HUAWEI Health

Obtain User Behaviour and Health data effortlessly

Hello!

I am Anil Ghimire

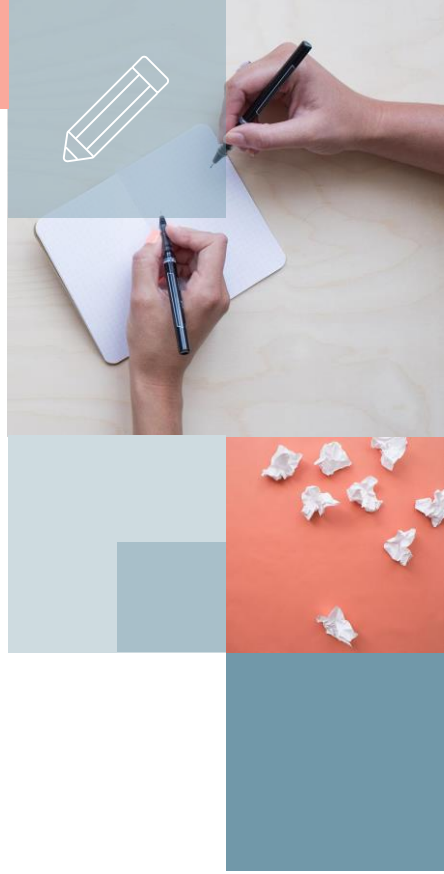
Developer Technical Support Engineer (DTSE)

You can find me at https://www.linkedin.com/in/anil-ghimire/

Background

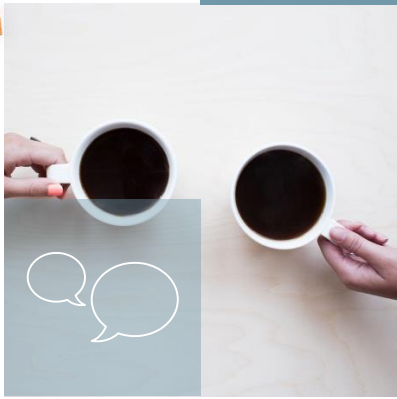
Contents

- mHealth and Health APIs
- HUAWEI Health kit Overview
- Awareness Kit Overview
- Practical Use Cases
- Sample Codes



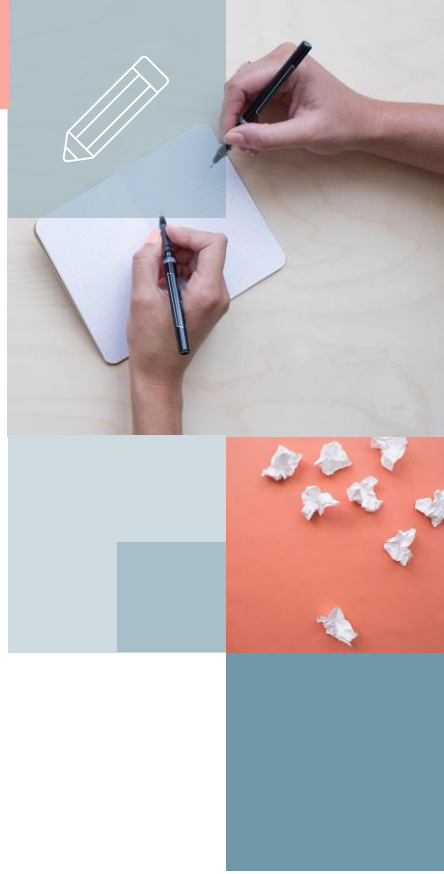
mHealth


Health
5,202
steps
40
min
Exercise records
22/03 Yoga
79 kcal
Heart
23/03 Heart rate
71 bpm
Sleep
23/03
56 min
Stress
23/03
30 Normal



mHealth

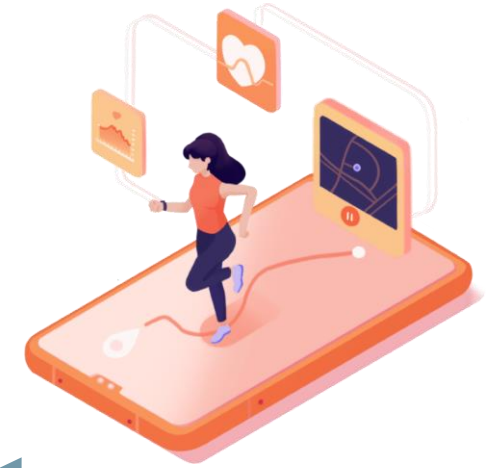
- Monotiring and sharing health info using smart devices and health tracking apps.
- Smartphones are hub for additional sensors.
- Smart watches are best additional sensors.
- Wearable sensors has endless possibilities.

