Phunware SDK On-boarding

Welcome to Phunware SDKs

Welcome to Phunware's SDK Documents. On this page you will find instructions on how to on-board Phunware's SDKs for Mobile Engagement and Wayfinding for both iOS and Android development environments. After you have completed the on-boarding process, Follow the links to additional SDK documentation and the SDK repos.

What Do Phunware SDKs Do?

Phunware Core SDK

Phunware's Core SDK is required and integrated into all Phunware modules to provide security; login functionality; Org, User, and Role Management, Application keys, Content Management functionality, and Analytics.

Because Core is fully integrated, you will not need to import this SDK separately or complete any additional steps to add this SDK.

Included in Phunware's Core SDK are:

- Security-Login
- Org Management
- User and Role Management
- Application Management
- Content Management Module
- Analytics Module

Security Protocols

- SSL used for all transmission
- Building Bundles are encrypted with AES-256

Important Links

Android	iOS
Location Based Services > Android SDK References	Location Based Services > iOS SDK References
Mobile Engagement > Android SDK References	Mobile Engagement > iOS SDK References
Mapping SDK > Android Repo	Mapping SDK > iOS Repo
Location SDK > Android Repo	Location SDK > iOS Repo
Mobile Engagement SDK > Android	Mobile Engagement SDK > iOS Repo
Mapping SDK Classes > Android	Mapping SDK Classes > iOS
Location SDK Classes > Android	Location SDK Classes > iOS
Mobile Engagement SDK Classes > Android	Mobile Engagement SDK Classes > iOS
All Platforms	
LBS (Mapping) Configuration Guides (MaaS Portal)	
Marketing Automation Configuration Guides (MaaS Portal)	
LBS (Mapping) Training Videos (MaaS Portal)	
Marketing Automation Training Videos (MaaS Portal)	

Mobile Engagement SDK (for Marketing Automation)

Marketing Automation campaigns are displayed as notifications on the mobile devices of app users, and can appear even when devices are locked. When an user taps the notification, the app

home page is displayed by default. If a specific promotion or metadata is configured, then the click will display the promotional content or the page specified by the metadata.
Marketing Automation campaigns with promotions configured through MaaS portal can also appear in a Message Center (in addition to being received as a notification) where users can click to a message detail page. From here, users can click a link to launch a webview displaying the attached HTML promotion.

Additional development is required to enable a Message Center for SDK implementations. Message Center is available out-of-the-box for Vertical Solutions implementations.

Users can delete messages:

- From the list view
  - iOS: by tapping Edit and selecting items to delete
  - Android: by long pressing an item and confirming deletion
- From the detail view, by tapping the delete link

Users can also swipe between message from a message detail view.

Broadcast Campaigns (Alerts and Notifications)	**Geofence Entry/Exit Campaigns**	**Beacon Entry/Exit Campaigns**
Reach broad app audiences with messages distributed at specific times.	Target messaging to reach audiences when they are in close proximity to your location.	Use beacons to target audiences at a more granular level, such as departments, wings or near product displays.
Target specific audience segments with more contextually relevant messages based on user actions or preferences.	Use geo-fences to target a location / physical venue level or a wider surrounding areas like a campus, neighborhood, or city.	Use beacons to target audiences in the moment when they are in your venues.
Drive and measure foot traffic to physical locations using notification messages to announce events.	Target app audiences with hyper-relevant contextual messages as they leave your venues to reward them for stopping by and to encourage return visits. AND Drive and measure foot traffic to physical locations using notification messages to announce events.	
Enhance your notifications and reward your audiences with rich HTML promotional content (offers, coupons, specials etc.).		

Location-Based Services (for Mapping and Wayfinding)

Phunware's Location Based Services (LBS) module is driven by the Mapping and Location SDKs, providing application developers with a set of tools to enable the display of indoor venue maps, points of interest, and wayfinding throughout these venues.

Developers can utilize these LBS SDKs to offer app users either:

- Static wayfinding, where users can interact with a map with highlighted routes and step by step directions, but do not see their actual position or movements reflected on the map
- Dynamic wayfinding, where users can interact with a map with highlighted routes and step by step directions, including a blue dot that mirrors users' their actual position and movements.

