BaaS SDK(iOS 版)
利用マニュアル

3.2.0 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	2014/03/10	新規作成
1.1.0	2014/03/19	位置連動配信についての記載を追加
2.0.0	2014/05/09	「3.5 ビーコンを利用する」「3.6 コンテンツを利用する」を追記
2.1.0	2014/09/26	「3.1.2. サンプルコードの実装」にクライアント登録完了についての記載を追加
3.0.0	2014/10/03	推奨開発環境を Xcode6 に更新 「3.2. 位置連動配信を利用する」に Xcode6 でビルドする場合の手順を追加
3.1.0	2014/12/05	改版
3.1.1	2015/1/12	改版
3.1.2	2015/01/21	改版
3.2.0	2015/03/05	改版

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、BaaS SDK(iOS 版)の利用方法を説明します。

1. 動作環境

1.1. 対応 OS

本 SDK は iOS6/iOS7/iOS8 に対応しています。

※ビーコン検出の機能を利用するためには、iOS7/iOS8 が必須です。

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。

外部サイトからダウンロード
- ZipArchive v1.2：https://code.google.com/p/ziparchive/
- FMDB v.2.3：https://github.com/ccgus/fmdb

iOS SDK に収録
- libz
- libsqlite
- CoreLocation.framework

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

1.3. 開発環境

本 SDK の推奨開発環境は Xcode6、ビルドの Base SDK は”iOS8”です。

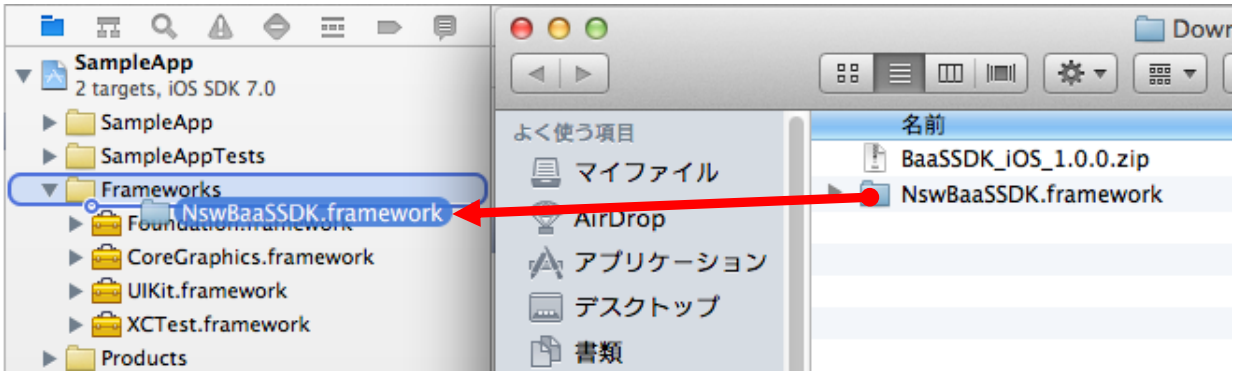
2. セットアップ

2.1. 開発環境のセットアップ

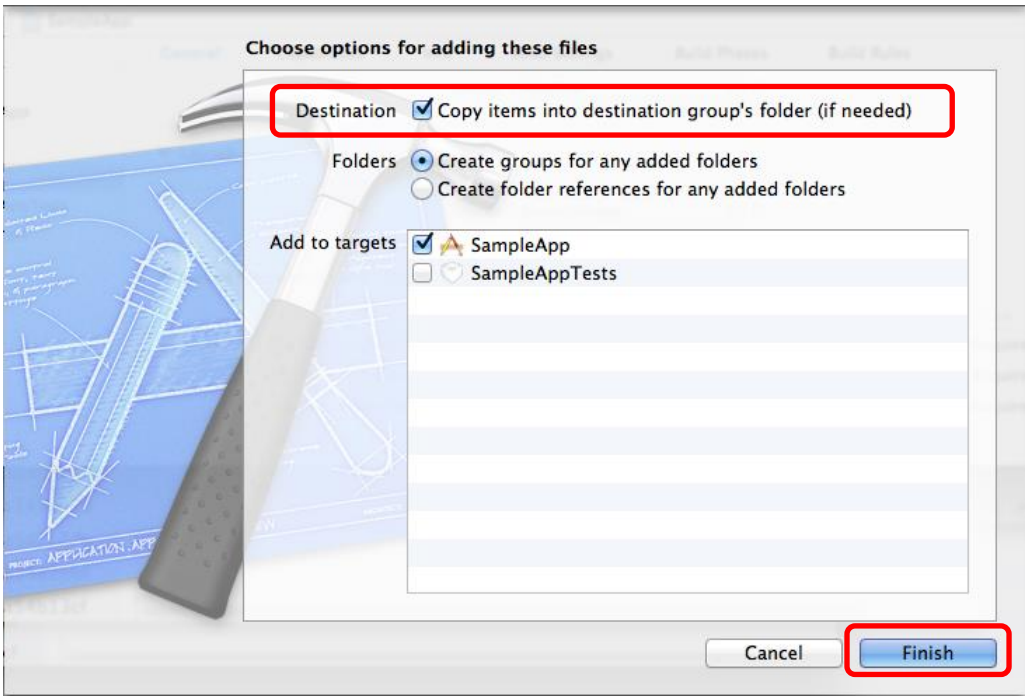
ここでは、開発環境のセットアップ手順を、Xcode5 で作成可能な「Single View Application」を例に説明します。
以下の手順を行うことで、BaaS SDK を利用できるようになります。