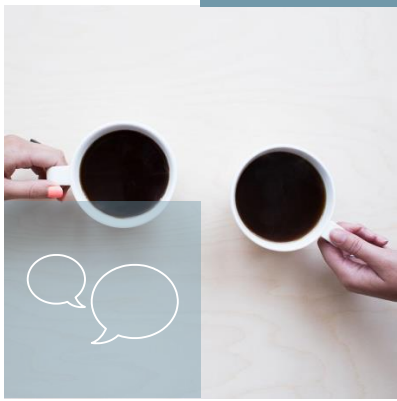


Health APIs

- Fitness trackers and wearables gather enormous amount of data.
- Need for a platform to effectively use the data.
- Industry giants Google, Apple, Samsung, Huawei has platforms to manage the health data.

HUAWEI Health Kit



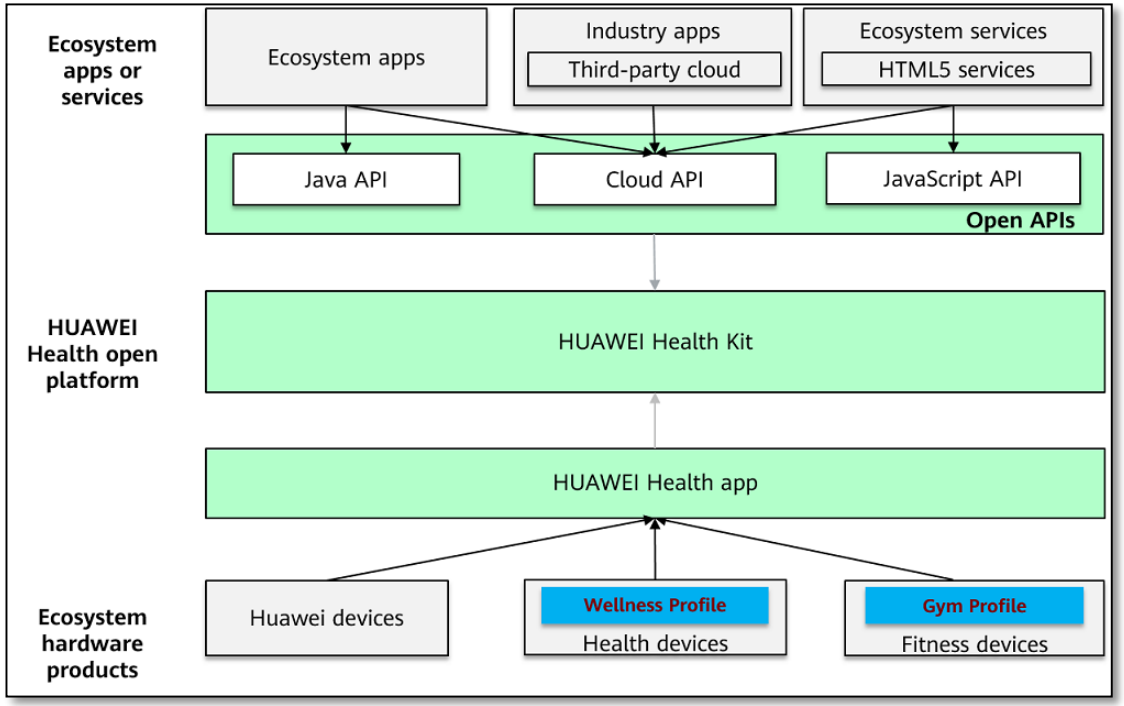


Overview

HUAWEI Health kit connects the hardware devices and ecosystem apps to provide

- **Consumers:** Health care, workout guidance and ultimate service experience.
- **Developers:** a open data platform and API capabilities to upload and use data.