This guide provides the most common Location Based Services use cases important to SDK developers. It is NOT an exhaustive list of the use cases that can be developed using the Phunware Mapping and Location SDKs.

The APIs and data libraries provided offer a framework and expose the necessary data to offer

mapping/wayfinding experiences. In addition, a sample including basic user interface code is provided, which can be used as a starting point by developers. This basic UI can be used as is or customized and enhanced to support specific app use cases and branding needs. For example, additional use cases (not included in this document) that a developer might add to an app include saving a custom POI or sharing a user's location. These types of customized behaviors are driven by the application code using the data libraries and APIs available through the SDK.

SDK On-boarding

Android On-boarding IOS On-boarding

Android Requirements

- Android SDK 4.0.3+ (API level 15) or above
- Android Support Library v4 24.0+

General On-boarding Steps for ALL Android SDKs

Step 1: Add the Phunware Maven remote repository

Insert this block into allprojects -> repositories: Add the following to your `repositories` tag in your top level `build.gradle` file.

```
allprojects {
    repositories {

        maven {
            url "https://nexus.phunware.co
        }
    }
}
```

Step 2: Add Needed Permissions

In the Android manifest, the following permissions are needed:

NOTE: BLE and Android SDK version 4.3 (API 18) or above are required if Bluetooth is actually is going to be used. If your users only are going to use Bluetooth for positioning, you should change the android:required attribute to true (this will filter out non-BLE devices at Google Play). Observe that the WAKE_LOCK permission can be removed, but background navigation performance might become worse.

```
<uses-permission android:name="android.per
<uses-permission android:name="android.per
<uses-permission android:name="android.per
<uses-permission android:name="android.per
<uses-permission android:name="android.per
<uses-permission android:name="android.per
<uses-permission android:name="android.per
<uses-feature android:name="android.hardwa
android:required="false"/>
```

Step 3 - Retrieve App ID, Access Key and Signature Key from MaaS Portal

Add your app to Mass Portal.

1. Login with your Org credentials
2. Click Apps in the left nav
3. Click New at the top right
4. Enter your App Name
5. Select the App Platform
6. Select the App Category (for app store)
7. Click Save
8. In the App list find you app and click the ellipsis.
9. Select Show Keys in the action menu
10. The keys are exposed that you need to enter in the Manifest.

The App UUID is required if you are using Phunware's App Builder to create your apps.

Step 4: Add Phunware key resources to strings.xml for App Id, Access Key, Signature Key

Add the keys obtained in step 4 to strings.xml.

```
<string name="app_id">APPID</string>
<string name="access_key">ACCESSKEY</strin
<string name="sig_key">SIGKEY</string>
```

Step 5: Add Phunware keys for App Id, Access Key, and Signature Key to Manifest

Add the keys obtained in step 4 to Manifest.

```
<meta-data android:name="com.phunware.APPL
android:value="@string/app_id" />
<meta-data android:name="com.phunware.ACCE
android:value="@string/access_key" />
<meta-data android:name="com.phunware.SIGN
android:value="@string/signature_key" />
```

On-boarding Steps for Android Mapping/Location SDKs

Step 1: Add the Mapping SDK as a dependency in your app's build.gradle file

These two lines includes all dependencies necessary to use the Phunware Mapping and Location SDKs.

The Core SDK (and all of its parts) is automatically integrated with the Mapping SKD.

NOTE: The SDK version number in should always reflect the most recent build from the GitHub repository.

```
apply plugin: 'com.android.application'


android {
    ...
}

dependencies {
    ...
    compile 'com.phunware.mapping:mapping:
 compile 'com.phunware.location:provider-m
    ...
}
```

Step 3: Add your Google Maps Api Key to AndroidManifest.xml

Create a string resource that contains your Google Maps API Key.

If you need help getting a Google Maps API Key, please find instructions here: Get a Google Maps API key

```
<meta-data android:name="com.google.androi
           android:value="@string/google_m
```

On-boarding Steps for Android Engagement SDK

Step 1: Setup GCM Service in MaaS

If you are planning on sending Alerts and Notifications (Broadcast Campaigns), you have obtained Android and iOS keys needed to push notifications from your app.

