Lecture #12
Awareness and Nearby

Mobile Applications 2018-2019

Happy New Year!

2019


LIN
С РОЖДЕСТВОМ ХРИСТОВЫМ

Google Awareness API

Google Awareness API

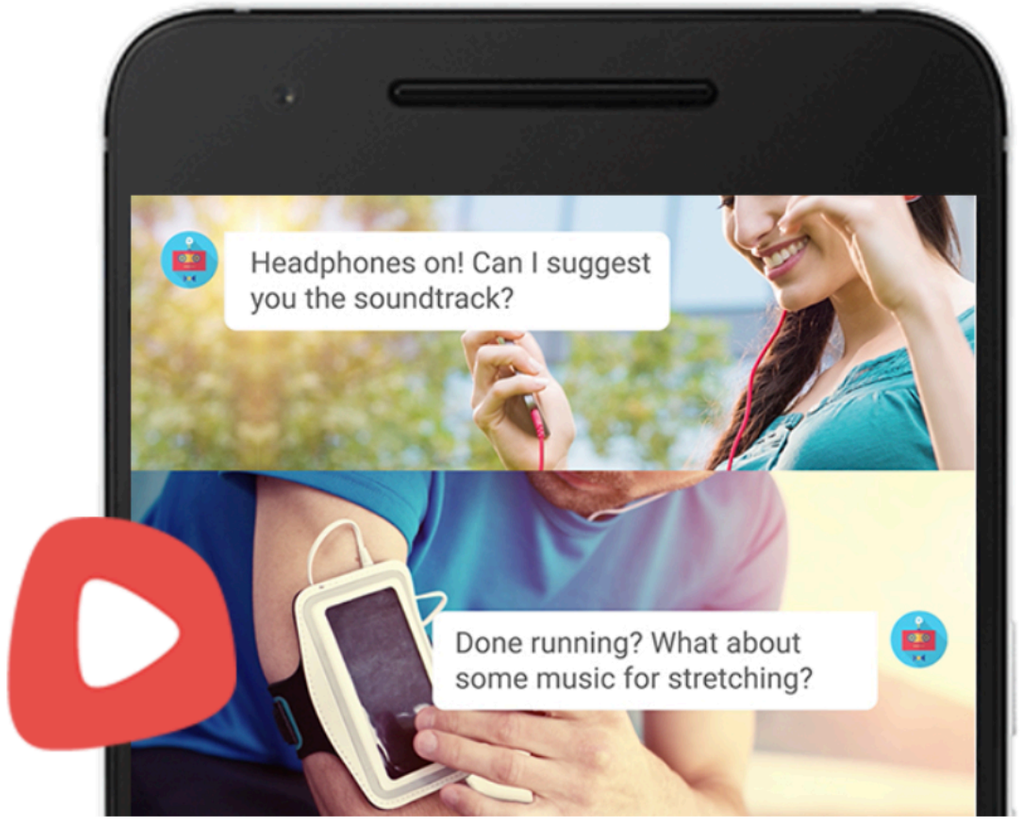
- Part of Google Play Services.

Google Awareness API

- Part of Google Play Services.
- Many signals, one API.

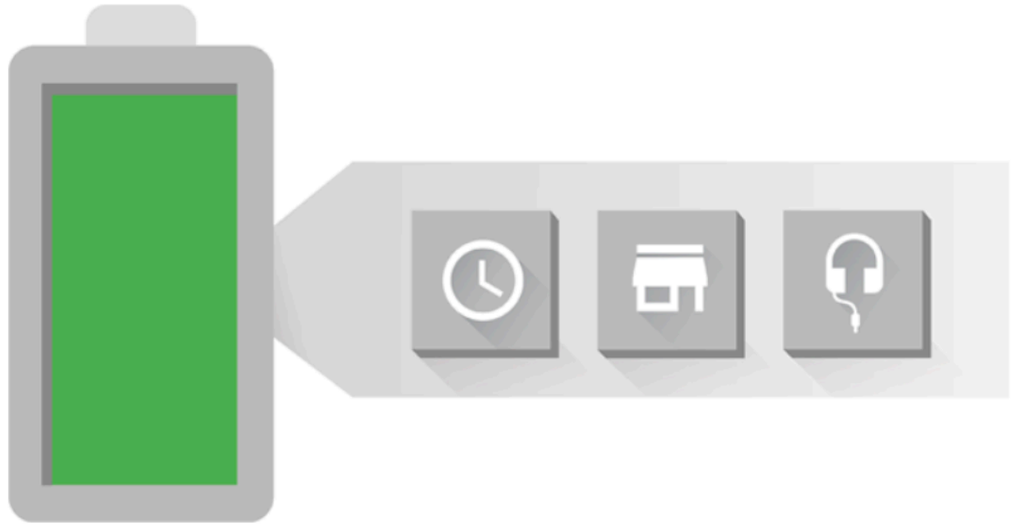
Google Awareness API

- Part of Google Play Services.
- Many signals, one API.
- High quality data.