Overview


Ecosystem apps or services
Ecosystem apps
Industry apps
Third-party cloud
Ecosystem services
HTML5 services
Java API
Cloud API
JavaScript API
Open APIs
HUAWEI Health open platform
HUAWEI Health Kit
HUAWEI Health app
Ecosystem hardware products
Huawei devices
Wellness Profile
Health devices
Gym Profile
Fitness devices

Security and Privacy

- Data Security Management
- User Authorization management
- Privacy Compliance Management

Basic Concept

Data Type

Health Kit provides standard defination of data types for easy understanding of the sampling data.

Data Controller

It is used to represent the raw data collection source, a unique identifier is generated when data collector is created.

Data Sampling Point

Sample of specific type of data collected by specific Data collector at specific moment.

Sampling Dataset

Is a container for storing sampling points that come from same Data Controller.

Activity Records

It records the user activity details within a period of time, consisting of the activity type, app information, time period, etc.

Health Records

It records the user health details within a period of time, health record type, data controller, etc.

Basic Health Capabilities

Basic health capabilities provides atomic data openness using which the app can add, delete, modify, and query.

Scopes

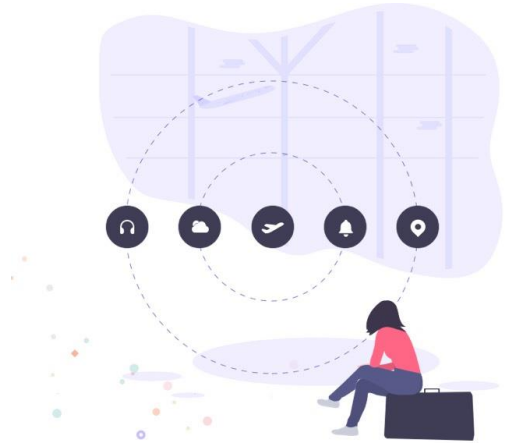
- **Personal Data:** Height, weight.
- **Health data:** Sleep, Heart rate, Nutrition, Stress.
- **Fitness Data:** Step, Distance, Speed, Calories, Strength, Activity.
- **Restricted Data:** Blood Glucose, Blood Pressure, Oxygen Saturation.
- **Activity Record**

Extended Health Capabilities

With Extended health Capability, you can obtain health data and activity data from the Huawei Health app or write your data to the app.

- Open Data types
  - **Health data:** Sleep, Heart Rate, Body fat.
  - **Restricted Data type:** Blood Oxygen, Body temperature.
  - **Health Solution:** Gender, DOB, height, weight, Real-time heart rate.
  - **Activity Solution:** Step, distance, calories, intensity, walking, running, cycling, Real-time activity data.
  - **Device Information:** Device name, device model and more.

Awareness Kit



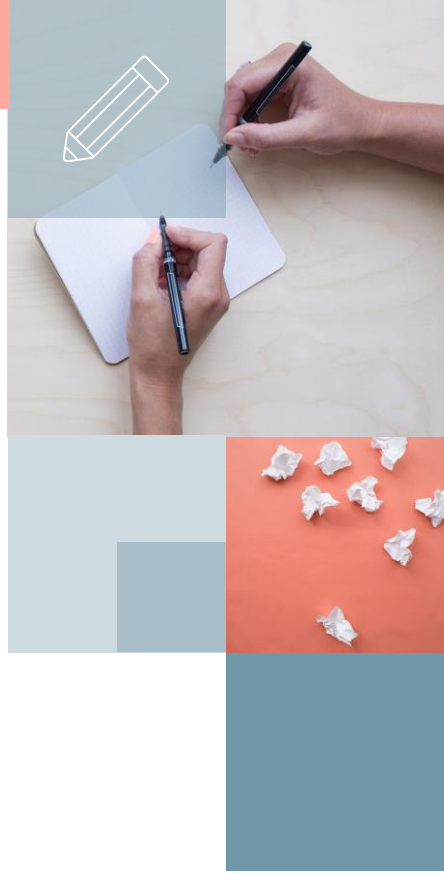
Overview

- **Capture**

  Obtains user conditions, such as the time, location, weather, behavior (walking/running/driving), headset status (plugged or not), beacons (registered or connected), light intensity, and car Bluetooth status (connected or not).