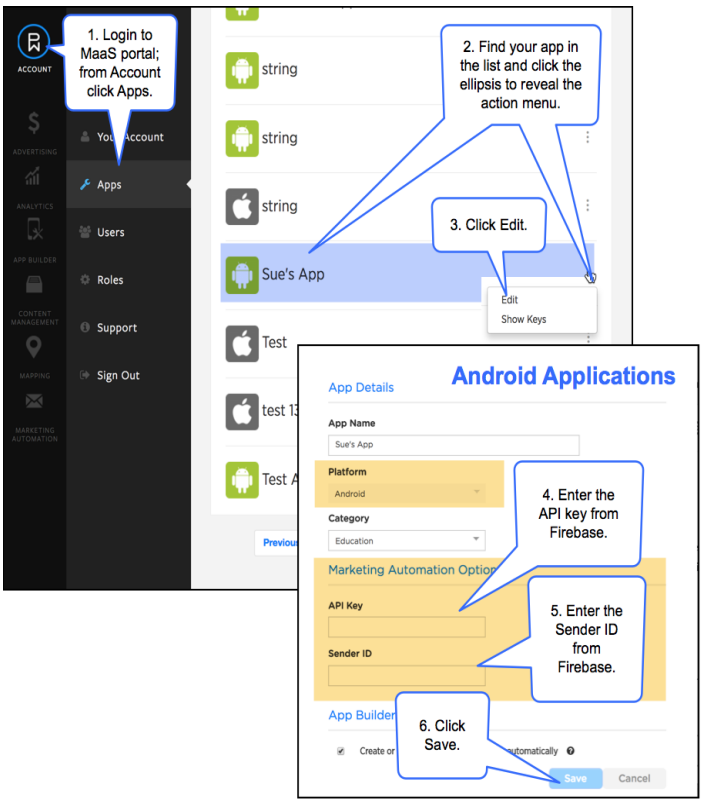
To obtain Android API Key and Sender ID read:

Google Cloud Messaging Guides-Registering Client Apps

Navigate to Phunware's MaaS portal to find your App.

After you have obtained your keys:

1. Go to the MaaS portal Accounts -> Apps and find the desired App in the alphabetical list
2. Click on the ellipsis to reveal the Action Menu
3. Click Edit
4. Enter your keys under Marketing Automation Options
5. Click Save



Step 2: Add the Mobile Engagement SDK as a dependency in your app's `build.gradle` file

The Core SDK is automatically imported with the Mobile Engagement SDK.

```
apply plugin: 'com.android.application'

android {
  ...
}

dependencies {
  ...
     compile 'com.phunware.engagement:mobil
     ...
}
```

Step 3: (Optional) Add beacon support if you are using beacons

If you would like to take advantage of the Mobile Engagement SDK's beacon support, simply add the `beacon-location-manager` dependency.

With properly configured beacons in your environments, no other code changes are required to take advantage of beacon-based messaging.

```
dependencies {
  ...
  compile 'com.phunware.engagement:mobile-
  compile 'com.phunware.engagement:beacon-
  exclude group: 'com.phunware.engagement'
  }
  ...
}
```

Step 4: Add Location and Storage permissions to Manifest

This allows you to utilize location-based messages and beacon messaging.

> NOTE: Background location notifications currently cannot work with runtime permissions required for apps targeting Android SDK level 23 and higher, so your targetSdkVersion in your build.gradle file must be 22 or lower.

```
<uses-permission android:name="android.per
```

Step 5: Configure the Mobile Engagement SDK with your environment

You should only initialize the Mobile Engagement SDK once, after you initialize PwCoreSession. Once it's complete, you can access the Location, Message and Attribute managers directly.

Once initialization is complete, users will be automatically notified with your custom broadcast messages. If you have location and storage permissions they will also be able to receive messages for location events, like entering a retail store.

```
public class MyApplication extends Applica

  @Override
  public void onCreate() {
    super.onCreate();
 //initialize PwCoreSession
 PwCoreSession.getInstance().registerKeys(

 //initialize Engagement
    new Engagement.Builder(this)
   .appId(/*your app ID from the MaaS port
         .build();

 //start location manager to receive locat
 Engagement.locationManager().start();
  }

}
```

Step 6: Designate an Activity to launch from notifications

Notifications can be customized by extending the `NotificationCustomizationService.`

