3bitter SWAMP SDK 導入ガイド基本編

BLT – F.O.X Edition

Revision 1.3.0

2016/8/23

改訂履歴

Revision	更新日	更新内容サマリー	更新者
1.0.0	2016/03/06	初版作成	上田
1.1.0	2016/03/16	AdSupport.Framework を削除	上田
1.2.0	2016/04/21	3-5 及び 4-2 を「アプリ使用中のみの許可」主体に改訂	上田
1.3.0	2016/08/23	F.O.X.システムイベント計測連携について追記	上田

１ はじめに

1. 本 SDK を使用してできること

本 SDK（Software Development Kit）は、使用するアプリケーションに対し、3bitter 株式会社が提供する専用ビーコン端末の制御使用、ビーコン端末で決定されるサービス用ビーコン領域或いは 3bitter 株式会社の運営管理下において利用可能な領域の使用、及び、ビーコン領域判定の、3bitter 社が提供するビーコンネットワークサービスへの記録送信を可能とします。これら機能により、ビーコン領域限定コンテンツを限定解除したり、限定コンテンツの利用状況を把握して、プロモーションに活かすといった応用が考えられます。

2. 本ガイド活用の前提事項

本ガイドの記述には、

- Xcode を使用した iOS 向けアプリケーション開発の基礎知識
- iOS での位置情報処理フレームワーク（CoreLocation）についての基礎知識

を前提とした箇所が含まれますので、必要に応じてこれらの情報を確認し、理解を深めてください。

また、SDK のクラス等の詳細仕様の確認は、「TbBTSDK 仕様書（BLT エディション）／SWAMP SD 仕様書（BLT エディション）」を参照ください。

※最新版での改訂箇所には黄色マーキングを（大まかに）しています。前版をお持ちの場合は、差分確認箇所としてください。

3. 本ガイドの目標範囲

本ガイドでは

- 3bitter 提供の SWAMP SDK を組み込んだアプリケーションをビルドできる

- 3bitter 提供のビーコンの検出をアプリから制御できる
- 3bitter 提供のビーコンを検出したタイミングで、事前に関連付けしてあるコンテンツを表示させることができる

ことを目標として、導入及び開発に必要な事項を扱います。

（「TbBLTBasicSample」サンプルアプリが対応するレベルのアプリケーションとなります）

※　以上を越えた使い方、企画・設計のケーススタディ、Tips 等、については、別途「応用版ガイド」やサポート資料での扱いとさせていただきます。

4. 対象のバージョン

最新の SDK バージョンは 1.1.3 です

後日 SDK がアップデートされた場合、本バージョンのガイドの一部記載事項が無効となる可能性がありますので、ご了承下さい。

5. 必要要件

本 SDK は、iOS7.0 以上の iPhone 及び iPod touch 端末の動作環境を対象としています。

※　シミュレータについては、インストール＆許可確認までは動作しますが、ビーコンのモニタリングはできないので、動作確認はできません。

6. 制約事項

SDK には Swift 開発言語でコーディングされたアプリケーションから SDK のクラスを呼び出せるよう「TbBTSDK-Bridging-Header.h」ファイルを同梱していますが、本ガイドでは現在、Swift 言語からの SDK 使用実装ではなく Objective-C での使用実装例のみ記載しています。

また、本 SDK と他の位置情報処理が併用できるよう全体として適切な挙動とするにはアプリケーション側に任せる処理が多めとなっていますが、実装コード例中に含まれているビーコン処理に直接関わる部分でない補助的な処理（フェイルリカバリ用ステップ等）については基本的に説明を省略しています。ご了承ください。

7. SDK 入手方法

現在は直接のお渡しとさせていただいています。

8. その他

Unity 使用のアプリの場合には、本 SDK の Unity 向けプラグインを別途用意しております。

9. 問い合わせ

con@3bitter.com までお問い合わせください。

２ ユーザー体験と SDK の役割概要（基本編）

２-１ [ステップ 1] 位置情報サービス使用許可への同意

本処理は、ステップ 2 の処理を実行する前に最低一度実行される必要があります。また、ユーザーがその後設定を変えなければ、一度のみの実行で済みます。

[処理の流れ]
アプリケーション：位置情報サービス使用メリットの説明 → SDK によるビーコン連動サービス用の準備 -> ステップ 2 へ。

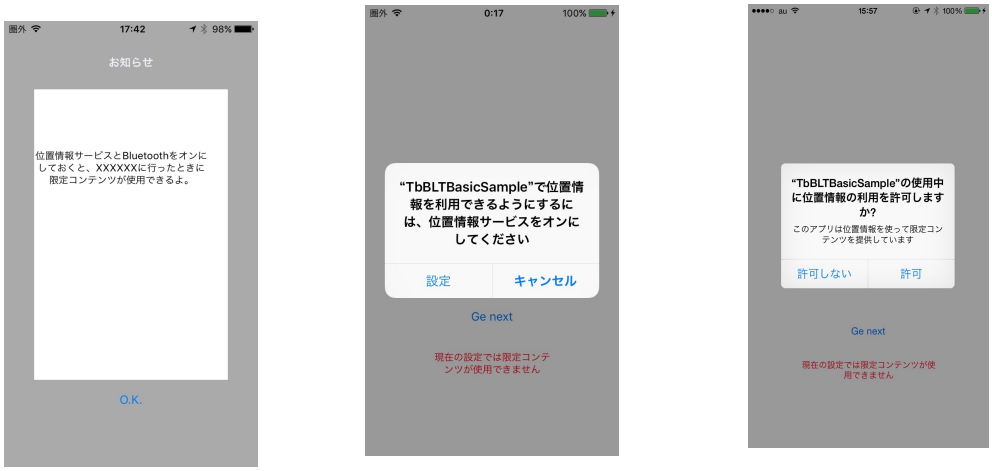

図 1 ユーザー許可の確認

２-２ [ステップ 2] ビーコン領域検知からの、特定コンテンツ連動へのアクション（ユーザーへの限定コンテンツ提供）

3bitter ビーコンは、(簡易的に表現すると) キーコード値にて管理されています。このキーコードに対し、特殊コンテンツを事前に関連付けしておくことで、ユーザー端末が該当のビーコンを発見した場合に限り、該当のコンテンツを表示することができます。