- **Barrier**

  Notifies your app when a user meets preset conditions (such as entry into an area and length of stay)



Awareness Capabilities

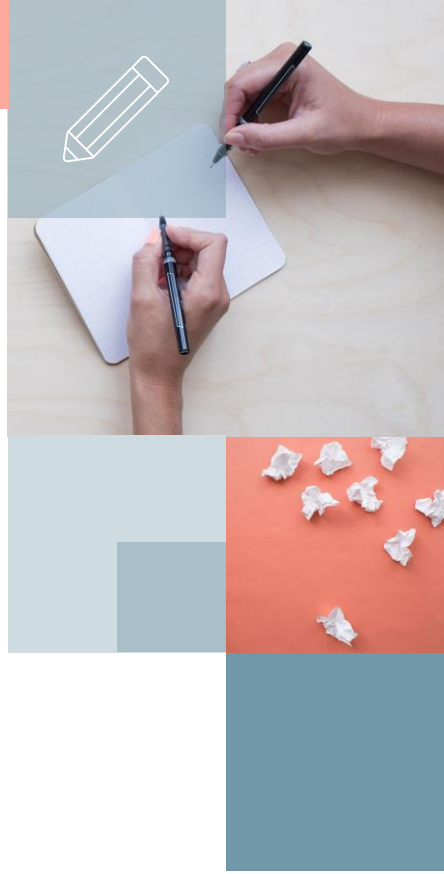
- **Headset:** Connected / Disconnected.
- **Behavior:** Staying Still, Walking, Running, Cycling or Driving.
- **Location:** Fused Location, Geofence.
- **Time:** Working day, Holiday, Weekend, Morning, Afternoon, Night, Sunrise and Sunset.
- **Weather:** Temperature, Humidity, Wind direction, wind power.

Wearable Development

Wearable Sensors Capabilities

- Step Counter
- Barometer
- Heart Rate
- Body State

https://developer.harmonyos.com/en/docs/documentation/doc-guides/device-sensors-guidelines-0000001050199987



Development Process

Preparation

- ApplyHuawei ID
- Apply Health Kit*

Development

- Integrate HMS Core
- Develop app using Android studio / DevEco Studio.

Verification

- Submit for release verification of the health service.*

Release

- Submit app to review for release.

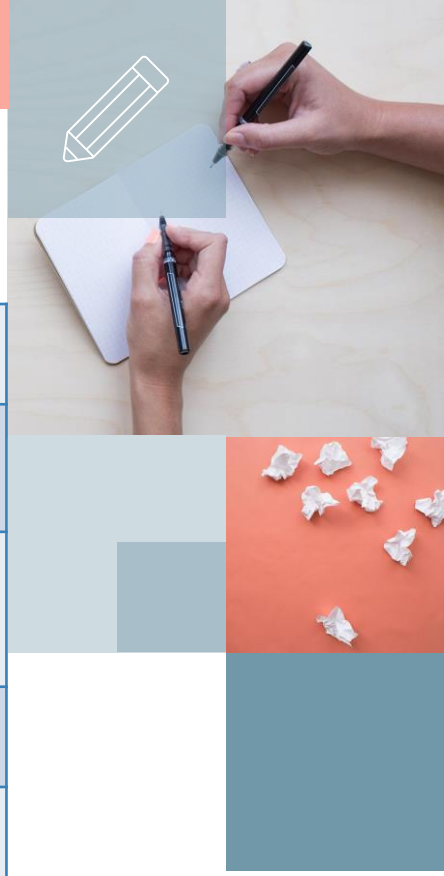
Note:
* Only for Health Kit

Development Environment

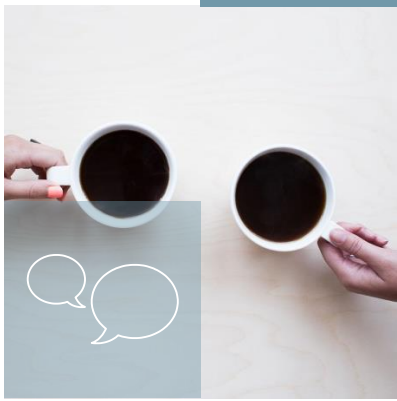
Kits		OS support	HMS Core / SDK version
Health Kit	○ Basic Capabilities	○ Android 6.0+ ○ EMUI 5.0+	○ HMS core: 4.2.0.300+ ○ SDK: 6.4.0.301 ○ Huawei Health app: 11.0.0.512+
	○ Extended Capabilities	○ EMUI 6.0+	○ HMS core: 5.1.0.300+ ○ SDK: 6.3.0.300 ○ Huawei Health app: 11.0.0.512+
Awareness Kit	○ Time, Audio, Ambient, Screen, Wi-Fi, Weather	○ Android 7.0+ ○ EMUI 5.0+	○ HMS core: 4.0.2.300+
	○ Dark mode	○ Android 10.0+ ○ EMUI 10.0+	
	○ App, Location, Beacon	○ EMUI 5.0+	