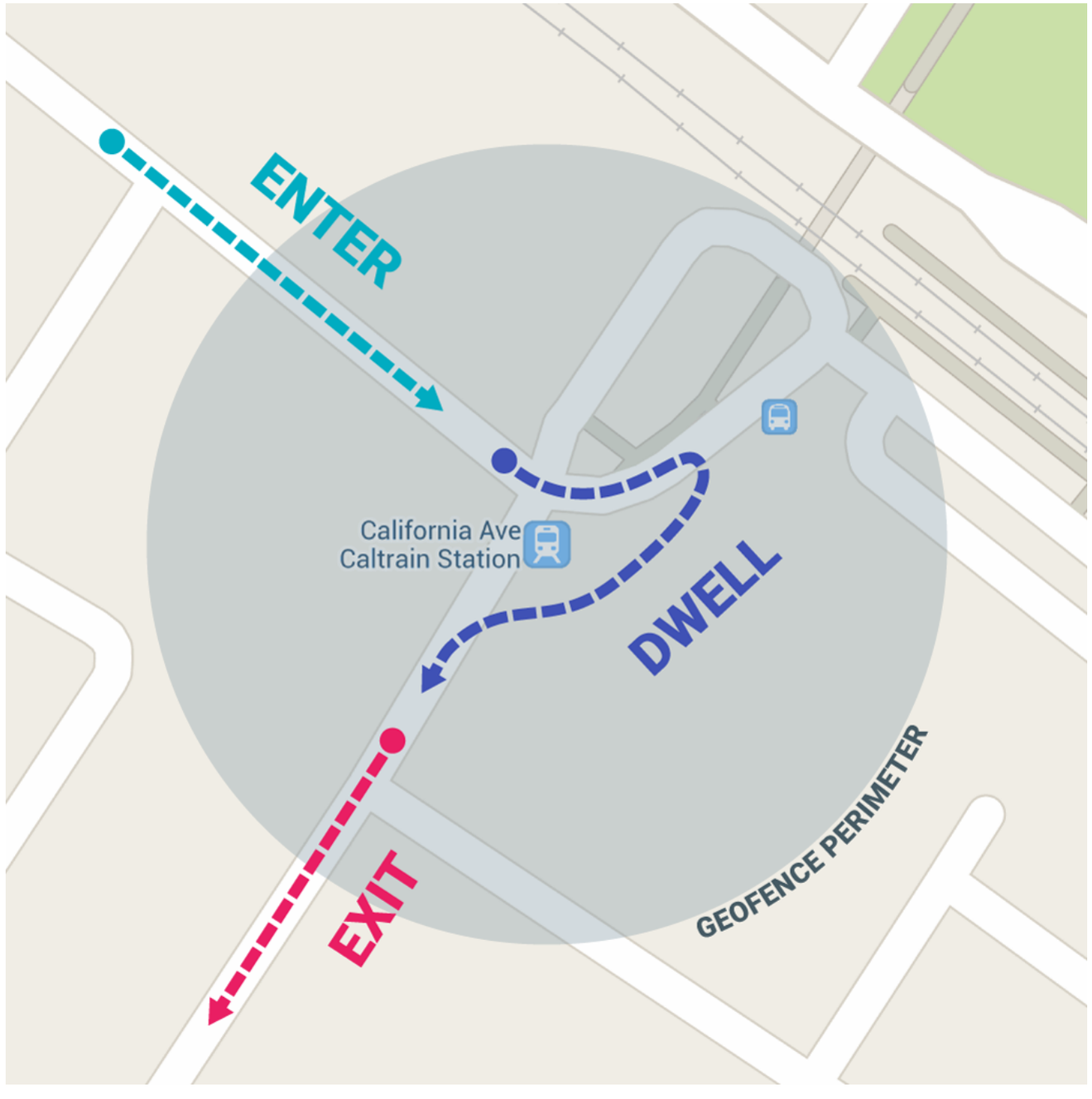
Google Awareness API

- Part of Google Play Services.
- Many signals, one API.
- High quality data.
- Smart battery savings.