1.　SDK をプロジェクトに組み込む

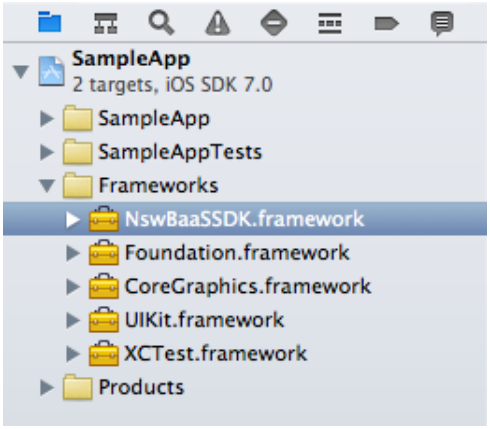
ダウンロードした「BaaSSDK_iOS_x.x.x.zip」を解凍し、「NswBaaSSDK.framework」をプロジェクトの「Frameworks」フォルダにドラッグ&ドロップしてください。



必要であれば、ファイルをコピーするオプションを有効にして Finish をクリックしてください。



以下のように、「Frameworks」フォルダ内に「NswBaasSDK.framework」が追加されます。

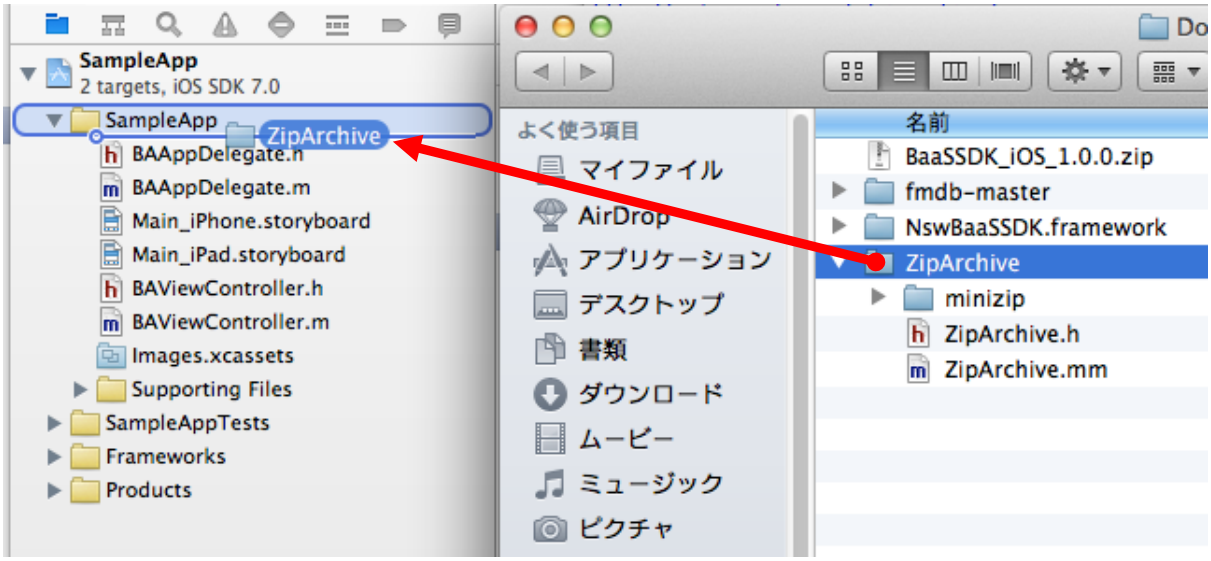


2.　必要なライブラリを追加する

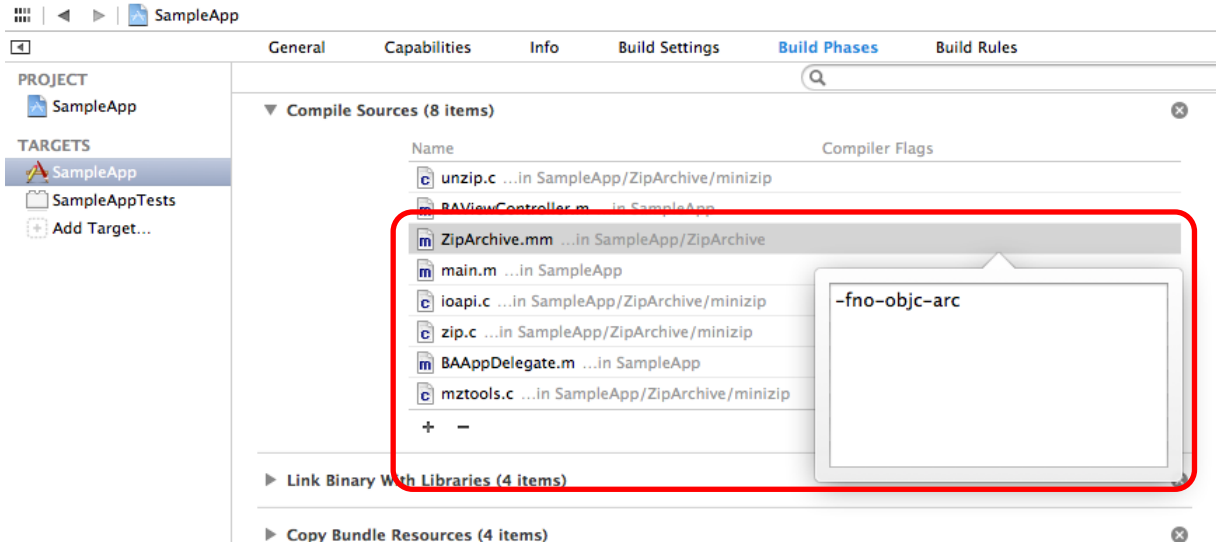
外部ライブラリをダウンロードし、プロジェクトに取り込みます。

- ZipArchive：http://code.google.com/p/ziparchive/

ダウンロード、解凍してできた「ZipArchive」フォルダをプロジェクトの「SampleApp」フォルダにドラッグ&ドロップしてください。