図 2 ユーザー端末での限定コンテンツの表示

[処理の流れ]

アプリ: 位置情報サービス使用の準備 → アプリ：3bitter ビーコンの検出指示 → OS :ビーコンの検知 → アプリ：ビーコンキーコードの確認指示 → SDK → ビーコンキーコードの応答とトラッキング送信 → アプリ:応答されたキーコードでのコンテンツ取得

３導入の流れ

３-１[構成の確認]

SDK は大きく、以下の 3 種類のファイル群から成ります

- ヘッダファイル
- ライブラリ
- サポートファイル

圧縮ファイルを解凍し、以下のファイルが含まれているか、内容を確認して下さい。

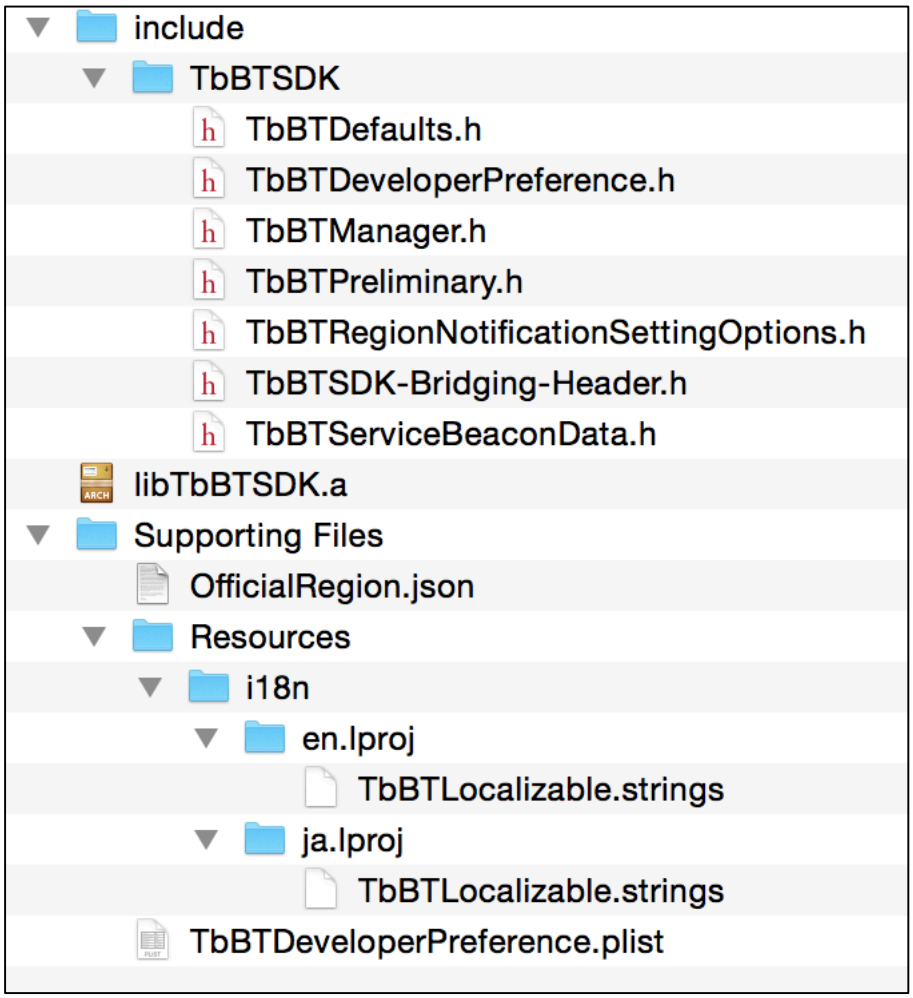

図 2 SDK ファイル構成

3-2 導入ステップ 1 - [Deployment Target]の確認

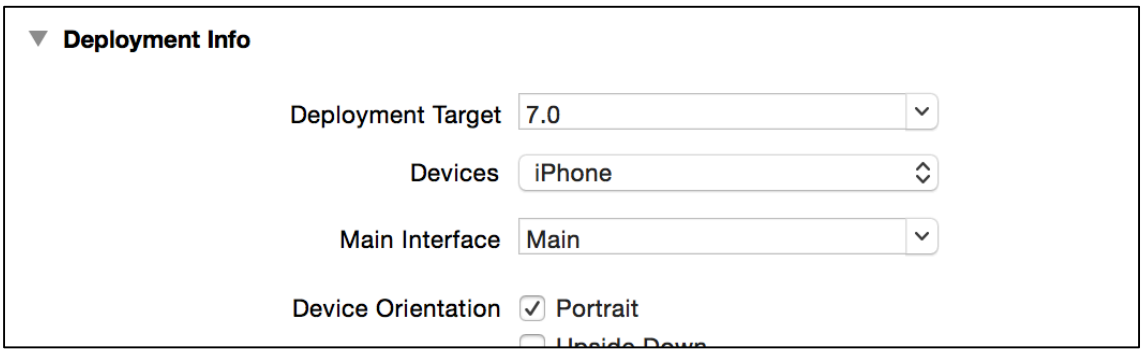

図 3　ターゲット指定

Xcode のプロジェクトで、「Project」を選択した場合の［Info］－［Deployment Target］及び、「Targets」を選択した場合の、「General」-「Deployment Info」