Fence API

Fence API



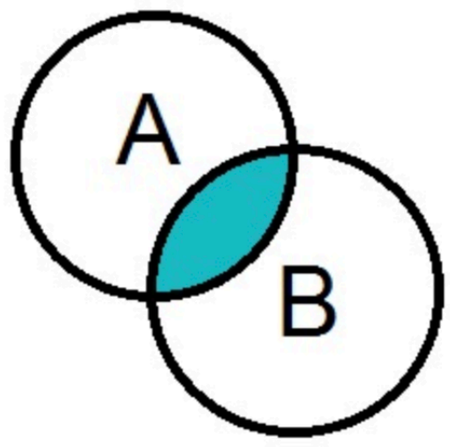
https://developer.android.com/training/location/geofencing

Fence API

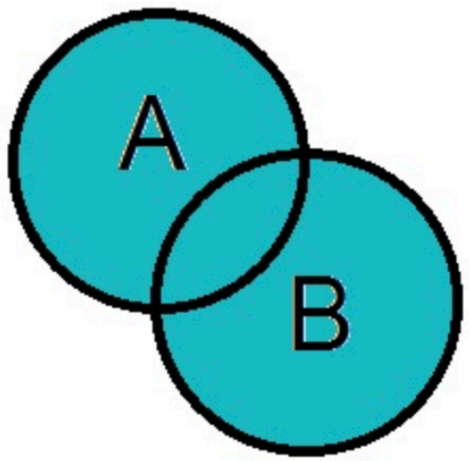
- The user's current location (lat/lng).
- The user's current activity (walking, driving, etc.).
- Device-specific conditions, such as whether, the headphones are plugged in.
- Proximity to nearby beacons.

Fence API

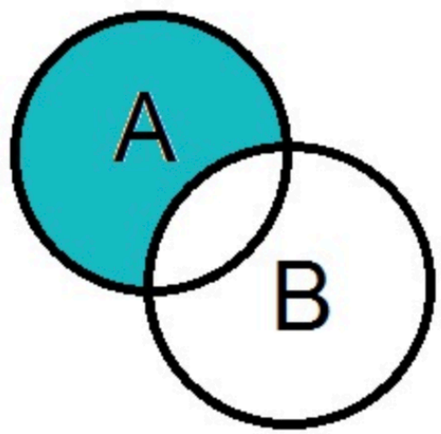
- User plugs in headphones and starts walking.
- User enters a 100-meter geofence before 5 p.m. on a weekday.
- User enters range of a specific BLE beacon.



A AND B



A OR B



A NOT B

Get an API key from the Google Developers Console

- Go to the Google Developers Console.
- Select a project, or create a new one.
- Click Continue to enable the Awareness API.
- On the Credentials page, create an Android key and set the API credentials.
- In the create key dialog, restrict your usage to Android apps — enter your app's SHA-1 fingerprint and package name.
- Click Create.

Get an API key from the Google Developers Console

- Go to the Google Developers Console.
- Select a project, or create a new one.
- Click Continue to enable the Awareness API.
- On the Credentials page, create an Android key and set the API credentials.
- In the create key dialog, restrict your usage to Android apps — enter your app's SHA-1 fingerprint and package name.
- Click Create.

```
AIzaSyBdVl-cTICSwYKrZ95LoVuw7dbMuDt1KG0
```

Configuration

```
<application>
  ...
  <meta-data
      android:name="com.google.android.awareness.API_KEY"
      android:value="AIzaSyBdVl-cTICSwYKrZ95LoVuw7dbMuDt1KG0"/>
</application>
```

Configuration

```
<application>
  ...
  <meta-data
      android:name="com.google.android.awareness.API_KEY"
      android:value="AIzaSyBdVl-cTICSwYKrZ95LoVuw7dbMuDt1KG0"/>
</application>
```

```
android.content.Context context;
GoogleApiClient client = new GoogleApiClient.Builder(context)
        .addApi(Awareness.API)
        .build();
client.connect();
```

Fence API

```
implementation 'com.google.android.gms:play-services-awareness:16.0.0'
```

Fence API

```
    implementation 'com.google.android.gms:play-services-awareness:16.0.0'
```

```
val walkingFence = DetectedActivityFence.during(DetectedActivityFence.WALKING)
```