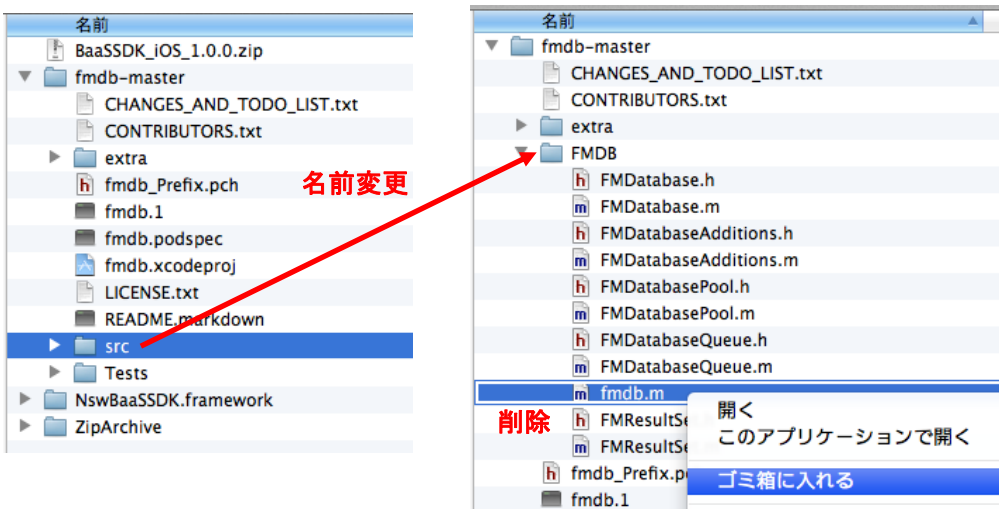


ZipArchive は ARC（Automatic Reference Counting）が利用できないため、ARC 対象から外します。
TARGETS – SampleApp – Build Phases – Compile Sources の項目で、「ZipArchive.mm」をダブルクリックして表示されるポップアップに「-fno-objc-arc」と入力してください。

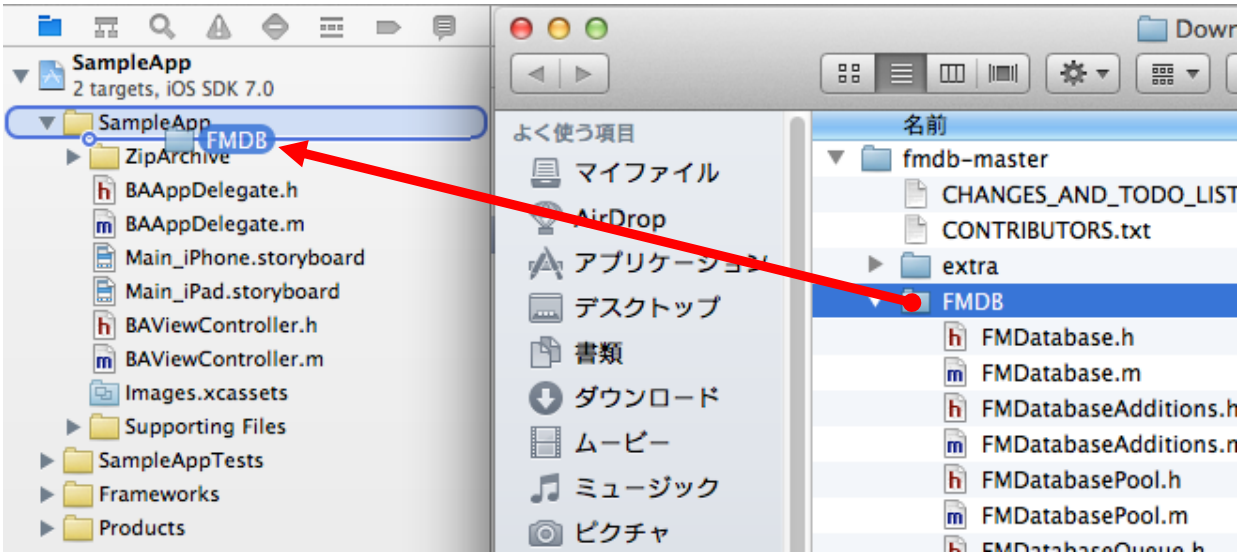


- FMDB：https://github.com/ccgus/fmdb

「fmdb-master」フォルダにある「src」フォルダを「FMDB」に名前変更し、「FMDB」フォルダ内の「fmdb.m」ファイルを削除してください。



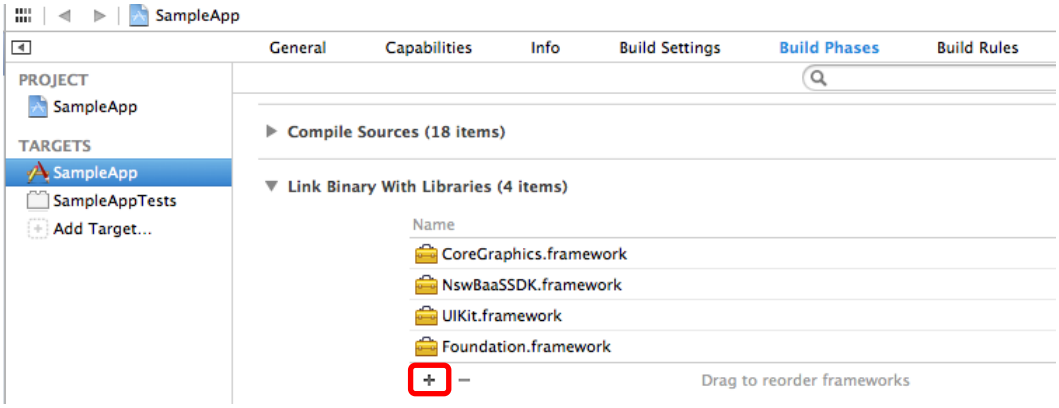
「FMDB」フォルダをプロジェクトの「SampleApp」フォルダにドラッグ&ドロップしてください。