で Target が 7.0 以上(の任意値)になっていることを確認します

３-３導入ステップ２- フレームワークとライブラリの追加

以下のフレームワークを使用します。

- CoreLocation.framework (Required)
- CoreBluetooth.framework (Required)

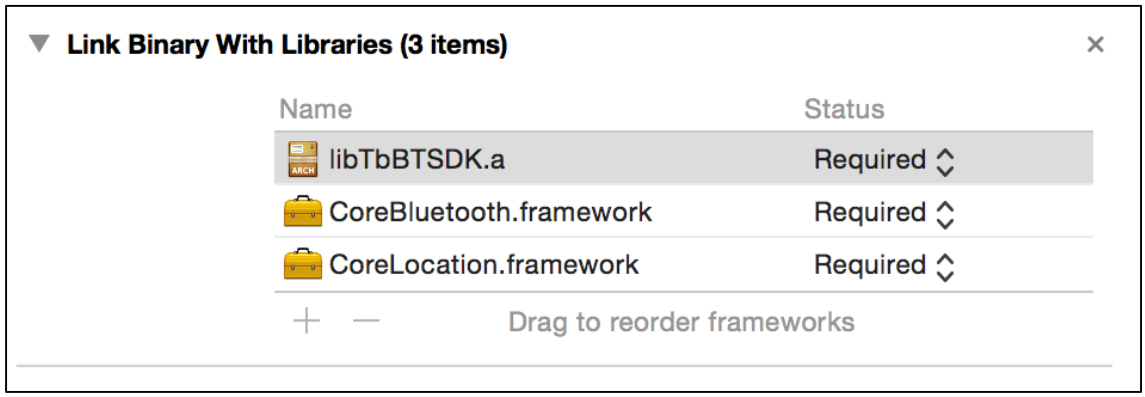

図 4　使用ライブラリ

SDK のファイル群を Drag & Drop で、「Copy Items if needed」にチェックを入れて、プロジェクトに追加します。

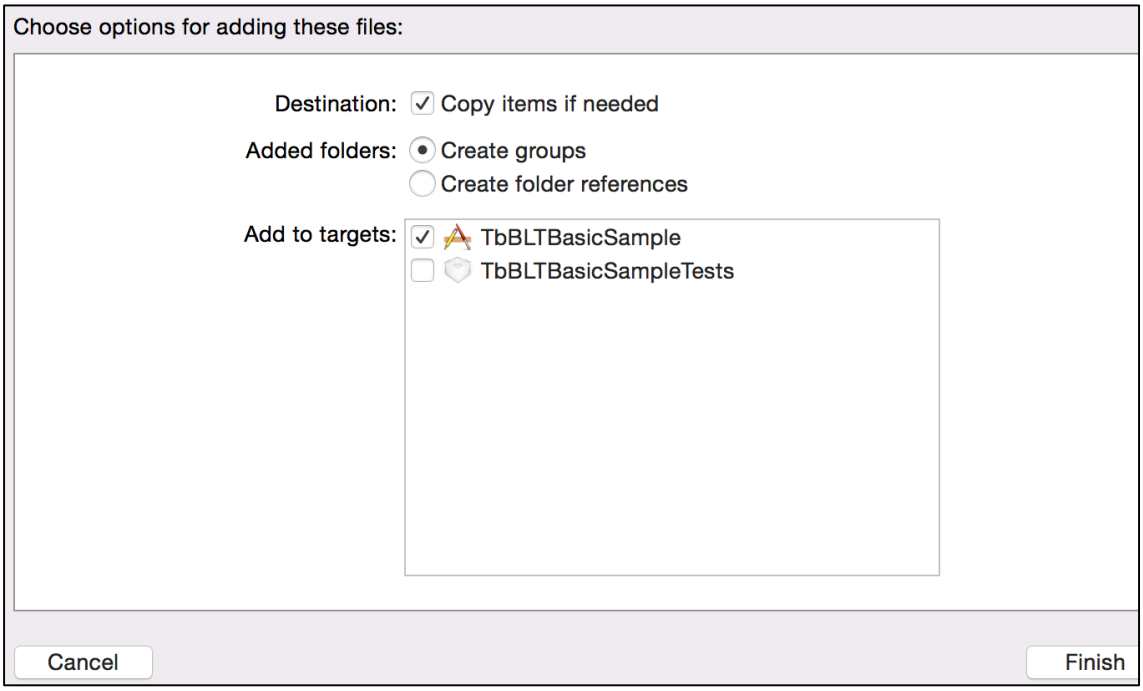

図 5　SDK ファイルコピー時のダイアログ設定

例えば、こんな感じ。

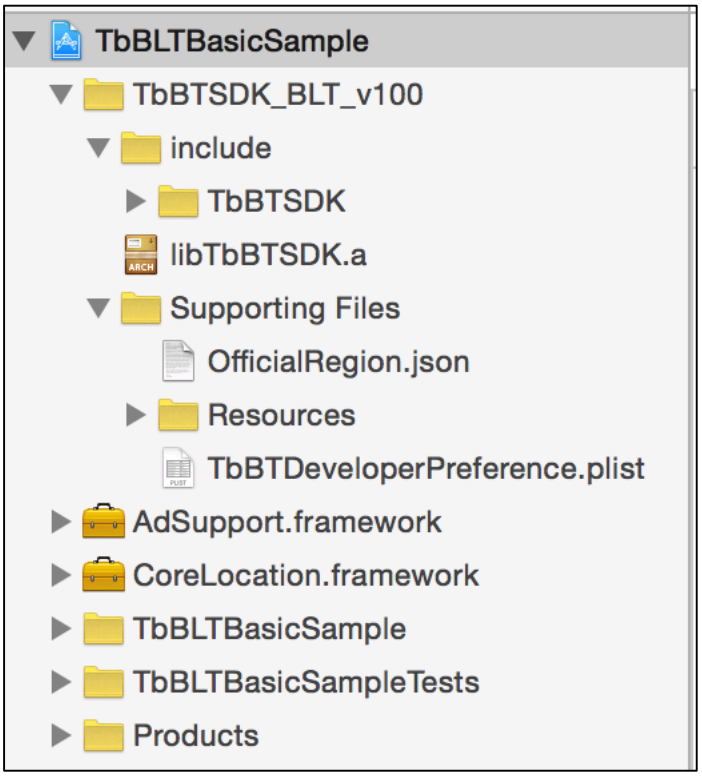

図 6　プロジェクトへの SDK 追加例（Xcode）

3-4 導入ステップ 3 - Other Linker Flags の指定

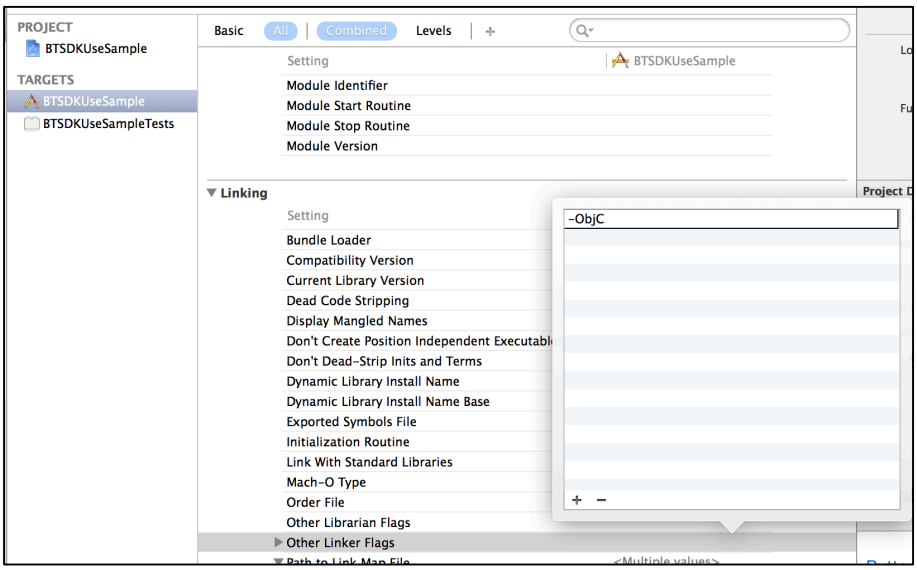

図 7　リンカーフラグ指定

プロジェクトの［Build Settings］–［Linking］–［Other Linker Flags］を指定します。「-ObjC」を追加します。

3-5 導入ステップ 4 - Info.plist への Description の登録 [iOS8.0 以上端末向け]

Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.1.2
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
NSLocationWhenInUseUsageDescription	String	このアプリは位置情報を使って限定コンテンツを提供しています
Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Status bar tinting parameters	Dictionary	(1 item)
▶ Supported interface orientations	Array	(1 item)

図 8　NSLocationWhenInUseUsageDescription の設定例