Fence API

```
// Declare variables for pending intent and fence receiver.
private var mPendingIntent: PendingIntent? = null
private var mFenceReceiver: FenceReceiver? = null

// Initialize myPendingIntent and fence receiver in onCreate().
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  // ...
  val intent = Intent(FENCE_RECEIVER_ACTION)
  mPendingIntent = PendingIntent.getBroadcast(this@MainActivity, 0, intent, 0)
  mFenceReceiver = FenceReceiver()
  registerReceiver(mFenceReceiver, IntentFilter(FENCE_RECEIVER_ACTION))
  // ...
}
```

Fence API

```
// Register the fence to receive callbacks.
// The fence key uniquely identifies the fence.
Awareness.getFenceClient(this).updateFences(
  FenceUpdateRequest.Builder()
    .addFence(FENCE_KEY, walkingFence, mPendingIntent!!)
    .build()
)
.addOnSuccessListener { Log.i(TAG, "Fence was successfully registered.") }
.addOnFailureListener { e → Log.e(TAG, "Fence could not be registered: $e") }
```

Fence API

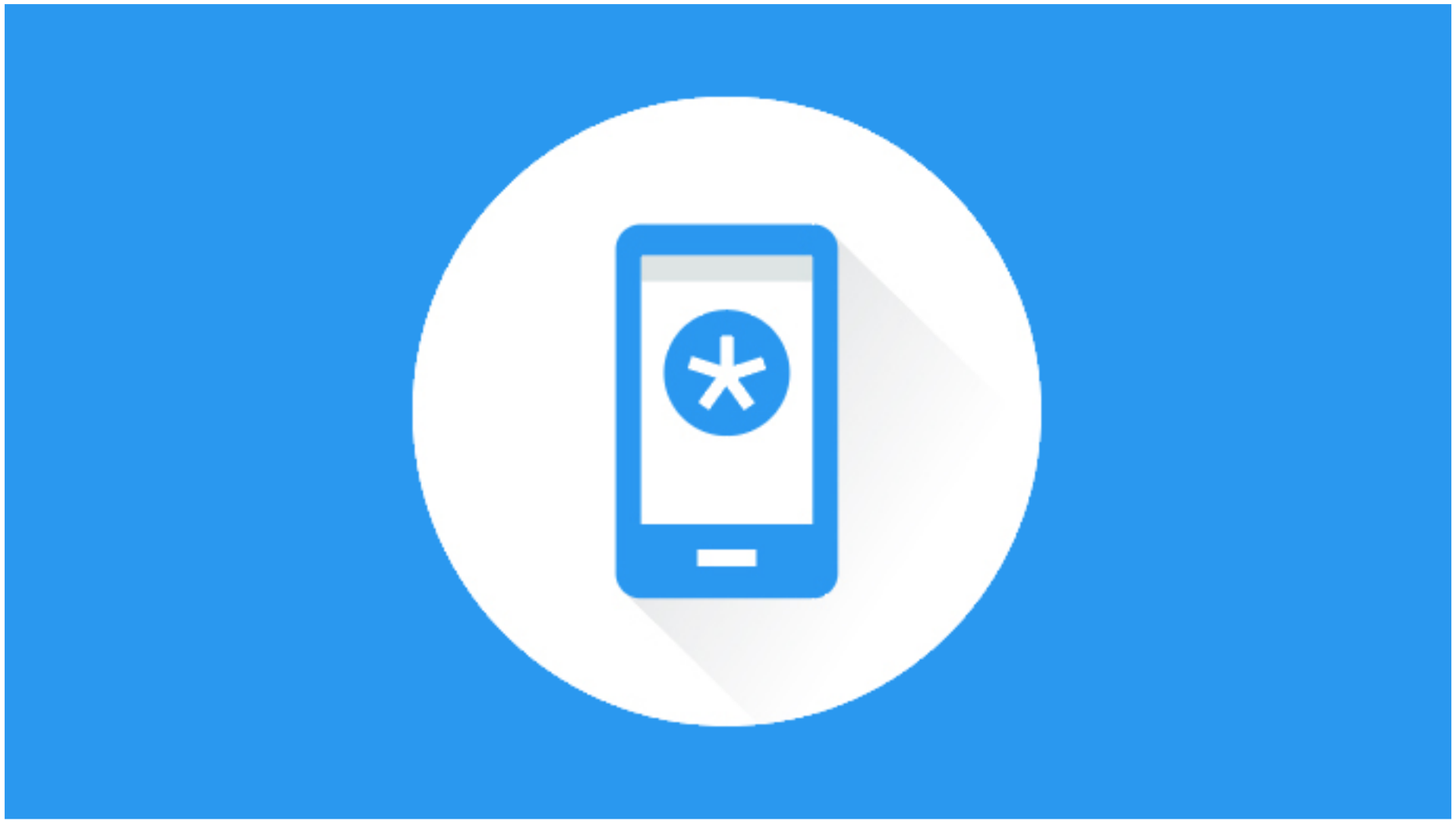
```
/**
 * A basic BroadcastReceiver to handle intents from from the Awareness API.
 */
inner class FenceReceiver : BroadcastReceiver() {
  override fun onReceive(context: Context, intent: Intent) {
    if (!TextUtils.equals(FENCE_RECEIVER_ACTION, intent.action)) {
        mLogFragment.logView
            .println("Received an unsupported action in FenceReceiver:
                    action=${intent.action!!}")
     return
    }
    // The state information for the given fence is em
    val fenceState = FenceState.extract(intent)

    if (TextUtils.equals(fenceState.fenceKey, FENCE_KEY)) {
      val fenceStateStr: String = when (fenceState.currentState) {
        FenceState.TRUE → "true"
        FenceState.FALSE → "false"
        FenceState.UNKNOWN → "unknown"
        else → "unknown value"
      }
```

```
/**
 * A basic BroadcastReceiver to handle intents from from the Awareness API.
 */
inner class FenceReceiver : BroadcastReceiver() {
  override fun onReceive(context: Context, intent: Intent) {
    if (!TextUtils.equals(FENCE_RECEIVER_ACTION, intent.action)) {
       mLogFragment.logView
           .println("Received an unsupported action in FenceReceiver:
                     action=${intent.action!!}")
     return
    }
    // The state information for the given fence is em
    val fenceState = FenceState.extract(intent)

    if (TextUtils.equals(fenceState.fenceKey, FENCE_KEY)) {
      val fenceStateStr: String = when (fenceState.currentState) {
        FenceState.TRUE → "true"
        FenceState.FALSE → "false"
        FenceState.UNKNOWN → "unknown"
        else → "unknown value"
      }
      mLogFragment.logView.println("Fence state: $fenceStateStr")
    }
  }
}
```


```
/**
 * A basic BroadcastReceiver to handle intents from from the Awareness API.
 */
inner class FenceReceiver : BroadcastReceiver() {
  override fun onReceive(context: Context, intent: Intent) {
    if (!TextUtils.equals(FENCE_RECEIVER_ACTION, intent.action)) {
       mLogFragment.logView
           .println("Received an unsupported action in FenceReceiver:
                     action=${intent.action!!}")
     return
    }
    // The state information for the given fence is em
    val fenceState = FenceState.extract(intent)

    if (TextUtils.equals(fenceState.fenceKey, FENCE_KEY)) {
      val fenceStateStr: String = when (fenceState.currentState) {
        FenceState.TRUE → "true"
        FenceState.FALSE → "false"
        FenceState.UNKNOWN → "unknown"
        else → "unknown value"
      }
      mLogFragment.logView.println("Fence state: $fenceStateStr")
    }
  }
}
```