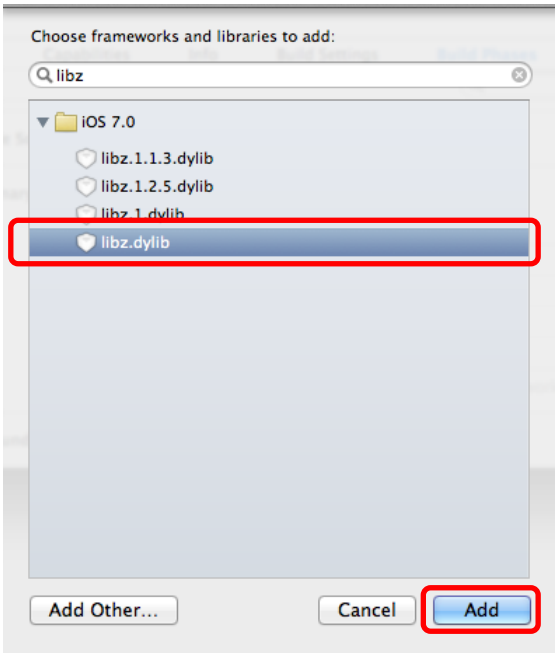
さらに、以下のライブラリをプロジェクトに追加してください。

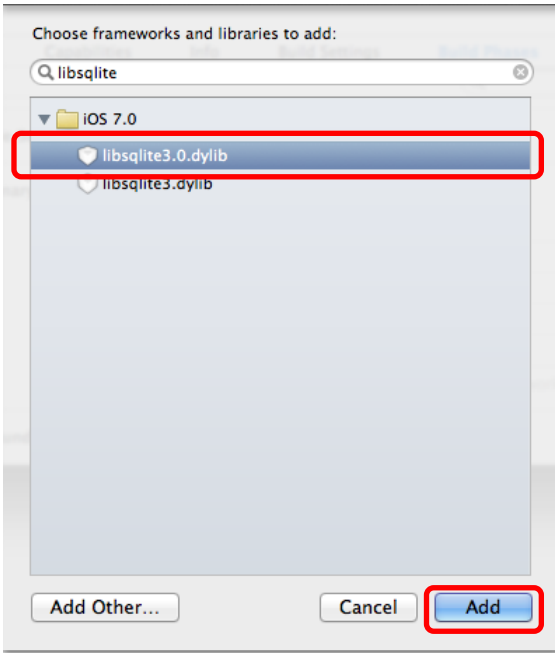
- libz.dylib (ZipArchive に必要)
- libsqlite.3.0.dylib (FMDB に必要)

TARGETS – SampleApp – Build Phases – Link Binary With Libraries の項目で、「＋」ボタンをクリックしてください。



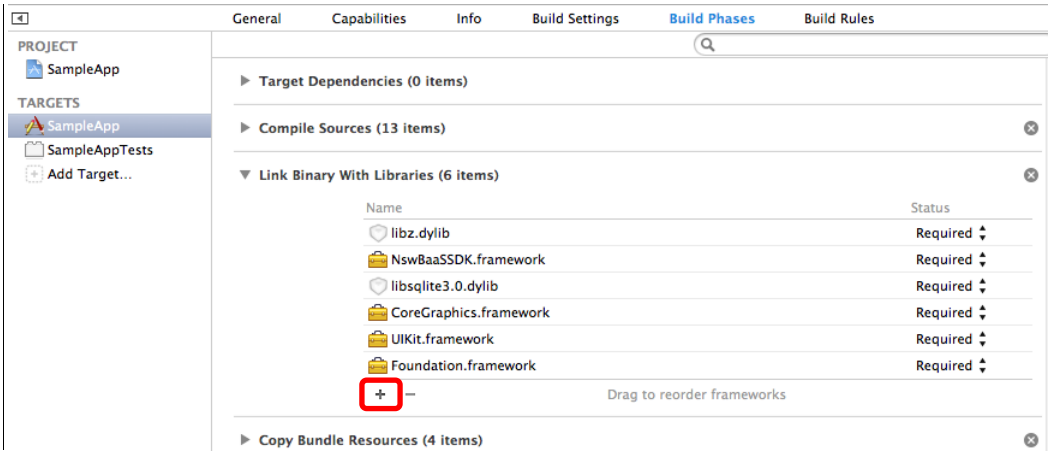
表示された項目から、「libz.dylib」、「libsqlite.3.0.dylib」を選択して Add をクリックしてください。



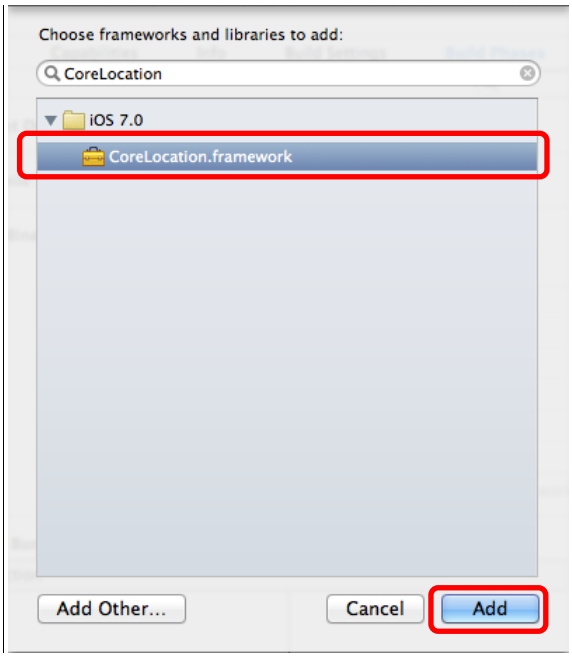


位置連動配信を利用する場合、位置情報の取得を行うために「CoreLocation.framework」を追加します。

TARGETS – SampleApp – Build Phases – Link Binary With Libraries の項目で、「＋」ボタンをクリックしてください。