［NSLocationWhenInUseUsageDescription］を設定します。iOS8.0 以降の場合に、「アプリを使用中のみ（※）位置情報を使用」のユーザー許可を確認するダイアログを表示するのに必要となります。

※ アプリがフォアグラウンド状態

ユーザーが許可に納得できるようなメッセージの設定をお願い致します

（代替）アプリが起動していない場合でもビーコンに反応する機能を実装する場合

図 9　NSLocationAlwaysUsageDescription の設定例

[NSLocationAlwaysUsageDescription]を設定します。「位置情報の使用を常に許可」をユーザーに求めるダイアログを表示するのに必要となります。

3-6 導入ステップ 5 - TbBTDeveloperPreference.plist の編集

TbBTDeveloperPreference.plist ファイルを開き、アプリ固有の設定をします。ただし、「TestMode」と「DeveloperAppCode」以外はデフォルトのまま使用できます（有効）。

Key	Type	Value
▼ Root	Dictionary	(6 items)
TestMode	Boolean	YES
DeveloperAppCode	String	(YOUR_APP_TOKEN_IS_HERE)
SDKEdition	String	BLT
SDKVersion	String	1.0.0
MaxDesignatableBeaconRegions	Number	20
RegionUpdateIntervalHours	Number	24

図 10 SDK 設定ファイル

「TestMode」テストモードでの使用。デフォルトは「YES」です。「YES」の場合、テストメソッドを使用して、領域イベントのテストログをサーバに記録して閲覧できます

※　アプリの公開時に、「NO」に切り替えをお願いします

「DeveloperAppCode」に、運営から取得したアプリケーション固有のサービス・アクセストークンを設定します

「MaxDesignatableBeaconRegions」は、SDK を使ってアプリで登録可能なビーコン領域数です。

※　OS によって１アプリで同時モニタリングできる領域（ビーコン以外の領域含む）が 20 に制限されているため、SDK 外で指定した領域を使う場合は実際にモニタリングされるビーコン領域数は更に少なくなる可能性があります

