void onTransitionEnter(final Beacon beacon) {
                        // ビーコン検出範囲に入った
                        Log.v("BEACON", "onTransitionEnter: " + beacon);

                    }

                    @Override
                    public void onContentFound(final Beacon beacon, Content content) {
                        // 検出したビーコンに関連したコンテンツが見つかった
                        Log.v("BEACON", "onContentFound: " + beacon + " content: " + content);

                    }
                });
                mBeacon.setRangingCallback(new RangingCallback() {
                    @Override
                    public void onRangeBeacon(final Beacon beacon) {
                        // 検出中のビーコンの距離計測
                        Log.v("BEACON", "onRangeBeacon: " + beacon);
                        // ※距離はおおよその目安です。電波状況や遮蔽物の影響により正確度が変化します。
                        Log.v("BEACON", "distance=" + beacon.distance + "[m]");
                    }
                });
            }
        }
```

startBeaconSearch()でビーコンの検出を行った場合は、かならず stopBeaconSearch()でビーコンの検出をストップするようにしてください。
ビーコンの検出は端末のバッテリーを消費するため、必要最小限で検出を実施するよう注意してください。

以上で終了です。

3.9. コンテンツを利用する

ポスモバ Web で登録したコンテンツを本 SDK で取得することが可能です。

3.9.1 コンテンツの取得

コンテンツの取得は、下記のように行います。

```
mButtonContent = (Button)findViewById(R.id.content);
// ボタンクリックでコンテンツを取得する場合
mButtonContent.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        GetContentsFilter gcf = new GetContentsFilter();
        // フィルターに「クーポンのみ」を指定
        gcf.setContentType(GetContentsFilter.TYPE_COUPON);
        // コンテンツの取得要求
        mManager.getContentsRequest(gcf, new GetContentsCallback() {
            @Override
            public void onLocalContentsCallback(GetContentsFilter request, List<Content> contents) {
                // SDK 内部保存情報のコールバック
                Log.v("CONTENT", "onLocalContentsCallback");
                if(contents != null){
                    for (Content content : contents) {
                        Log.v("CONTENT", "Content:" + content);
                    }
                }
            }
            @Override
            public void onDataUpdateNone(GetContentsFilter request) {
                // コンテンツアップデートなし
                Log.v("CONTENT", "onDataUpdateNone");
            }
            @Override
            public void onDataUpdateExist(GetContentsFilter request) {
                // コンテンツアップデートあり
                Log.v("CONTENT", "onDataUpdateExist");
            }
        });
    }
});
```

コンテンツ取得要求では、まず SDK 内部保存コンテンツ情報を onLocalContentsCallback でコールバックし、その後通信による新規・更新コンテンツのチェックが行われます。通信によるチェックで onLocalContentsCallback にて渡されたコンテンツ情報に変更がある場合は、onDataUpdateExist がコールバックされるため、再度コンテンツ取得要求を実施し最新の情報を取得するような実装を行ってください。コンテンツ取得の詳細は API リファレンスをご参照ください。

以上で終了です。

4．SDK 自動処理

4.1. SDK 自動処理の流れ

本 SDK は、プッシュ通知受信時に自動処理を実施します。

※アプリケーション独自のカスタム動作を行う場合は、「3.6. プッシュ通知受信時のカスタム動作」を参照してください。

自動処理は以下の流れで実施されます。

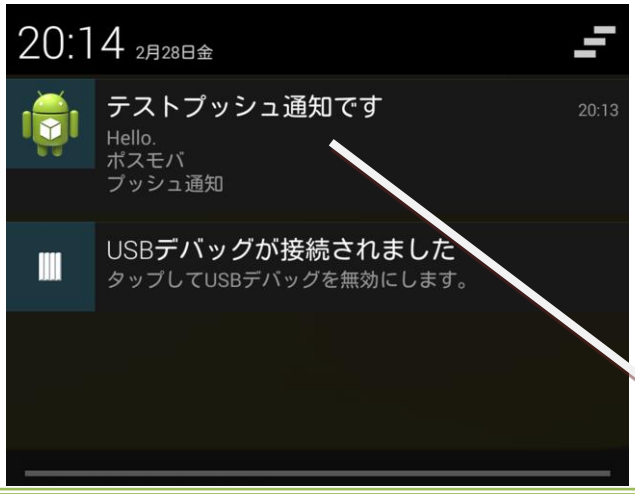
① プッシュ通知受信
↓
② ユーザが Notification をタップ
↓
③ ユーザが URL をタップ(URL 付の場合)
↓
④ ユーザがアプリ起動ボタンをタップ

この自動処理の中で、受信したメッセージに対する「受信」「開封」のステータスがポスモバサーバに送信されます。

① プッシュ通知受信

・「受信」ステータスをポスモバサーバへ送信

・Notification 表示

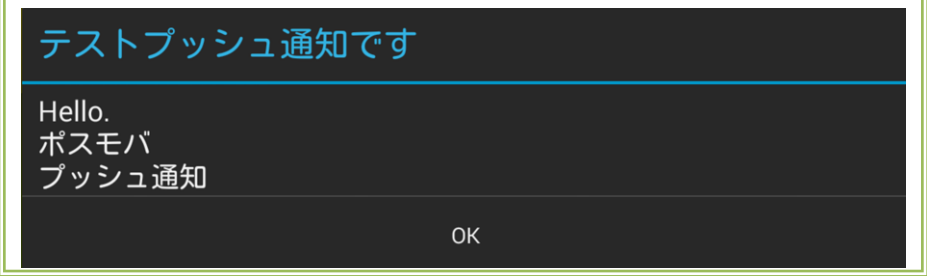


タップすると②の
ダイアログが表示される

② ユーザが Notification をタップ

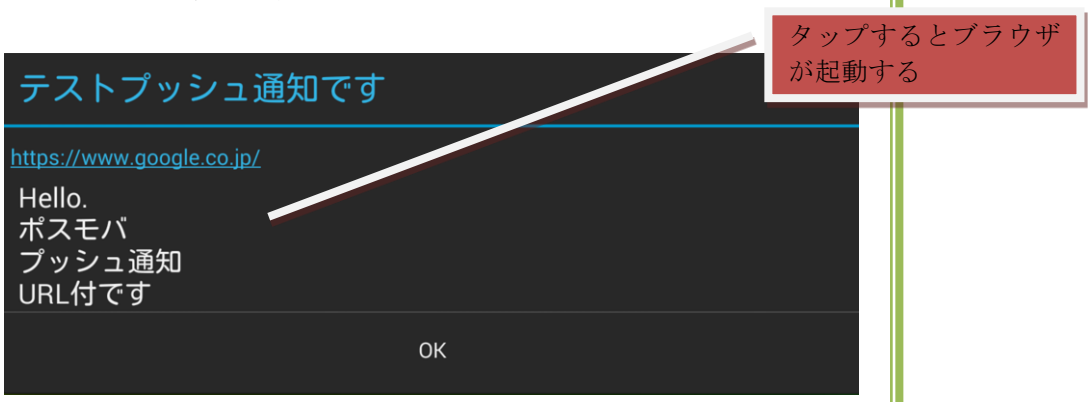
・ダイアログを表示

・「開封」ステータスをポスモバサーバへ送信



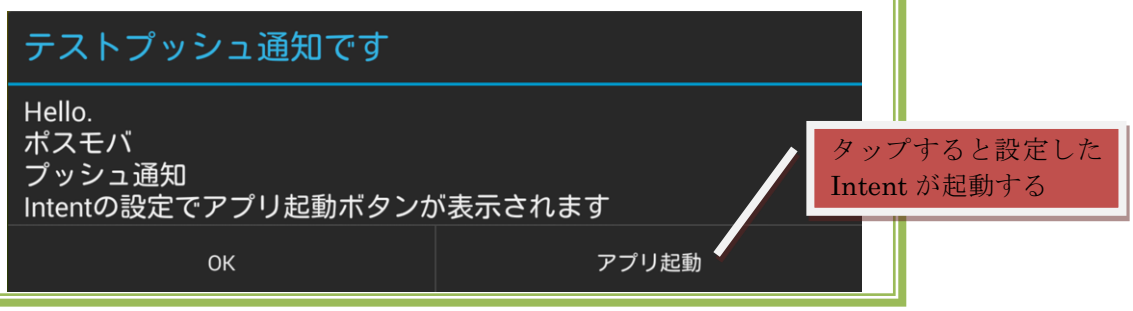
③ ユーザが URL をタップ(URL 付の場合)

・URL へのブラウザ起動



④ ユーザがアプリ起動ボタンをタップ

・設定された Intent を起動
（「3.7. ダイアログから起動する Intent の設定」を参照してください）



※ポスモバサーバへ送信した各種ステータスは、ポスモバ Web の配信分析で確認が可能です。

4.2. Notification やダイアログのカスタマイズ

Notification に表示するアイコンやサウンド、バイブレートは下記メソッドで変更が可能です。
詳細は API リファレンスをご参照ください。