https://developers.google.com/awareness/android-api/fence-api-overview

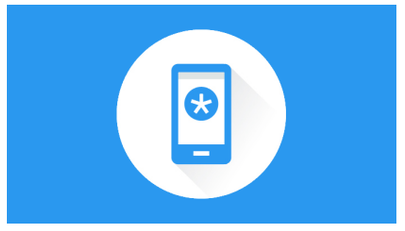
Snapshot API



Snapshot API

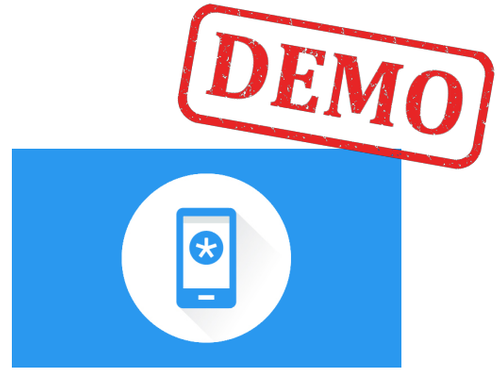
- Detected user activity, such as walking or driving.
- Nearby beacons that you have registered.
- Headphone state (plugged in or not).
- Location, including latitude and longitude.
- Place where the user is currently located.
- Weather conditions in the user's current location.