「RegionUpdateIntervalHours」は、「端末上のアプリが前回領域情報を更新してから何時間は更新処理をスキップするか」の設定です。デフォルトでは 24 時間です。

※　「SDK Edition」と「SDK Version」は、本 SDK 固有の値ですので、変更せずにご使用ください。（変更されているとサービス通信ができなくなる可能性が出ます）

４期待処理要件と開発の流れ

本章では２章で挙げたようなアプリ使用のユーザー体験を実現するための基本実装について、簡易説明していきます。

４-１開発ステップ 1. サービス専用ビーコン領域情報の準備と、トラッキング有効判定のための設定

アプリを起動したタイミングで、サービス専用ビーコン領域の取得をサービス・サーバと連動して、また、アプリケーションのメタ情報を使用して有効なトラッキングができるようにセットアップをします。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
    (NSDictionary *)launchOptions {

    _skipBLT = NO;
    [TbBTPreliminary setUpWithCompletionHandler:^(BOOL success) {
        if (!success) { // Failed to set up
            _skipBLT = YES;
        }
    }];
    // 処理結果が不要なら、これも可能
    // [TbBTPreliminary setUpWithCompletionHandler:nil];
```

コード例　4-1　ビーコン連動サービス用の準備処理

TbPreliminary.h をインポートし、クラスメソッドの setUpWithCompletionHandler: をコールします。このメソッドは非同期処理を行い、準備を正常完了したら success=YES を返します。

（現行の OS では可能性は極小と思われますが）OS レベルで設定が保存できなかったり一度保存したデータが破損したりして、有効なビーコントラッキングができない状況と判断した場合は success=NO が返ります。

※処理的にはアプリに制限をかけることはありませんが、ビジネス的にデメリットが出ると判断される場合などには、機能制限をかける等の処理を、コールバックされた時に処理されるブロックに入れてください。

４-２開発ステップ 2. 位置情報の使用の許可（及び Bluetooth の確認）

※ SDK 導入前に処理が組み込み済みであれば、追加は不要です

```
// For iOS 7.x
if (NSFoundationVersionNumber_iOS_8_0 > NSFoundationVersionNumber) {
    [self prepareLocManager];
    [self prepareBeaconManager];
    [self gotoMenuPage];
    return;
}
// for iOS 8.0 ～
if (![TbBTManager isBeaconEventConditionMet]) {
    if  (!_locServiceStateDetermined || !_locServiceForAppDetermined)
        { // Location service status not checked
        _stateLabel.text = @"現在の設定では限定コンテンツが使用できません";
        [self checkLocServiceStateAndContinue];
    }
} else if (!_bluetoothStateDetermined){ // Bluetooth status not checked
    yet
    if (!appDelegate.locManager) {
        [self prepareLocManager];
    }
    [self prepareBeaconManager];
    [self checkBluetoothState];
} else { // Every setting is O.K.
    [self gotoMenuPage];
}
```

コード例 ４-2 バージョン別位置情報サービス許可確認の分岐

iOS8.0 以降の場合、明示的に CLLocationManager 型インスタンスの requestWhenInUseAuthorization メソッド（あるいは、requestAlwaysAuthorization メソッド）をコールして、ユーザーにアプリ個別（またはシステムサービスとしての）

位置情報サービス使用許可の確認をする必要があります。
上のコード例では、iOS8.0 以降のバージョンの場合は
checkLocServiceStateAndContinue メソッドをコールしています（続くコード例参照）。

iOS7.x の場合は、初めて位置情報サービスを使用した機能を使おうとしたタイミングでユーザーにダイアログが表示されます。
※　この場合もユーザー許可により、後述の コード例 4-5　ユーザー位置情報サービス許可状態変更時のコールバック処理が実行されます。

```
- (void)checkLocServiceStateAndContinue {
    NSLog(@"-- %s --", __func__);
    AppDelegate *appDelegate = (AppDelegate *)[UIApplication sharedApplication].
        delegate;
    if (!appDelegate.locManager && ![self prepareLocManager]) {
        // Can not use. Skip beacon service ..
        NSLog(@"何らかの制約によりサービス機能を初期化できません。スキップします");
        [self gotoMenuPage];
    } else if (![CLLocationManager locationServicesEnabled]) {
        NSLog(@"位置情報サービス自体がオフ");
        if (NSFoundationVersionNumber_iOS_8_0 <= NSFoundationVersionNumber) {
            assert(appDelegate.locManager);
            [appDelegate.locManager requestWhenInUseAuthorization];// Show loc
                service dialog by framework
        } else {
            _locServiceStateDetermined = YES;
        }
    } else if ([CLLocationManager authorizationStatus] ==
        kCLAuthorizationStatusNotDetermined) {
        NSLog(@"アプリに対しての位置情報サービス許可がされていない");
        _locServiceStateDetermined = YES;
        if (NSFoundationVersionNumber_iOS_8_0 <= NSFoundationVersionNumber) {
            [appDelegate.locManager requestWhenInUseAuthorization];// Show app
                permission dialog by framework
        }
    } else if ([CLLocationManager authorizationStatus] ==
        kCLAuthorizationStatusAuthorizedWhenInUse
                || [CLLocationManager authorizationStatus] ==
                    kCLAuthorizationStatusDenied
                || [CLLocationManager authorizationStatus] ==
                    kCLAuthorizationStatusRestricted) {
        NSLog(@"アプリに対しての位置情報サービス許可が決定した");
        // Location Service is O.K. Prepare beacon manager
        _locServiceForAppDetermined = YES;
    } else {
        _locServiceStateDetermined = YES;
        _locServiceForAppDetermined = YES;
        // Location Service is O.K. Prepare beacon manager
    }
}
```

コード例　4-3　位置情報サービス使用の許可リクエスト（複数回のコール用実装）