This launches your activity from a notification with message and promo information.

```
<activity
    android:name=".MessageActivity" >
    <intent-filter>
        <action android:name="android.inte
        <category android:name="android.in
        <data android:mimeType="engagement
    </intent-filter>
</activity>
```

```
if (getIntent().getAction() == Intent.ACTI

    Message intentMessage =
getIntent().getParcelableExtra(MessageMana
    Engagement.analytics().trackCampaignAp
            intentMessage.campaignType());

    boolean hasPromo = getIntent()
            .getBooleanExtra(MessageManage
    if (hasPromo) {
        long messageId = intentMessage.cam
  Engagement.messageManager().getMessage(m
      @Override
      public void onSuccess(Message data)
          //do something with the message
      }

      @Override
      public void onFailed(Throwable e) {
          Log.e(TAG, "Failed to get messag
      }
  });
   }
```

Step 7: Show Messages

Using the `MessageManager` you can easily show a list of available messages.

> NOTE: Calls to the MessageManager are asynchronous, and may require loading data from the network.

```
public class MessageListFragment extends F

  @Override
  public void onActivityCreated(@Nullable
    super.onActivityCreated(savedInstanceS

    Engagement.messageManager().getMessage
      @Override
      public void onSuccess(List<Message>
        //do something with the Messages
      }

      @Override
      public void onFailed(Throwable e) {

      }
    });
  }


}
```

Step 8: Customizing Notifications

This service allows app developers to customize notifications. It must be implemented initially even if the user would like to just use standard out of the box notifications.

```
 <manifest ...>
  <application>
    <service android:name=".MyNotification
      <intent-filter>
        <action android:name="com.phunware
      </intent-filter>
    </service>
  </application>
</manifest>
```

```
public class MyNotificationEditService ext

  @Override public void editNotification(N
notificationBuilder) {
    notificationBuilder
        .setSound(RingtoneManager.getDefau
        .setSmallIcon(R.drawable.ic_motorc
  }

}
```

Step 9: Enable Push Notifications New in Version 3.1.2 is the ability to use the SDK with or without push notifications as a feature. Enabling push notifications is as easy as enabling them through a simple call to the Engagement API	`Engagement.enablePushNotifications(this.ge`

iOS Requirements

- iOS 9.0 or greater
- Xcode 8 or greater

General On-boarding Steps for ALL iOS SDKs

Step 1: Initialize Cocoapods and add SDK(s) to podfile

a. Initialize Cocoapods if your project does not already use it. If your environment does not already have Cocoapods installed, enter "sudo gem install cocoapods" in Terminal. After Cocoapods is installed, navigate to your project's root directory in the Terminal. Enter "pod init" in Terminal.

b. In the Podfile file that is generated, modify the text in Xcode or your preferred text editor. Add the line "pod 'PWMapKit'" or pod 'PWEngagment'" after the "target" line and before the "end" line. Save the file.

PWMapKit Includes	PWEngagement includes
• PWCore • PWMapping • PWLocation	• PWCore • PWEngagement

c. Enter "pod update" in Terminal while in the directory that contains the Podfile.

d. Use the .xcworkspace file that was generated by the pod install from now on.

e. In your project's Info.plist file, add a row with key: "Privacy - Location When In Use Usage Description" (Xcode should autocomplete this) and for value, add what you want your users to see when prompted for Location use by the device.

f. (For Mobile Engagement) Navigate to Build Phases inside your Xcode Projects and click on Link Binary With Libraries add following frameworks:

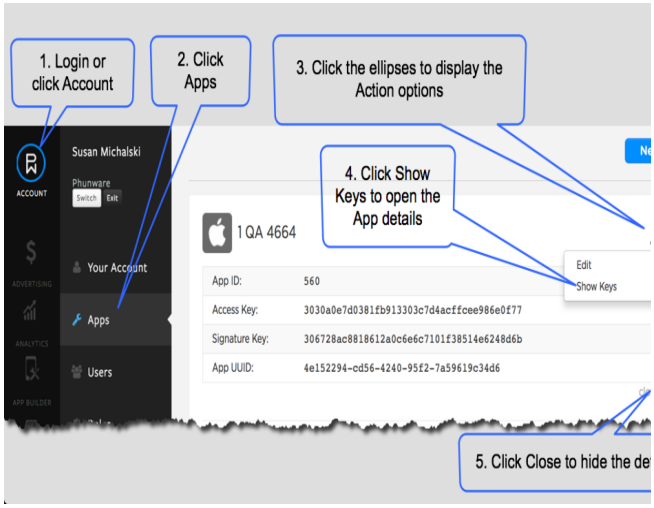
- CoreLocation.framework
- libz.tbd
- libsqlite3.tbd

```
pod 'PWMapKit'
```

