Android Mapping and Location SDKs - Getting Started and Integration Guide

This documentation is no longer actively supported and may be out of date. Going forward, please visit and bookmark our new site (https://docs.phunware.com/) for up-to-date documentation.

(v 3.1.1 and Newer of Location SDK-Android)

This guide provides instructions for getting started with the Location SDK and integrating Location SDK for routing. It is only applicable for users of the Android Location SDK v 3.1.1 and newer.

*If you are using version 3.0.0 - 3.1.0 of the Android Location SDK, view Integrating a Location Provider

*If you are using a version of the Location SDK for Android earlier than 3.0.0 contact Phunware Support for SDK assistance (support@phunware.com).

Android Mapping SDK Installation - Getting Started

This guide is a quick start to adding a Phunware Map to an Android app. Android Studio is the recommended development environment for building an app with the Phunware Mapping SDK.

Step 1- Add the Phunware Maven remote repository.

Insert this block into `allprojects -> repositories`:

```
allprojects {
    repositories {

  maven {
      url
"https://nexus.phunware.com/content/groups/publ
ic/"
  }
    }
}
```

Step 2 - Add the Mapping SDK as a dependency in your app's `build.gradle` file

This one line includes all dependencies necessary to use the Phunware Mapping SDK.

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

android {
 ...
}

dependencies {
 ...
    compile
'com.phunware.mapping:mapping:3.1.2'
    ...
}
```

Step 3 - Add your Google Maps Api Key to `Android Manifest.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.android.geo.API_KEY"

android:value="@string/google_maps_api_key"/>
```

Step 4 - Add Phunware keys for App Id, Access Key, Signature Key and Encryption Key

Your App Id, Access Key, Signature Key and Encryption Key are found on the MaaS portal on the Applications tab.

See: Config Guides (Core)

```
<meta-data
android:name="com.phunware.APPLICATION_ID"
android:value="@string/app_id" />
<meta-data
android:name="com.phunware.ACCESS_KEY"
android:value="@string/access_key" />
<meta-data
android:name="com.phunware.SIGNATURE_KEY"
android:value="@string/signature_key" />
<meta-data
android:name="com.phunware.ENCRYPTION_KEY"
android:value="@string/encrypt_key" />
```

Step 5 - Add MapFragment to your layout

This is where the Phunware Mapping SDK will render the map.

NOTE: the fragment will be found via the *R.id.map* id.

```
<fragment
xmlns:android="http://schemas.android.com/apk/r
es/android"

android:name="com.phunware.mapping.MapFragment"
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
```

Step 6 - Get the Map asynchronously in your activity:

The **PhunwareMapManager** is created and the Phunware API keys are registered in this step.

Then the MapFragment is located and the map is loaded asynchronously.

Your implementation of **OnPhunware MapReadyCallback** will be called when the map is ready.

```
private PhunwareMapManager mapManager;

    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);

  ...

        mapManager =
PhunwareMapManager.create(this);

        // register the Phunware API keys

PwCoreSession.getInstance().registerKeys(this);

        MapFragment mapFragment = (MapFragment)
getFragmentManager()
                .findFragmentById(R.id.map);
        if (mapFragment != null) {

mapFragment.getPhunwareMapAsync(this);
        }
    }
```

Step 7 - Make sure your activity implements `OnPhunwareMapReadyCallback`

Once the **PhunwareMap** is ready, set it in the **PhunwareMapManager** and add a building by id.

Building additions are also asynchronous, so a callback will notify you of success or failure.

```
public class MainActivity extends
AppCompatActivity implements
OnPhunwareMapReadyCallback {

 ...

    @Override
    public void onPhunwareMapReady(PhunwareMap
phunwareMap) {
        mapManager.setPhunwareMap(phunwareMap);


mapManager.addBuilding(getResources().getIntege
r(R.integer.building_id),
                new Callback<Building>() {
            @Override
            public void onSuccess(Building
building) {
            }

            @Override
            public void onFailure(Throwable
throwable) {
            }
        });
    }
```

Step 8 - Do something interesting with the Building

Once the map and building are loaded, move and zoom the view so you can see the initial floor.

```
public class MainActivity extends
AppCompatActivity implements
OnPhunwareMapReadyCallback {
    @Override
    public void onPhunwareMapReady(final
PhunwareMap phunwareMap) {
        mapManager.setPhunwareMap(phunwareMap);


mapManager.addBuilding(getResources().getIntege
r(R.integer.building_id),
                new Callback<Building>() {
            @Override
            public void onSuccess(Building
building) {
                FloorOptions initialFloor =
building.initialFloor();

building.selectFloor(initialFloor.getLevel());
                // animate the camera to the
building at an appropriate zoom level
                // so we can see the building
                CameraUpdate cameraUpdate =
CameraUpdateFactory

.newLatLngBounds(initialFloor.getBounds(), 4);

phunwareMap.getGoogleMap().animateCamera(camera
Update);
            }
            @Override
            public void onFailure(Throwable
throwable) {
            }
        });
    }
```

Integrating a Managed Provider

Managed Providers offer a combination of signal location providers. This combination results in higher accuracy rates for location in dynamic way finding.