Snapshot API



```
Awareness.getSnapshotClient(this).detectedActivity
    .addOnSuccessListener { dar →
        val arr = dar.activityRecognitionResult
        // getMostProbableActivity() is good enough for basic Activity detection.
        // To work within a threshold of confidence,
        // use ActivityRecognitionResult.getProbableActivities() to get a list of
        // potential current activities, and check the confidence of each one.
        val probableActivity = arr.mostProbableActivity

        val confidence = probableActivity.confidence
        val activityStr = probableActivity.toString()
        mLogFragment.logView.println("Activity: $activityStr,
                                      Confidence: $confidence/100")
    }
.addOnFailureListener { e → Log.e(TAG, "Could not detect activity: $e") }
```

Snapshot API




```
Awareness.getSnapshotClient(this).detectedActivity
   .addOnSuccessListener { dar →
     val arr = dar.activityRecognitionResult
     // getMostProbableActivity() is good enough for basic Activity detection.
     // To work within a threshold of confidence,
     // use ActivityRecognitionResult.getProbableActivities() to get a list of
     // potential current activities, and check the confidence of each one.
     val probableActivity = arr.mostProbableActivity

     val confidence = probableActivity.confidence
     val activityStr = probableActivity.toString()
     mLogFragment.logView.println("Activity: $activityStr,
                                   Confidence: $confidence/100")
  }
.addOnFailureListener { e → Log.e(TAG, "Could not detect activity: $e") }
```

https://developers.google.com/awareness/android-api/snapshot-api-overview

Nearby Messages API

- A Publish-Subscribe API.

Nearby Messages API

Nearby Messages API

- Bluetooth.

Nearby Messages API

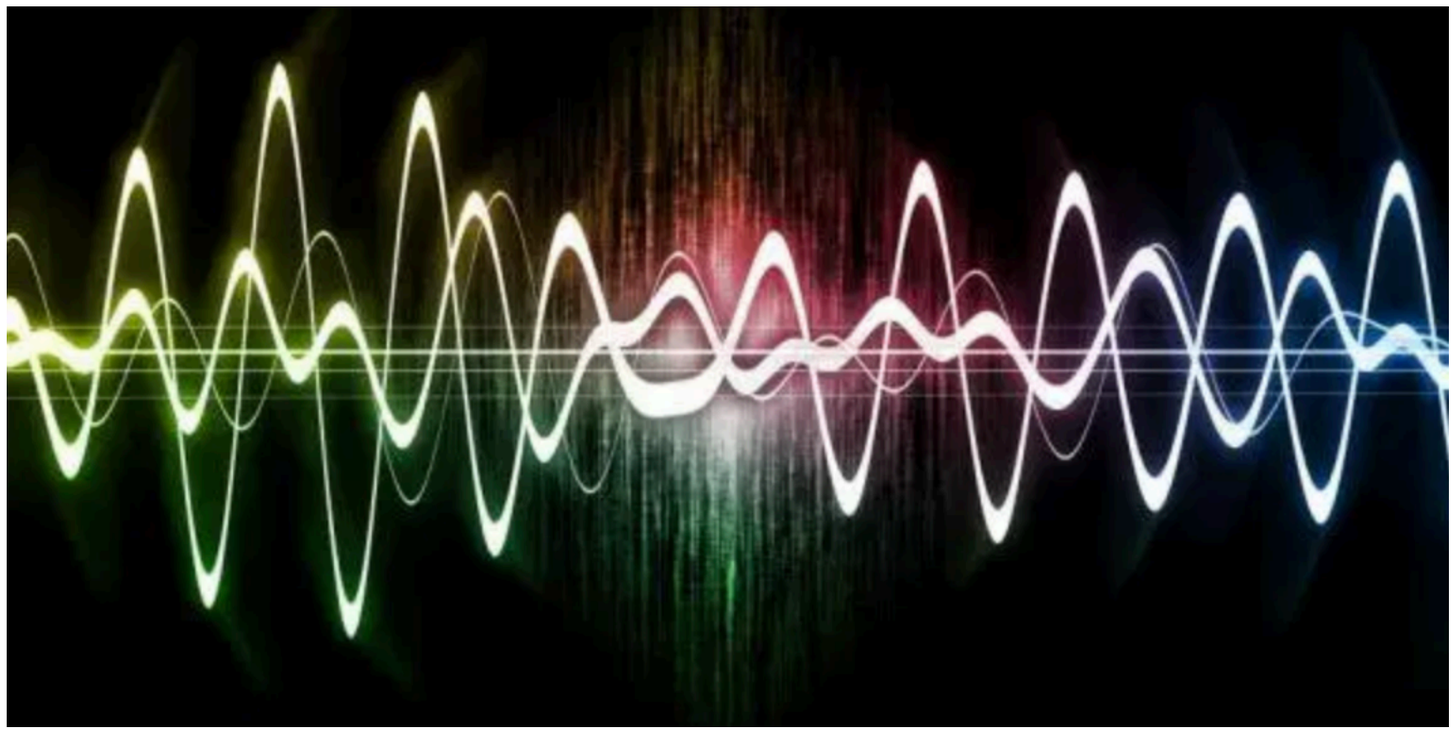
- Bluetooth.
- Bluetooth Low Energy.