```objc
// Can be skipped if location manager already exists
- (BOOL)prepareLocManager {
    AppDelegate *appDelegate = (AppDelegate *)[UIApplication sharedApplication]
        .delegate;
    appDelegate.locManager = [[CLLocationManager alloc] init];
    appDelegate.locManager.delegate = appDelegate;

    if (appDelegate.locManager) {
        return YES;
    }
    return NO;
}
```

コード例　4-4 位置情報サービスマネージャの初期化

- CLLocationManager クラスの クラスメソッド locationServicesEnabled をコールして、OS での位置情報サービスの使用がオンになっているかをチェックします。オフの場合は CLLocationManager クラスのインスタンスを生成後、requestAlwaysAuthorization メソッドをコールすると、フレームワークにより使用を促すダイアログが表示されます。

- iOS8 以上（かつ既に位置情報サービスがオン）場合には、CLLocationManager authorizationStatus が kCLAuthorizationStatusNotDetermined の場合は requestWhenInUseAuthorization メソッドをコールして、位置情報の使用を常にアプリが使用できるように許可を求めます。

（Info.plist 内の NSLocationWhenInUseUsageDescription に設定のメッセージが表示されます）

- CLLocationManager を未使用のアプリの場合、新規インスタンスを生成
- CLLocationManager.delegate を適切なクラスのインスタンスに指定(コード例では、アプリ上に常時存在している AppDelegate を指定しています)

  ※ 他の処理で CLLocationManager のインスタンスを使用している場合は、デリゲートメソッド内処理を適切にすることで別途存在しているインスタンスを使用することも可能です

ユーザー選択により許可がされると、CLLocationManagerDelegate の locationManager:didChangeAuthorizationStatus:コールバックメソッドが呼ばれます。

```
- (void)locationManager:(CLLocationManager *)manager didChangeAuthorizationStatus:
    (CLAuthorizationStatus)status {
    NSLog(@"Location service authorization changed");
    if (status == kCLAuthorizationStatusNotDetermined) {
        NSLog(@"Service enabled but auth not determinted. Confirm again for app");
        [[NSNotificationCenter defaultCenter] postNotificationName:kBaseLocServiceEnabled
            object:self];
    } else if (status == kCLAuthorizationStatusAuthorizedWhenInUse) {
        NSLog(@"Callback WhenInUse permitted");
        [[NSNotificationCenter defaultCenter] postNotificationName:kAlwaysLocServicePermitted
            object:self];
    } else if (status == kCLAuthorizationStatusRestricted || status ==
        kCLAuthorizationStatusDenied) {
         NSLog(@"Callback alwarys denied/restricted");
        [[NSNotificationCenter defaultCenter] postNotificationName:kAlwaysLocServiceDenied
            object:self];
    }
}
```

コード例 4-5 ユーザー位置情報サービス許可状態変更時のコールバック

「アプリの使用中に許可」を表す kCLAuthorizationStatusAuthorizedWhenInUse ステータスと、「拒否された」系のステータス kCLAuthorizationStatusRestricted/kCLAuthorizationStatusDenied で処理を分け、UI 処理で「次へ進む」や「自前アラートを再度出す」という処理につなげるのが一般的かもしれません。

４-３開発ステップ３．ビーコン連動メインクラスの初期化指定

位置情報の許可が得られたら、本サービスが機能する状態（ただし Bluetooth がオフにされると検出に失敗します）なので、SDK の検出機能を使う準備します。

・ 初めてのアクティベートのタイミングで、サービス使用に関わる確認事項に対する同意がされたことを明示する同意フラグ値：YES を与えて、メインクラス TbBTManager のインスタンスを初期生成します。

・

```
- (BOOL)prepareBeaconManager {

    _btManager = [TbBTManager sharedManager];
    if (!_btManager) {
        _btManager = [TbBTManager initSharedManagerUnderAgreement:YES];
    }
    if (_btManager) {
        _btManager.delegate = self;
        return YES;
    }
    return NO;
}
```

・ **コード例 ４-6 TbBTManager の初期化**

・ [TbBTManager sharedManager]メソッドをコールすると、アプリケーションで共有される１つのマネージャインスタンス(TbBTManager クラス)が返されますが、初回に「サービスの利用をする」という明示的な同意フラグ（ユーザーが条件を確認した後を想定しています）を initSharedManagerUnderAgreement:YES のメソッドをコールしていない限り、常に nil が返ります。一度だけ、initSharedManagerUnderAgreement:YES をコールするようにしてください。
（現仕様では バックグラウンド起動での処理の簡易化のため１度目の生成で同意フラグ：YES が渡っていると SDK 側でも同意状態を保持するので、２回目以降は＋**sharedManager** クラスメソッドで生成できます）

ユーザーの意思に連動して機能の使用を明示的に止める場合は、(同意フラグ値として) NO が渡されると、以降 sharedManager で nil が返され、サービスの使用はできなくなります。

・ TbBTManager のインスタンスに、delegate のオブジェクトを指定します。

4-4 開発ステップ3'. 現状での Blutooth 使用可否の状況確認

アプリ側のみで Bluetooth の状況チェックをすることもできますが、TbBTManager にチェックメソッドを用意しています。チェック結果のコールバック先として、TbBTManagerDelegate インタフェースの実装が必要です。