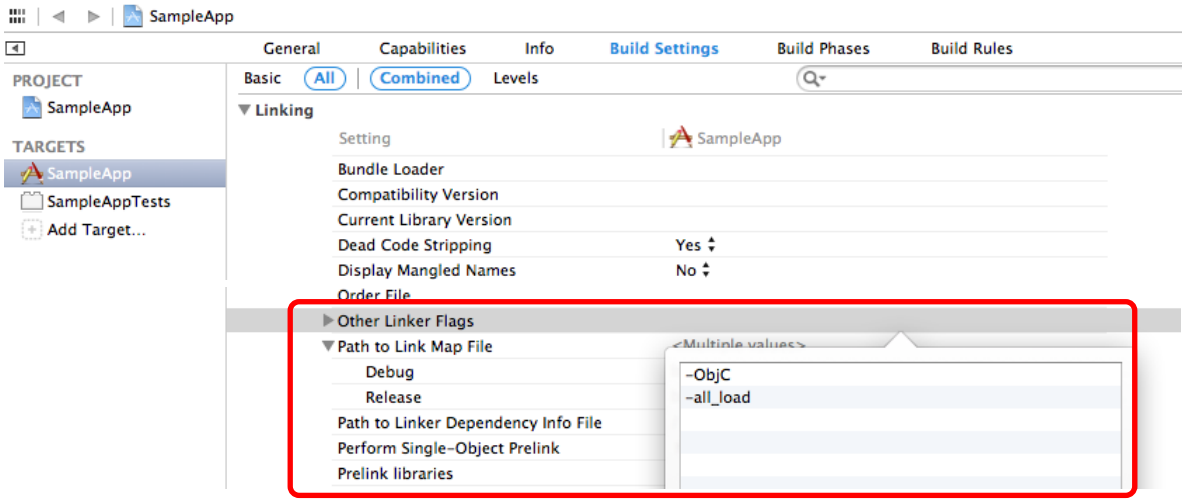


表示された項目から、「CoreLocation.framework」を選択して Add をクリックしてください。



3. リンカーフラグを設定する

TARGETS – SampleApp – Build Settings – Linking の項目で、「Other Linker Flags」をダブルクリックして表示されるポップアップに「-ObjC」と「-all_load」を追加してください。



以上で、開発環境のセットアップは終了です。BaaS SDK を使用する準備ができました。

3．利用方法

3.1. プッシュ通知を利用する

ポスモバ Web から通常配信を行った場合に、メッセージをプッシュ通知として受信するための方法を説明します。

3.1.1. 動作条件

iOS6/7 の設定には、プッシュ通知を通知センターに表示を許可するかどうかの設定があります。各バージョンで以下の項目で確認できます。

iOS6： 設定→通知→各アプリ→通知センター
iOS7： 設定→通知センター→各アプリ→通知センターに表示

この設定は通知センターに表示するかどうかの設定のため、この設定に依らず、アプリはプッシュ通知を受信することができます。

通知センターへの表示が許可されていない場合、ポスモバ Web から通常配信を行ったとしても通知センターには表示されません。ただし、アプリが最前面にいる状態のときは、プッシュ通知を受信したことをアプリが知ることができるためアプリの処理を実行することが可能です。

3.1.2. サンプルコードの実装

ここでは、BaaS SDK を利用してプッシュ通知を受信するために必要な基本的な実装を説明します。
記載しているコード例を全て記載したサンプルコードも同梱していますので、ご利用ください。

1.　ポスモバ Web にて発行された SDK-ID を使って初期設定を行います。

```
#define SDK_ID @" xxxxxx.aaa.bbb.ccc.ddd"  //ポスモバ Web にて発行された SDK-ID に置き換えてください

//プロパティを定義
@interface AppDelegate() <NBNswBaaSManagerDelegate>
@property NBNswBaaSManager *manager;
@end

- (BOOL)application:(UIApplication *)application
 didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    //NSW BaaS Manager の初期設定
    if (!self.manager) {
        self.manager = [NBNswBaaSManager sharedManager];
        self.manager.delegate = self;
        [self.manager initializeWithSdkId:SDK_ID];
    }

    return YES;
}
```

アプリ初回起動時に「"アプリ名"はあなたにプッシュ通知を送信します。よろしいですか？」というアラートが表示されるので、「OK」をタップすると、プッシュ通知を受信できるようになります。
このアラートの表示は iOS 標準の動作のため、非表示にすることはできません。

2.　UIApplicationDelegate プロトコルから API をコールします。

UIApplicationDelegate プロトコルの各メソッドとコールすべき API の対応は以下のようになります。

UIApplicationDelegate プロトコル	NBNswBaaSManager クラス
application:didFinishLaunchingWithOptions:	didFinishLaunchingWithOptions:
application:didRegisterForRemoteNotificationsWithDeviceToken:	didRegisterForRemoteNotificationsWithDeviceToken:
application:didFailToRegisterForRemoteNotificationsWithError:	didFailToRegisterForRemoteNotificationsWithError:
application:didReceiveRemoteNotification:	didReceiveRemoteNotification:
application:didReceiveRemoteNotification:fetchCompletionHandler:	didReceiveRemoteNotification:
application:didReceiveLocalNotification:	didReceiveLocalNotification:
applicationDidEnterBackground:	pause:
applicationWillEnterForeground:	resume:

各メソッドでの API コールの例は以下のようになります。

```
- (BOOL)application:(UIApplication *)application
 didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    /*  省略：NSW BaaS Manager の初期設定  */

    //起動時のオプションを SDK に送る
    [self.manager didFinishLaunchingWithOptions:launchOptions];

    return YES;
}
```

```
- (void)applicationWillResignActive:(UIApplication *)application
{
    //SDK の処理を中断する
    [self.manager pause];
}
```

```
- (void)applicationDidBecomeActive:(UIApplication *)application
{
    //SDK の処理を再開する
    [self.manager resume];
}
```

```
- (void)application:(UIApplication *)application
 didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)devToken
{
    //取得したデバイストークンを SDK に送る
    [self.manager didRegisterForRemoteNotificationsWithDeviceToken:devToken];
}
```

```
- (void)application:(UIApplication *)application
 didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
{
    //エラー情報を SDK に送る
    [self.manager didFailToRegisterForRemoteNotificationsWithError:error];
}
```

```
- (void)application:(UIApplication *)application
 didReceiveRemoteNotification:(NSDictionary *)userInfo
{
    //受信した RemoteNotification の情報を SDK に送る
    [self.manager didReceiveRemoteNotification:userInfo];
}
```

```
- (void)application:(UIApplication *)application
 didReceiveLocalNotification:(UILocalNotification *)notification
{
    //受信した LocalNotification の情報を SDK に送る
    [self.manager didReceiveLocalNotification:notification];
}
```

```
- (void)application:(UIApplication *)application
 didReceiveLocalNotification:(UILocalNotification *)notification
{
    //受信した LocalNotification の情報を SDK に送る
    [self.manager didReceiveLocalNotification:notification];
}
```

iOS7 で追加された application:didReceiveRemoteNotification:fetchCompletionHandler:メソッドを使用する場合は、application:didReceiveRemoteNotification:メソッドと同様に NBNswBaaSManager クラスの didReceiveRemoteNotification:メソッドをコールしてください。

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
fetchCompletionHandler:(void (^)(UIBackgroundFetchResult result))completionHandler
{
    //受信した RemoteNotification の情報を SDK に送る
    [self.manager didReceiveRemoteNotification:userInfo];
    completionHandler(UIBackgroundFetchResultNoData);
}
```

アプリが最前面にいない状態でもこのメソッドはコールされますが、その場合 BaaS SDK はバックグラウンドでは処理を行いません。

3. クライアント登録を判定します。

[NBNswBaaSManager initializeWithSdkId:]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[NBNswBaaSManager isRegistered]と以下のコールバック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。

```
- (void)didRegister
{
    NSLog(@"クライアント登録成功");
}

- (void)didFailToRegister:(NBRegisterError)reason
{
    NSLog(@"クライアント登録失敗");
    //クライアント登録に失敗した場合本コールバックが呼ばれる。
    //自動でリトライするため、登録が成功した段階で didRegister:がコールされる。
}
```

4. プッシュ通知受信時の処理を実装します。

プッシュ通知を受信すると、NBNswBaaSManagerDelegate プロトコルの didReceivedPushMessage:メソッドがコールされます。

具体的に呼び出されるのは以下の場合です。
- アプリが最前面にいる状態でプッシュ通知を受信した場合
- 通知センターにある通知をユーザーがタップした場合

受信したメッセージ本文をアラートで表示する例は以下のようになります。

```
/**
 * Push 配信メッセージ受信時に呼び出される。
 */
- (void)didReceivedPushMessage:(NBPushMessage *)message
{
    //例：メッセージ本文をアラートで表示する
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:nil
                                                    message:message.body
                                                   delegate:nil
                                          cancelButtonTitle:@"OK"
                                          otherButtonTitles:nil];
    [alert show];
}
```

以上でプッシュ通知を受信できるようになりました。

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」、「位置連動配信を使用したくない」ユーザー向けに、それぞれの有効／無効を切り替える機能があります。

本 SDK のデフォルトでは「プッシュ通知を受信する」「位置連動配信を使用する」にて動作するようになっているため、ユーザーの切り替えではなく常に位置連動配信を使用したくないような場合は、初期化処理(initializeWithSdkId)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。

```
//Push 通知を有効、位置連動配信を無効にする。
[self.manager setPermissionPush:YES location:NO];
```

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、ユーザー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。

3.2.1. 動作条件

本機能は「3.1.3. SDK パーミッションの設定」にて、「位置連動配信を使用する」を設定した場合のみ動作します。

アプリが位置情報を取得するために、端末設定で以下の設定がされていることが動作条件となります。
- 位置情報サービスが ON になっていること
- Wi-Fi が ON になっていること（アクセスポイントに接続している必要はありません）
- アプリの位置情報の利用が許可されていること

また、iOS7 の場合、アプリがバックグラウンドで動作するために、アプリがホームボタンをダブルクリックして表示されるマルチタスク画面に残っていることと、端末設定で以下の設定がされていることが動作条件となります。
- アプリのバックグラウンド更新が許可されていること

各設定は以下の項目で確認できます。
位置情報サービス：設定→プライバシー→位置連動サービス
Wi-Fi：設定→Wi-Fi
アプリの位置情報の利用許可: 設定→プライバシー→位置連動サービス→各アプリ
アプリのバックグラウンド更新の許可: 設定→一般→App のバックグラウンド更新→各アプリ

3.2.2. フレームワークの追加

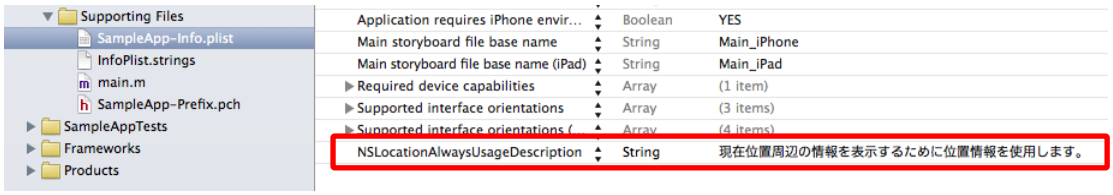
位置連動配信を利用する場合、位置情報の取得を行うために「CoreLocation.framework」を追加します。
詳細は「2.1. 開発環境のセットアップ」の手順 2 を参照してください。

以上で位置連動配信を受信できるようになりました。

位置連動配信を受信した場合の動作は、プッシュ通知受信時と同様に NBNswBaaSManagerDelegate プロトコルの didReceivedPushMessage:メソッドがコールされます。

3.2.3. 位置情報を取得する説明の記載

プロジェクトを Xcode6 でビルドする場合には、Info.plist に NSLocationAlwaysUsageDescription キーで位置情報を取得するための説明を記載してください。
説明の内容にはアプリごとに用途にあったものを記載してください。
※この記載が無いと、アプリは位置情報を取得できず位置連動配信が動作しません。



アプリでは位置情報の取得時に以下のように表示されます。

3.3. 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。

3.3.1. ランドマークの取得

ランドマーク情報を取得するには、NBNswBaaSManager クラスの requestLandmarks メソッドをコールします。非同期で通信を行い、ランドマークの取得終了後に NBNswBaaSManagerDelegate プロトコルの didFinishedRequestLandmark:メソッドがコールされます。