Development Platforms

Platforms	OS support
Java/ Kotlin	• Android
REST APIs	• Web Services
JavaScript APIs	• HarmonyOS
Reactive Native, Cordova, Xamarin, Flutter	• Only Basic Health Capabilities
Harmony OS (Wearables)	• Only Basic Health Capabilities



Practical Use cases And Sample Codes



Use-cases: Links

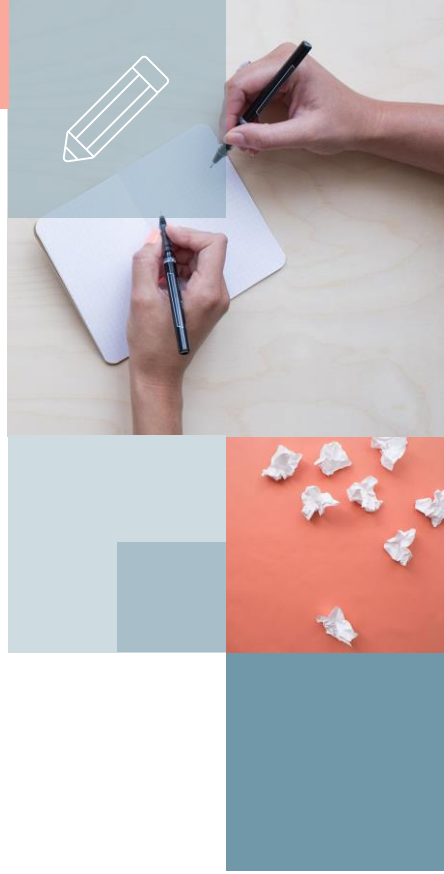
- **Step Count of the day:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/steps-delta-scene-0000001050822049
- **Running Activity Records:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/write-sports-recording-scene-0000001050782024
- **Reading real-time step count:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/real-time-steps-data-0000001054876912
- **Reading Sleep:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/read-client-sleep-scene-0000001091427304
- **Heart rate Alert:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/client-basketball-write-scene-0000001144924115
- **Querying Daily Activities:** **https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/extended-querying-dailyactivity-0000001053253403**
- **Querying Health Data:** **https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/extended-querying-singleworkout-record-0000001053333367**

Sample Codes

Sample Codes	Use Cases
Request Authorization from User	• Needed to get authorization from app user.
Data Controller; Read and Write Data	• Read and Write data from Huawei Health Kit.
Activity Record Controller	• Perform Activity data in Health Kit
Health Record	• Perform Health Data in Health Kit
Subscribing Health Data	• Subscribing the health data
Awareness Kit	• Getting User behavior

Request Authorization from User

```
private void requestAuthorization() {
    // Add scopes to apply for. The following only shows an example. You need to add scopes according to your specific needs.
    String[] scopes = new String[]{
            // View and store the step count in Health Kit.
            Scopes.HEALTHKIT_STEP_READ, Scopes.HEALTHKIT_STEP_WRITE,
            // View and store the height and weight in Health Kit.
            Scopes.HEALTHKIT_HEIGHTWEIGHT_READ, Scopes.HEALTHKIT_HEIGHTWEIGHT_WRITE,
            // View and store the heart rate data in Health Kit.
            Scopes.HEALTHKIT_HEARTRATE_READ, Scopes.HEALTHKIT_HEARTRATE_WRITE
    };

    // Obtain the intent of the authorization process.
    // The value true indicates that the authorization process of the Health app is enabled,
    // and false indicates that the authorization process is disabled.
    Intent intent = mSettingController.requestAuthorizationIntent(scopes, b: true);

    // Open the authorization process screen.
    Log.i(TAG, msg: "start authorization activity");
    startActivityForResult(intent, REQUEST_AUTH);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Process only the response result of the authorization process.
    if (requestCode == REQUEST_AUTH) {
        // Obtain the authorization response result from the intent.
        HealthKitAuthResult result = mSettingController.parseHealthKitAuthResultFromIntent(data);
        if (result == null) {
            Log.w(TAG, msg: "authorization fail");
            return;
        }

        if (result.isSuccess()) {
            Log.i(TAG, msg: "authorization success");
        } else {
            Log.w(TAG, msg: "authorization fail, errorCode:" + result.getErrorCode());
        }
    }
}
```