Step 2: Get App Keys from MaaS Portal

- MaaSAppId : The application ID matching the server choice.
- MaaSAccessKey : The accesskey for your application.
- MaaSSignatureKey : The signature key for your application.

1. Login to MaaS portal and click Apps in the menu.
2. Find your app in the App list and click the ellipsis to open the action menu.
3. Click Show Keys.
4. Copy these keys from MaaS and paste the value in the corresponding plist rows.



Step 3: Add Core call in AppDelegate

In your application's AppDelegate, add the following:

> TIP: Be careful not to add spaces around the keys as this will cause the build to fail.

```
[PWCore setApplicationID:@"YOUR_APPI
                accessKey:@"YOUR_ACCE
            signatureKey:@"YOUR_SIGN
```

Step 4: Set Required background modes values.

`App registers for location updates:` (This value allows the app to keep users informed of their location, even while it is running in the background.)

`App downloads content in response to push notifications`: (This value allows the app to regularly download remote notifications.)

NSLocationAlwaysAndWhenInUseUsageDescription: (This is the message you want to display on the prompted alert when the user grants the app permission to use the location service.)

On-boarding Steps for iOS Mapping/Location SDKs

Step 1: Setup the map view

The primary use of the components of PWMapKit revolve around creating a map view, displaying points of interest, showing the user's location and indoor routing.

The example shows how to instantiate a PWMapView and load a building from the sample application, where "mapView" is a PWMapView property of calling class.

```
self.mapView = [[PWMapView alloc] ir
self.mapView.delegate = self; // se
protocol

__weak typeof(self) weakself = self
[PWBuilding buildingWithIdentifier:<
completion:^(PWBuilding *building, N
    if (building) {
        [weakself.mapView setBuildir
    }
}];
```

Step 2: Register the Location Provider (for Blue Dot implementations)

The PWMapView can display a user location on the map if a location provider is registered with the PWMapView.

The location providers are in the PWLocation framework, and each different provider requires different steps to set up.

(See Readme at https://github.com/phunware/maas-location-ios-sdk to view setup examples of all different provider options).

Once the location provider is initialized, the call may be used to register the provider with the PWMapView.

```
PWManagedLocationManager *manager =
initWithBuildingId:<YOUR_BUILDING_ID
[mapView registerLocationManager:mar
```

Step 3: Calculate and show a route

Simply feed the PWRoute class method two points of interest and whether or not accessibility should be considered when calculating the route.

accessibility:YES will not use access points marked as "inaccessible" in the portal, e.g. stairs between floors.

```
__weak typeof(self) weakSelf = self
[PWRoute initRouteFrom:<#(PWPointOf
*)#> accessibility:<#(BOOL)#> comple
    [weakSelf.mapView navigateWithRo
PWMapView
}];
```

Step 4: Update current route

As a user traverses a route, PWMapKit will post notifications to tell the developer when the next routing instruction is reached.

```
[[NSNotificationCenter defaultCente
selector:@selector(myInstructionUpda
name:PWRouteInstructionChangedNotif
```

On-boarding Steps for iOS Engagement SDK

Step 1: Add p12 Push Notification Certificate to App in MaaS Portal

A. In your Apple Developer's Console get your p12 Certificate. You will download this certificate to MaaS Portal to enable Broadcast Campaigns.

Apple's documentation

Apple Push Notification services (APNs) tutorial

B. Once it's created, download the push production certificate and add it to Keychain Access. Then, from Keychain Access, export both the certificate and key.

(Right click to view the Export option) as a .p12 and set a password.

C. Log in to Maas portal and navigate to the app created for your application and update the following.

Click **Apps** to go to your application list.

Either Click **NEW** or find the App you want to send push notifications for and click **Edit** in the Action menu

Complete the Marketing Automation section as follows:

p12 Certificate: Upload the certificate you downloaded to your local drive

Password: The password you setup for the push certificate.