Phunware Managed Providers:

Provider	Settings/Keys	Description
Cisco Beacon Point	• Confidence Factor • Mobile SDK Key	Cisco vBLE (virtual bluetooth low energy) location provider

Mist	• Confidence Factor • Mobile SDK Key	Mist vBLE (virtual bluetooth low energy) location provider

Step 1 - Add the Phunware Maven remote repository. Insert this block into `allprojects` -> `repositories`:

```
allprojects {
    repositories {
        maven {
            url
"https://nexus.phunware.com/content/groups/publ
ic/"
        }
    }
}
```

Step 2 - Add Phunware key resources to strings.xml for App Id, Access Key, Signature Key

Navigate to portal, find your app, and add your access key and signature key to setup your application.

```
<string name="app_Id">APPID</string>
<string name="access_Key">ACCESSKEY</string>
<string name="sig_Key">SIGKEY</string>
```

Step 3 - Add Phunware keys for App Id, Access Key, and Signature Key to Manifest

Note that encryption key may not be provided in portal under your app's settings. If it isn't, you may leave it empty.

```
<meta-data
    android:name="com.phunware.APPLICATION_ID"
    android:value="@string/app_id"/>
<meta-data
    android:name="com.phunware.ACCESS_KEY"
    android:value="@string/access_key"/>
<meta-data
    android:name="com.phunware.SIGNATURE_KEY"
    android:value="@string/signature_key"/>
<meta-data
    android:name="com.phunware.ENCRYPTION_KEY"
    android:value="@string/encrypt_key"/>
```

Step 4 - Add managed provider as a compile dependency.

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

dependencies {
 ...
 compile 'com.phunware.mapping:mapping:3.1.2'
 compile
'com.phunware.location:provider-managed:3.1.0'
 ...
}
```

Step 5 - Set the Location Provider on the PhunwareMapManager

It's important to note that Managed Provider must be built and set on my MapManager after the the map is ready and the building has been loaded.

There are a few values that will be needed to build a ManagedProvider, they are as follows:

- application: reference to the users application
- context: Context of the application
- buildingId: Id of the building you currently have loaded

```
@Override
public void onPhunwareMapReady(final
PhunwareMap phunwareMap) {
    mapManager.setPhunwareMap(phunwareMap);
    this.phunwareMap = phunwareMap;

    mapManager.addBuilding(buildingId,
venueGuid,
            new Callback<Building>() {
                @Override
                public void onSuccess(Building
building) {
                    if (building == null) {

Toast.makeText(MainActivity.this, "No
building", Toast.LENGTH_LONG)
                                .show();
                        return;
                    }

                    // ManagedProvider must be
set after the map is loaded

ManagedProviderFactory.ManagedProviderFactoryBu
ilder builder =
                            new
ManagedProviderFactory.ManagedProviderFactoryBu
ilder();

builder.application(getApplication())
                            .context(new
WeakReference<Context>(getApplication()))

.buildingId(String.valueOf(buildingId));
                    ManagedProviderFactory
factory = builder.build();
                    PwManagedLocationProvider
provider = (PwManagedLocationProvider) factory

.createLocationProvider();

mapManager.setLocationProvider(provider,
building);
      ...
```

Step 6 - Enable Location Updates

Enable location updates after we have the map and building as well as after setting the provider

```
@Override
public void onPhunwareMapReady(final
PhunwareMap phunwareMap) {
 ...
    mapManager.addBuilding(buildingId,
venueGuid,
            new Callback<Building>() {
                @Override
                public void onSuccess(Building
building) {
     ...
     // enable my location (blue dot) after
setting the location provider
     mapManager.setMyLocationEnabled(true);

     }

    }

}
```

Step 7 - Manage Location Updates when in the Background

In order to ensure that we handle lifecycles correctly we must stop requesting location updates when we background, and begin requesting them when we are in the foreground.

```
@Override
protected void onPause() {
    super.onPause();


 // If you have permission to access location
and you don't have a building then proceed

 if (canAccessLocation() && building == null) {
     if (mapFragment != null) {
         mapFragment.getPhunwareMapAsync(this);
     }
 }
    if (mapManager != null) {
        mapManager.setMyLocationEnabled(false);
    }
}

@Override
protected void onResume() {
    super.onResume();

    if (mapManager != null) {
        mapManager.setMyLocationEnabled(true);
    }
}
```