Nearby Messages API

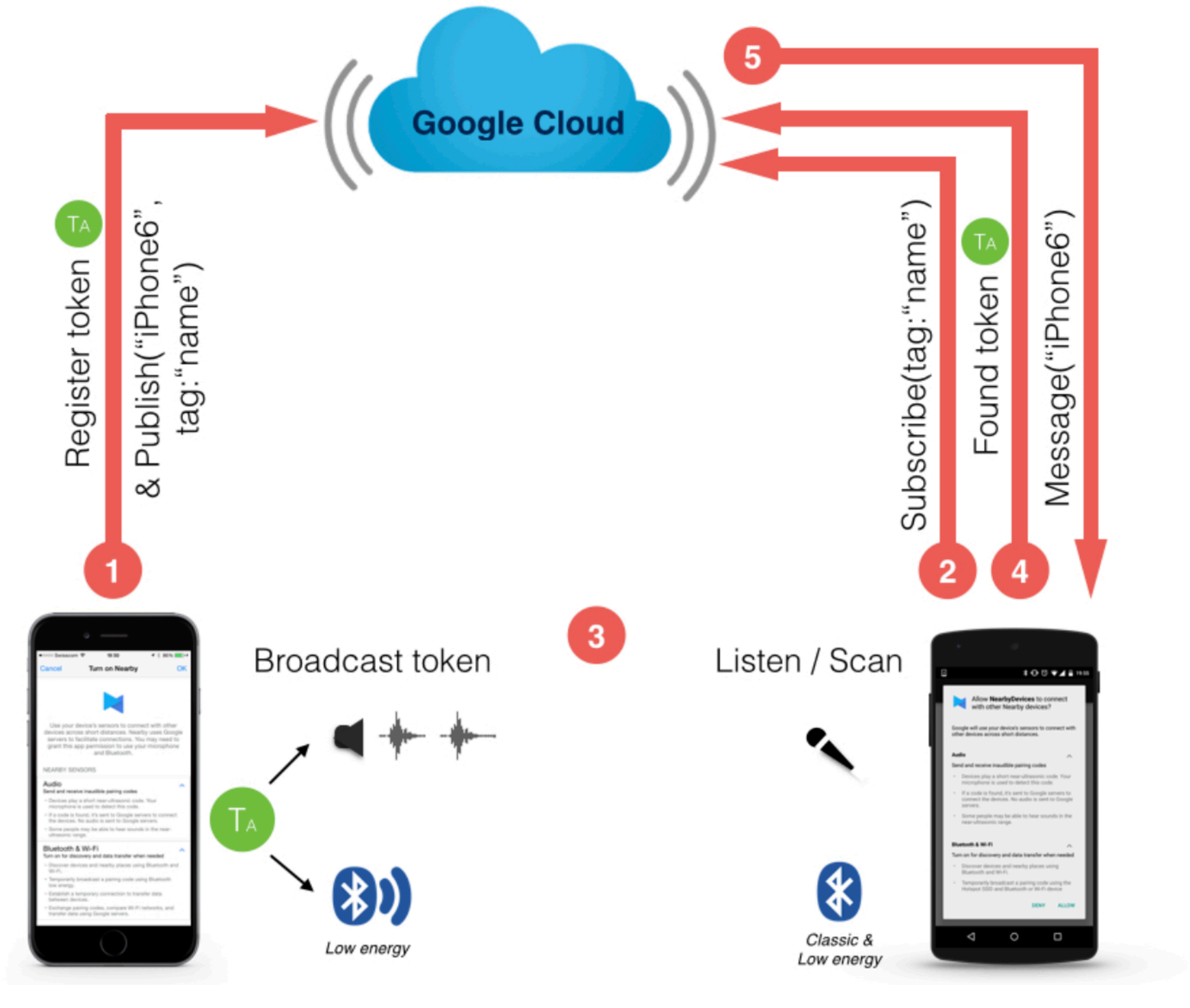
- Bluetooth.
- Bluetooth Low Energy.
- Wi-Fi.