```
- (void)checkBluetoothState {
        NSLog(@"--%s--", __func__);
    // [self prepareBeaconManager] is required before this call
    TbBTManager *btManager = [TbBTManager sharedManager];
    assert(btManager != nil);
    assert([btManager.delegate isEqual:self]);
    [[TbBTManager sharedManager] checkCurrentBluetoothAvailability];
}
```

```
- (void)didDetermineBlutoothAvailability:(BOOL)available {
    NSLog(@"--%s--", __func__);
    if (!available) {
        //[self showCustomAlert];
        [self showDefaultAlertWithCBFramework];
    } else { // Now Bluetooth is ON
        _bluetoothStateDetermined = YES;
       [self performSelectorOnMainThread:@selector(gotoMenuPage) withObject:nil
           waitUntilDone:NO];
    }
}
```

コード例 4-7 Bluetooth 状態チェックとコールバック

TbBTManager クラスの checkCurrentBluetoothAvailability をコールします。
端末側での正しい状態のチェックにタイムラグが発生するので、デリゲート指定されているオブジェクトに非同期に結果をコールバック
(didDetermineBluetoothAvailablity:) します。
このメソッドは YES/NO のみを返し、発信しているビーコン電波の受信が可能であれば、available に YES を、ビーコン電波の受信ができない状態であれば、NO を返しま

す。

CoreBluetooth フレームワーク使う場合、以下のようなアラートダイアログが表示されます。

４-５開発ステップ 4. 専用ビーコン情報の取得と、ビーコン検出の開始

ビーコン領域専用コンテンツを表示するビューを呼び出す前（またはビューを再表示するタイミング）で、ビーコン検出機能が有効な状態であれば、ビーコンの検出を支持します。

```
AppDelegate *appDelegate = (AppDelegate *)[UIApplication sharedApplication]
    .delegate;
NSArray *tbBeaconRegions = [appDelegate.btManager initialRegions];
_workingRegion = [tbBeaconRegions objectAtIndex:0];
[appDelegate.locManager startRangingBeaconsInRegion:_workingRegion]; //
    Wait for callback
```

コード例　4-8　サービスビーコンの検出開始処理

- TbBTManager の initialRegions メソッドで、サービスビーコン領域情報を取得します。
- 領域情報はリストですが、現在は１つ目の情報がデフォルトの有効ビーコン情報です。
  １つ目の領域を指定すると、CLBeaconRegion 型のオブジェクトが返るので、CLLocationManager のメソッド startRangingBeaconsInRegion:にパラメータとして渡すことで、ビーコンの検出を開始します。

４-６開発ステップ５. 見つかったビーコンの情報送信と、キーコード取得

ステップ２で問題なくサービスビーコン領域のレンジング処理が指示された場合、アプリケーションがアクティブな間は約１秒間隔でビーコンの計測が続き、CLLocationManagerDelegate の locationManager:didRangeBeacons:inRegion: メソッドがコールされます。

※　失敗した場合は、locationManager:rangingBeaconDidFailForRegion:withError がコールされます。

このメソッド内にて、見つかったビーコンのトラッキング送信と、ビーコンキーの取得を行います。

処理を実行するのは、TbBTManager クラスの beaconsTrack: ofRegion メソッドです。

```
- (void)locationManager:(CLLocationManager *)manager didRangeBeacons:(nonnull
    NSArray<CLBeacon *> *)beacons inRegion:(nonnull CLBeaconRegion *)region {
    _btManager = [TbBTManager sharedManager];
    NSArray *beaconKeyDatas = [_btManager beaconsTrack:beacons ofRegion:region];
    if (beaconKeyDatas) { // may be inside of 3b beacon region
        if (beaconKeyDatas.count > 0) {
            NSMutableArray *beaconKeys = [NSMutableArray array];
            for(TbBTServiceBeaconData *beaconData in beaconKeyDatas) {
                [beaconKeys addObject:beaconData.keycode];
            }
            // stop ranging for 3b beacon
            [manager stopRangingBeaconsInRegion:region];
            // get beacon region mapped contents
            [self execMappedContentFetch:beaconKeys];
        } else {
            // stop ranging for 3b beacon
            [manager stopRangingBeaconsInRegion:region];
            // could not detected 3b beacon
            [self execThisActionBeacauseBeaconsNotDetectedInRegion];
            NSLog(@"Ranging finished.");
        }
    } // else do not stop (called again)
}
```

コード例　4-9　計測されたビーコンのキーコードの取得

処理ステップは、

1. TbBTManager インスタンスの取得
2. locationManager:didRangeBeacons:inRegion で返された beacons と region を TbBTManager の beaconsTrack:ofRegion のパラメータとして渡す。
3. beaconsTrack メソッドが応答として nil を返している間は何もしない（またはビーコンと関係ない別処理をする）

4-1. beaconsTrack が TbBTServiceBeaconData 型のリストを返した場合、ビーコンのキーコード(TbbTServiceBeaconData オブジェクトの keycode プロパティ)を抜き出し、事前にビーコンキーコードに関連付けをしてあるコンテンツ情報
を取得する。同時に、stopRangingBeacons:inRegion メソッドをコールして検出処理を止める。
4-2. beaconsTrack が空リストを返した場合、ビーコンが見つからなかった場合の処理を実行する。同時に、stopRangingBeacons:inRegion メソッドをコールして検出処理を止める。

注）アプリがアクティブな場合は延々と、バックグラウンド計測を開始した場合は OS が止めるまで 3 分程度 locationManager:didRangeBeaconsInRegion:がコールされるので、必要な処理が終わったり、エラーを検知したりビーコン検知がコールバックされずタイムアウト条件に達した場合は、stopRangingBeaconsInRegion メソッドで計測処理を停止してください。

[※　応答仕様]
beaconsTrack メソッドは、指定回数（現バージョンでは 5 回）期待されるパラメータ（サービス用領域情報と、didRange で返された beacons リスト）でコールされるまで、与えられた beaacons のうちの重複しない分の累積処理をし、nil 応答し続けます。

4-7 開発ステップ 6. 得られたビーコンのキーコードからの関連コンテンツ取得

ビーコンのキーコード（配列）が得られたら、キーコード毎にコンテンツを取り出します。

