# Wherami Android SDK Tutorial

## Usage

Step1: Configure the SDK:
There are two parameters:

- `CMS_HOST` The domain/IP hosting the latest dataset for localization
- `SITE_NAME` The name of the site the dataset to be loaded for localization

```
Client.Configure(CMS_HOST, SITE_NAME, getApplicationContext());
```

For example

**HKUST_v2**

```
Client.Configure("https://dy199-079.ust.hk", "HKUST", getApplicationContext());
```

Step 2a: Get the fingerprint version:

```
Client.GetDataVersion()
```

If null is returned, it must download the fingerprint to continue (See Step 3)
Step 2b: Check if any fingerprint update is available

```
Client.CheckDataUpdate(new Client.DataUpdateQueryCallback() {
            @Override
            public void onQueryFailed(Exception e) {
                  //Query failed
            }

            @Override
            public void onUpdateAvailable(String newVersion) {
                  //Update is available
            }

            @Override
            public void onLatestVersion() {
                  //Already on latest version
            }
      }, getApplicationContext());
```

Step 3: Update the fingerprint data if needed:

```
Client.UpdateData(new Client.DataUpdateCallback() {
      @Override
      public void onProgressUpdated(int percentage) {
            //Update progress
      }

      @Override
      public void onCompleted() {
            //Update completed
      }

      @Override
      public void onFailed(Exception e) {
            //Update failed
      }
   }, getApplicationContext());
```

Step 4a: Start localization in UI-less mode (with optional UI)
In this mode, the engine is managed by the application itself.

```
//Construct a MapEngine which will exist for the whole application lifetime
MapEngine engine = MapEngineFactory.Create(getApplicationContext());

//Alternatively construct a MapEngine to run as foreground service
MapEngine engine = MapEngineFactory.Create(getApplicationContext(),
MapEngineFactory.MODE_FOREGROUND_SERVICE);

//Initialize the engine
engine.initialize();

//Implement the LocationUpdateCallback and pass to engine to receive location
update.
engine.attachLocationUpdateCallback(this);

//Implement the PoiChangeCallback and pass to the engine to receive POI change
update.
engine.attachPoiChangeCallback(this);

//Start the engine
engine.start();
```

A UI can be created optionally by adding MapFragment programmatically:

```
FragmentManager fragMan = getFragmentManager();
FragmentTransaction fragTransaction = fragMan.beginTransaction();
//Pass the engine instance to the MapFragment
MapFragment mapFragment = MapFragment.newInstance(engine);
fragTransaction.add(destLayout.getId(),mapFragment, "mapFragment").commit();
```

//To stop localization, call:
engine.stop();

Step 4b: Start localization in UI mode
In this mode the MapFragment manages MapEngine, i.e. the MapEngine stops positioning once the MapFragment is not active. This provides a simple mean for localization feature drop-in by including the MapFragment. However the localization stops as the fragment is detached or paused.

Firstly, include the MapFragment in the Activity XML or programmatically:

```
FragmentManager fragMan = getFragmentManager();
FragmentTransaction fragTransaction = fragMan.beginTransaction();
//Not passing the engine instance to the MapFragment
MapFragment mapFragment = MapFragment.newInstance();
fragTransaction.add(destLayout.getId(),mapFragment, "mapFragment").commit();
```

The hosting Activity should implement the MapFragment.MapEngineReadyCallback

```
public void onMapEngineReady(MapEngine engine){
//Attach the necessary callbacks
//Implement the LocationUpdateCallback and pass to engine to receive location
update.
engine.attachLocationUpdateCallback(this);
//Implement the PoiChangeCallback and pass to the engine to receive POI change
update.
engine.attachPoiChangeCallback(this);
}
```