```
void    setNotificationLargeIconID(int id)
void    setNotificationSmallIconID(int id)
void    setNotificationSoundUri(android.net.Uri uri)
void    setNotificationVibratePattern(long[] pattern)
```

ダイアログの見た目は、本 SDK フォルダの「res」フォルダ内のリソースを変更することで、カスタマイズが可能です。

フォルダ	ファイル名	説明
layout	notify_dialog.xml	Android2.3.4 以前用のレイアウト
layout-v11	notify_dialog.xml	Android3.0 以降用のレイアウト
values	strings.xml	文字列定義
	styles.xml	Android2.3.4 以前用のスタイル
values-v11	styles.xml	Android3.0 以降用のスタイル

※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※
レイアウトファイルの id や、スタイル・文字列定義の name は、
SDK から参照しているため変更しないでください。
※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※※

Android3.0 以降用に見た目を変更する必要がない場合は、「layout-v11」「values-v11」フォルダを削除して問題ありません。

5. 付録

5.1. 困ったときは

- Logcat に下記エラーが表示される

・Project Number が不正です。
→「Google Developers Console」(https://console.developers.google.com/project)にて、
正しい Project Number を指定しているかご確認ください。

・SDK ID が不正です。
→「ポスモバ Web」(https://posmob.cs1.nswmps.jp/cnsl/)にて、
正しい SDK ID を指定しているかご確認ください。

・GCM サーバでエラーが発生しました。
→プッシュ通知受信用の RegistrationID 取得に失敗しています。
「Google Developers Console」(https://console.developers.google.com/project)にて、
Google Cloud Messaging for Android が有効になっているかご確認ください。

BaaS SDK(Android 版)利用マニュアル

2014 年 12 月　3.1.0 版

日本システムウエア株式会社

〒150-0036　東京都渋谷区南平台町 2-15
プロダクトソリューション事業本部
ポスモバサポート

mailto: mps-info@gw.nsw.co.jp