```
- (NSInteger)prepareContentsForTbBeacons:(NSArray *)beaconKeys {
    NSMutableArray *dummyCheckedArray = [NSMutableArray array];
    for (NSString  *beaconKey in beaconKeys) {
        OurContent *mappedContent = [self dummySearchWithDummyMap:beaconKey];
        if (mappedContent) {
            [dummyCheckedArray addObject:mappedContent];
        }
    }
    _currentMappedContnents = [NSArray arrayWithArray:dummyCheckedArray];
    if (dummyCheckedArray.count == 0) {
        return 0;
    }
    return _currentMappedContnents.count;
}
```

コード例　4-10 キーコードから関連するコンテンツ情報の取得

コンテンツがどこに存在するかで、ローカルデータベースから取得、リモートサーバにキーコードを送信して要求する等、適宜取得方式を選択して実装してください。

以上が基本的な実装ポイントとなります。

実際のアプリとしては、起動タイミングとの関係、CLLocationManager や CLLocationManagerDelegate となるクラスの使い方、UIViewController の構成などによって失敗時の代替処理等があると思いますが、アプリの仕様に合わせて実装ください。

効果測定ツール「Force Operation X（F.O.X)」 連携

F.O.X SDK（https://github.com/cyber-z/public-fox-ios-sdk/）を使用して、ビーコン領域限定イベントの発生状況を、F.O.X システム上にて確認することができます。

[導入]
F.O.X SDK のインストールおよび、設定方法に従い、F.O.X SDK の「ForceAnalyticsManager」クラスを使用できるようにします。

※ *OTHER_LINKER_FLAGS の設定が正しくないとビルドエラーが発生することがあります。エラーが出る場合は、Other Linker Flags に、$(inherited) を追記します*

[実装]

SWAMP SDK を使用して検知されたビーコンキーを取得したあとで、F.O.X SDK のアクセス解析用機能、ForceAnalyticsManager sendEvent:action:label:value メソッドをコールします。

```
TbBTManager *btManager = [TbBTManager sharedManager];
NSArray *beaconKeyDatas = [btManager beaconsTrack:beacons ofRegion:region];
if (beaconKeyDatas) { // may be inside of 3b beacon region
    if (beaconKeyDatas.count > 0) {
        NSMutableArray *beaconKeys = [NSMutableArray array];
        for(TbBTServiceBeaconData *beaconData in beaconKeyDatas) {
            [beaconKeys addObject:beaconData.keycode];
        }
        // stop ranging for 3b beacon
        [manager stopRangingBeaconsInRegion:region];
        // get beacon region mapped contents
        /* ここで事前にビーコンにマッピングされた限定イベントを発生させます */
        [self execMappedContentFetch:beaconKeys];
    } else {
        // stop ranging for 3b beacon
        [manager stopRangingBeaconsInRegion:region];
        // could not detected 3b beacon
        [self execThisActionBeacauseBeaconsNotDetectedInRegion];
        NSLog(@"Ranging finished.");
    }
```

beacon キーを取得できた場合、あらかじめ用意されたイベント用コンテンツ等

を準備する前後のタイミングでの処理に１行追加します（次の例参照）

```
# pragma mark Content Handling method

- (void)execMappedContentFetch:(NSArray *)beaconKeys {
    NSLog(@"-- %s --", __func__);
    // Get content from content store
    ContentManager *contentManager = [ContentManager sharedManager];
    NSUInteger numOfFoundContents = [contentManager
        prepareContentsForTbBeacons:beaconKeys];
    NSLog(@"No. of beacon mapped contents in this timing : %lu", (unsigned long
        )numOfFoundContents);
    if (numOfFoundContents > 0 && numOfFoundContents < 6) { // 同時検出6個以上のケ
        ースは検証だろうからとりあえず何もしない
    /*
     *   F.O.X イベント解析用に、各ビーコン領域で取得されたコンテンツの情報を送信（IDや名称
          など）
     */
        NSArray *contentsArray = [contentManager mappedContentsForTbBeacons];
        for (OurContent *content in contentsArray) {
            NSString *eventDescrption = [@"SWAMP領域内イベント"
                stringByAppendingString:content.title];
                    [ForceAnalyticsManager sendEvent:eventDescrption action:nil
                        label:nil value:1];
        }
    }
    // Post notification to the controller
    [[NSNotificationCenter defaultCenter] postNotificationName:
        kBeaconMappedContentsPrepared object:self];
}
```

図 11 F.O.X による検知ビーコン内イベントの記録送信例

event パラメータとして、SWAMP 領域での発生イベントであることを示す文字列を渡します。このときパラメータに、ユーザーに提供される限定コンテンツ ID などを付加すると、どのビーコンに反応したかが判ります。

action パラメータ値と label パラメータ値には nil を指定してください。