Nearby Messages API

- Bluetooth.
- Bluetooth Low Energy.
- Wi-Fi.
- Near-ultrasonic audio.



Nearby Messages API



Configuration

```
apply plugin: 'android'
...
dependencies {
  compile 'com.google.android.gms:play-services-nearby:16.0.0'
}
```

Configuration

```
apply plugin: 'android'
...

dependencies {
  compile 'com.google.android.gms:play-services-nearby:16.0.0'
}
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.sample.app" >
    <application ...>
        <meta-data
            android:name="com.google.android.nearby.messages.API_KEY"
            android:value="API_KEY" />
        <activity>
        ...
        </activity>
    </application>
</manifest>
```

Publish and Subscribe

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ...
  mMessageListener = new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.d(TAG, "Found message: " + new String(message.getContent()));
    }
    @Override
    public void onLost(Message message) {
      Log.d(TAG, "Lost sight of message: " + new String(message.getContent()));
    }
  }

  mMessage = new Message("Hello World".getBytes());
}

@Override
public void onStart() {
  super.onStart();
  ...
  Nearby.getMessagesClient(this).publish(mMessage);
  Nearby.getMessagesClient(this).subscribe(mMessageListener);
```

```
  ...
  mMessageListener = new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.d(TAG, "Found message: " + new String(message.getContent()));
    }

    @Override
    public void onLost(Message message) {
      Log.d(TAG, "Lost sight of message: " + new String(message.getContent()));
    }
  }

  mMessage = new Message("Hello World".getBytes());
}
```

```
@Override
public void onStart() {
  super.onStart();
  ...
  Nearby.getMessagesClient(this).publish(mMessage);
  Nearby.getMessagesClient(this).subscribe(mMessageListener);
}

@Override
public void onStop() {
  Nearby.getMessagesClient(this).unpublish(mMessage);
  Nearby.getMessagesClient(this).unsubscribe(mMessageListener);
  ...
  super.onStop();
}
```

Handling User Consent

```
if (ContextCompat.checkSelfPermission(this,
    Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
    mMessagesClient = Nearby.getMessagesClient(this,
       new MessagesOptions.Builder()
     .setPermissions(NearbyPermissions.BLE)
     .build());
}
```

Get Beacon Messages


```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ...
  mMessageListener = new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.d(TAG, "Found message: " + new String(message.getContent()));
    }

    @Override
    public void onLost(Message message) {
      Log.d(TAG, "Lost sight of message: " + new String(message.getContent()));
    }
  }
}

// Subscribe to receive messages.
private void subscribe() {
  Log.i(TAG, "Subscribing.");
  SubscribeOptions options = new SubscribeOptions.Builder()
    .setStrategy(Strategy.BLE_ONLY)
    .build();
  Nearby.getMessagesClient(this).subscribe(mMessageListener, options);
}
```

Get Beacon Messages

Subscribe in the background

```
// Subscribe to messages in the background.
private void backgroundSubscribe() {
    Log.i(TAG, "Subscribing for background updates.");
    SubscribeOptions options = new SubscribeOptions.Builder()
            .setStrategy(Strategy.BLE_ONLY)
            .build();
    Nearby.getMessagesClient(this).subscribe(getPendingIntent(), options);
}

private PendingIntent getPendingIntent() {
    return PendingIntent.getBroadcast(this, 0, new Intent(this,
        BeaconMessageReceiver.class), PendingIntent.FLAG_UPDATE_CURRENT);
}
```

Get Beacon Messages

Subscribe in the background

```
@Override
public void onReceive(Context context, Intent intent) {
  Nearby.getMessagesClient(context).handleIntent(intent, new MessageListener() {
    @Override
    public void onFound(Message message) {
      Log.i(TAG, "Found message via PendingIntent: " + message);
    }

    @Override
    public void onLost(Message message) {
      Log.i(TAG, "Lost message via PendingIntent: " + message);
    }
  });
}
```

https://developers.google.com/nearby/messages/android/get-beacon-messages

Lecture outcomes

- Create applications to be aware of multiple aspects of a users context, while managing battery and memory health.
- React to changes in the user's environment.
- Get instant details about the user's current environment, by accessing 7 signals from one simple API surface.
- Create nearby notifications to helps users discover what's around them.