```
- (IBAction)clickedButton:(id)sender {
    //ボタンがタップされたらランドマークを取得する。
    [self.manager requestLandmarks];
}

/**
 * ランドマーク取得(requstLandmark)終了時に呼び出される。
 */
- (void)didFinishedRequestLandmark:(NSArray *)landmarks
{
    NSLog(@"ランドマークの取得が終了しました");
}
```

3.4. その他 API を利用する

3.4.1. 利用者情報の設定

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。

```
//利用者情報 1 にシステムのロケールを、利用者情報 2 にシステムのタイムゾーンを設定する。
[self.manager setUserData:@"男性"]
                    data2:@"東京都"
                    data3:nil
                    data4:nil
                    data5:nil];
```

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。

3.4.2. SDK 自動処理の設定

本 SDK では、「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。
この自動処理を実行させたくない場合は、初期化処理（initializeWithSdkId)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。

```
//Push 通知受信時、SDK に処理をさせない。（デフォルトは YES）
[self.manager setPushMessageAutoPlay:NO];
```

3.5. ビーコンを利用する

ポスモバでは、iBeacon 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。

3.5.1. 動作条件

本機能は、iOS7 以降であること及び、端末設定で Bluetooth と位置情報サービスが ON になっていることが動作条件となります。

3.5.2. 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。

ビーコン情報を取得するには、NBNswBaaSManager クラスの requestBeacons メソッドをコールします。
非同期で通信を行い、ビーコンの取得終了後に NBNswBaaSManagerDelegate プロトコルの didFinishRequestBeacons:メソッドがコールされます。

```
- (IBAction)clickedButton:(id)sender {
    //ボタンがタップされたらビーコンを取得する。
    [self.manager requestBeacons];
}

/**
 * ビーコン取得(requstBeacons)終了時に呼び出される。
 */
- (void)didFinishRequestBeacons:(NSArray *)beacons
{
    NSLog(@"ビーコンの取得が終了しました");
}
```

3.5.3. ビーコンの検出

ビーコン検出は、下記のように行います。

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    //ポスモバ Web で登録したビーコンの UUID をご使用ください
    NSUUID *uuid1 = [[NSUUID alloc] initWithUUIDString:@"00000000-0000-0000-0000-000000000001"];

    //CLBeaconRegion の identifier はアプリで一意になるように決めてください
    self.region1 = [[CLBeaconRegion alloc] initWithProximityUUID:uuid1 identifier:@"region1"];
    self.beaconManager = [[NBNswBaaSManager sharedManager] sharedBeaconManager];
    if (!self.beaconManager) {
        NSLog(@"ビーコン検出機能が使用できない端末です");
    } else {
        self.beaconManager.delegate = self;
    }
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];

    NBBeaconStartResult result = [self.beaconManager startBeaconSearchInRegion:self.region1];
    if (result == kNBBeaconStartResultSuccess) {
        [self.beaconManager startMonitoringForBeacon:self.region1];
    } else {
        NSLog(@"ビーコン検出機能の開始に失敗しました");
    }
}

- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];

    [self.beaconManager stopMonitoringForBeacon:self.region1];
    [self.beaconManager stopBeaconSearchInRegion:self.region1];
}
```

```
/** ビーコン検出範囲内に入ったときに呼び出される */
- (void)didEnterBeaconRegion:(CLBeaconRegion *)region beacon:(NBBeacon *)beacon
{
    NSLog(@"ビーコン検出範囲に入った %@", beacon);
}

/** ビーコン検出範囲内から出たときに呼び出される */
- (void)didExitBeaconRegion:(CLBeaconRegion *)region beacon:(NBBeacon *)beacon
{
    NSLog(@"ビーコン検出範囲から出た %@", beacon);
}

/** ビーコン連動コンテンツが見つかったときに呼び出される */
- (void)didFindContent:(NBBeacon *)beacon content:(NBContent *)content
{
    NSLog(@"検出したビーコンに関連したコンテンツが見つかった %@", content);
}

/** 検出しているビーコンに対して頻繁(約１秒ごと)に呼び出される。 */
- (void)didRangeBeacons:(NSArray *)beacons inRegion:(CLBeaconRegion *)region
{
    NSLog(@"検出中のビーコン %@ %@", region.identifier, beacons);
}
```