Environment: Select Production environment for production apps.

Step 2: Beacon and Geofence Registration

As of PWEngagement, the *application developer* nee ds to implement the following delegate in launch options. This registers for the remote notifications and notifies user when app is launched from a local notification (Geofence/ Beacon).

```
[PWEngagement didFinishLaunchingWith
withCompletionHandler:^BOOL(PWMELoca

}]
```

Step 3: Remote Notification Handling

Apple has these primary methods for handling remote notifications. You will need to implement these in your application delegate, forwarding the results to PWEngagement:

```
- (void)application:(UIApplication *
didRegisterForRemoteNotificationsWit
{
    [PWEngagement
didRegisterForRemoteNotificationsWit
}

- (void)application:(UIApplication *
didFailToRegisterForRemoteNotificat:
{
    [PWEngagement didFailToRegisterF
}

- (void)application:(UIApplication *
didReceiveRemoteNotification:(NSDict
{
    [PWEngagement didReceiveRemoteNo
withNotificationHandler:^(PWMELocalN

    }];
}

- (void)application:(UIApplication *
didReceiveRemoteNotification:(NSDict
fetchCompletionHandler:(void (^)(UIB
{
    [PWEngagement didReceiveRemoteNo
fetchCompletionHandler:completionHar
withNotificationHandler:^(PWMELocalN

    }];
}
```

Step 4: Fetching/Updating Profile Attributes

PWMEAttributeManager customizes notifications to users' preferences. The PWMEAttributeManager is the central point for fetching and updating attributes associated with the device or user. You use an instance of this class to fetch and update profile and user attributes.

```
// Get Profile attribute meta data.
  [[PWMEAttributeManager sharedManag
fetchProfileAttributeMetadataWithCor
*error) {

  }];

// Get profile attributes.
  [[PWMEAttributeManager sharedManag
fetchProfileAttributesWithCompletior
*error) {

  }];

// Update profile attributes.
  [[PWMEAttributeManager sharedManag
updateProfileAttributes:<#userattrib

  }];
```

Step 5: Listening for and Receiving Notifications

The application can listen to the following to be notified of Message Read/Modified notifications:

- PWMEReceiveMessageNotificationKey
- PWMEDeleteMessageNotificationKey
- PWMEReadMessageNotificationKey

The application can be notified of Geofence events like entry/exits. Add/Remove and Modify Geofence notifications by subscribing to the following keys.

- PWMEAddGeoZonesNotificationKey
- PWMEDeleteGeoZonesNotificationKey
- PWMEEnterGeoZoneNotificationKey
- PWMEExitGeoZoneNotificationKey

```
// Adding an Observer
[[NSNotificationCenter defaultCenter
selector:@selector(didEnterZoneNotif
name:PWMEEnterGeoZoneNotificationKey

// Removing an Observer
[[NSNotificationCenter defaultCenter
name:PWMEEnterGeoZoneNotificationKey

- (void)didEnterZoneNotification:(NS
    NSString *identifier = notificat
    id<PWMEZone> msgZone;
    for (id<PWMEZone> zone in [PWEng
       if ([zone.identifier isEqual
           msgZone = zone;
       }
    }
   UIAlertView *alert = [[UIAlertVie
message:[zone identifier] delegate:n
otherButtonTitles:nil];
   [alert show];
}
```

Deep Linking (Option)

The Message object has metadata property which may consist of multiple key-value pairs that allow for deep link to different sections of the application (Ex: Launch Image, Initial screen, app launch behavior etc.)

By default a notification launches the App and goes to the HOME page. Metadata deep linking can direct the notification to open elsewhere in the app or to web URL.

```
[PWEngagement didReceiveRemoteNotif:
withNotificationHandler:^(PWMELocalN
        if (notification) {
            // Deep linking
   NSDictionary *metaData = notifica
   NSString *className = [metaData v
   Class class = NSClassFromString(c
   //display the view controller
        }
 }];
```

Promotions Handling (Option)

The campaign can be setup to have promotions configured as html pages. The promotions are loaded in a web-view.

```
UIWebView *webView;
PWMEZoneMessage *message; //message
NSString *promotionInformation = mes

[webView loadHTMLString:promotionInf
```