ビーコン検出機能には、周囲のビーコンを検出するもの(レンジング)と、ビーコン検出範囲への出入りを検出するもの(モニタリング)を用意しています。
ビーコン検出機能を開始するためのメソッドの引数 CLBeaconRegion の詳細については Apple の公式ドキュメント(https://developer.apple.com/jp/devcenter/ios/library/documentation/LocationAwarenessPG.pdf)を参照ください。

モニタリングを開始した後、停止せずにアプリがバックグラウンド状態に移行すると、そのままビーコン検出範囲への出入り検出を続け、適宜 didEnterBeaconRegion:と didExitBeaconRegion:が呼び出され、ビーコン連動コンテンツがあった場合は通知センターに通知が表示されます。このときの通知センターに表示される文言はポスモバ Web で設定した「受信時メッセージ本文」です。
通知センターにある通知をユーザーがタップした場合、「3.1 プッシュ通知を利用する」と同様に didReceivedPushMessage:メソッドがコールされます。

レンジング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。

3.6. 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。

3.6.1. 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。

```
- (IBAction)clickedButton:(id)sender {
    [NBNswBaaSManager sharedManager].contentDelegate = self;

    //ボタンがタップされたらコンテンツを取得する。
    NBContentsRequest *request = [NBContentsRequest new];
    //「クーポンのみ」を指定
    request.contentType = kNBContentTypeFlagCoupon;
    // コンテンツの取得要求
    [self.manager requestContents:request];
}

/** コンテンツ取得(requestContents)終了時に呼び出される。*/
- (void)didLoadLocalContents:(NBContentsRequest *)request contents:(NSArray *)contents
{
    // SDK 内部保存情報のコールバック
    NSLog(@"取得済みコンテンツ %@", contents);
}

/** コンテンツ情報変更ありの時に呼び出される。 */
- (void)didReceiveUpdateExist:(NBContentsRequest *)request
{
    NSLog(@"コンテンツアップデートあり");
}

/** コンテンツ情報変更なしの時に呼び出される。 */
- (void)didReceiveNoUpdateExist:(NBContentsRequest *)request
{
    NSLog(@"コンテンツアップデートなし");
}
```

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 didLoadLocalContents: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 didLoadLocalContents:にて渡されたコンテンツ情報に変更がある場合は、didReceiveUpdateExist: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. 処理内容

前述の「3.4. SDK 自動処理の設定」で SDK 自動処理が有効の場合、本 SDK はプッシュ通知受信時に下記の処理を実施します。
SDK 自動処理を有効にすることによって、ポスモバ Web の配信分析画面でメッセージの受信数や開封数を確認できるようになります。

プッシュ通知はアプリの状態によって通知センターに表示される場合と、表示されない場合があります。本 SDK の自動処理ではこの動作の違いを吸収し、プッシュ通知受信時には通知センターに通知を表示します。

4.1.1. 通知センターにある通知をユーザーがタップした場合

通知センターにある通知をユーザーがタップした場合、以下の処理を行います。(アプリが最前面にいる状態でも、バックグラウンドにいる状態でも同様の処理です。)

1. ポスモバサーバにメッセージの開封通知を送信します。
2. プッシュ通知に連携 URL が設定されている場合、外部ブラウザを起動し、設定された URL の Web ページを表示します。
3. NBNswBaaSManagerDelegate プロトコルの didReceivedPushMessage:メソッドをコールします。

4.1.2. アプリが最前面にいる状態でプッシュ通知を受信した場合

アプリが最前面にいる状態でプッシュ通知を受信した場合は、即時 NBNswBaaSManagerDelegate プロトコルの didReceivedPushMessage:メソッドをコールします。このとき、ユーザーが能動的にメッセージを開封したわけではないため、開封通知は送信しません。

4.1.3. 位置連動配信を受信した場合

位置連動配信を受信した場合、ポスモバサーバにメッセージの受信通知を送信します。
その後の SDK 自動処理は「4.1.1. 通知センターにある通知をユーザーがタップした場合」および「4.1.2.アプリが最前面にいる状態でプッシュ通知を受信した場合」と同様です。

5. 付録

5.1. 困ったときは

5.1.1. NSData クラスの detectEncoding:メソッドが見つからずにアプリが強制終了する

「2.1. 開発環境のセットアップ」の手順 3 でリンカーフラグを設定していない場合に発生します。ビルド設定を確認してください。

5.1.2. プッシュ通知が受信できない

プッシュ通知が受信できない場合、よくある原因がいくつかあります。

1.　プッシュ用証明書とアプリのビルド設定が一致していない

プッシュ用証明書には開発環境用と製品環境用があります。開発環境用は Debug ビルド、製品環境用は Release ビルドと対応しており、ポスモバに登録したプッシュ用証明書とアプリのビルド設定が一致していない場合、プッシュ通知を受信することができません。

2.　端末の設定でプッシュ通知の表示を許可していない

設定 – 通知センターで、アプリに対してプッシュ通知の表示が許可されているかどうかご確認ください。初回起動時のプッシュ通知の送信許可の確認アラートで「許可しない」をタップした場合にプッシュ通知の表示をしない設定となります。

5.1.3. ビーコン検出の delegate メソッドがコールされない

機種と OS のバージョンの組み合わせによって、ビーコン検出の動作が不安定になる場合があります。
iPhone4s＋iOS7.0 の組み合わせの場合、バックグラウンドでモニタリングの didExitBeaconRegion:beacon メソッドが呼ばれるまでに 4 分程度かかる場合があります。

アプリがフォアグラウンドにいる場合は、モニタリングとレンジングを同時にスタートすることで、モニタリングの delegate メソッドが呼ばれるまでのタイムラグを減らすことができます。

BaaS SDK(iOS 版)利用マニュアル

2015 年 3 月　3.2.0 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