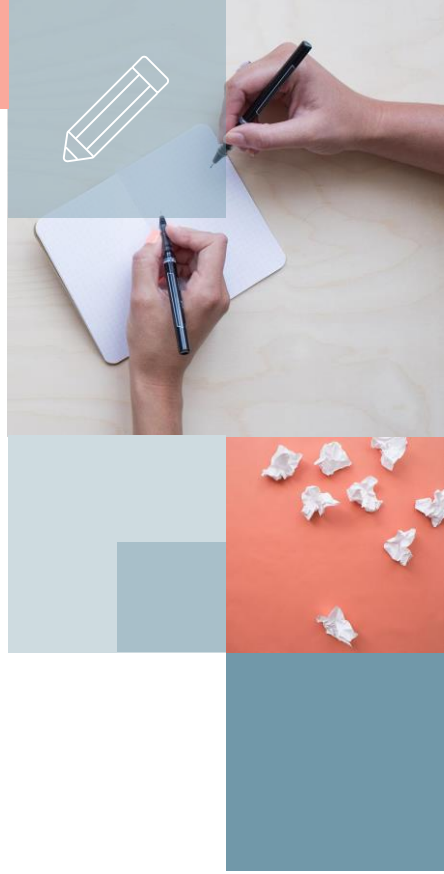


Data Collector: Initialize

```
// Step 1: Build a DataCollector object.
DataCollector dataCollector = new DataCollector.Builder().setPackageName(getApplicationContext())
        .setDataType(DataType.DT_CONTINUOUS_STEPS_DELTA)
        .setDataStreamName("STEPS_DELTA")
        .setDataGenerateType(DataCollector.DATA_TYPE_RAW)
        .build();

// Step 2: Create a sampling dataset based on the data collector.
final SampleSet sampleSet = SampleSet.create(dataCollector);

// Step 3: Build the start time, end time, and incremental step count for a DT_CONTINUOUS_STEPS_DELTA sampling point.
SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd hh:mm:ss");
Date startDate = null;
Date endDate = null;
try {

    startDate = dateFormat.parse( source: "2020-03-17 09:00:00");
    endDate = dateFormat.parse( source: "2020-03-17 09:05:00");
} catch (ParseException e) {
    Log.w(TAG,  msg: "Time parsing error");
}
```

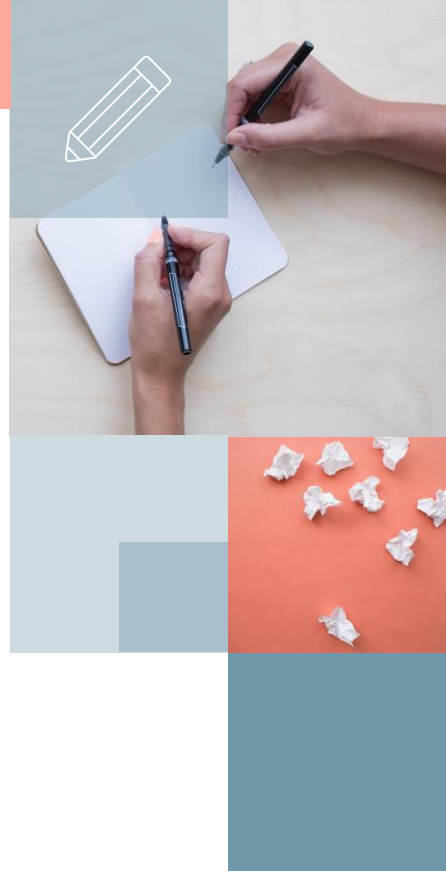
Data Controller : insert data

```
// Step 4: Build a sampling point for steps.
SamplePoint samplePoint = sampleSet.createSamplePoint()
        .setTimeInterval(startDate.getTime(), endDate.getTime(), TimeUnit.MILLISECONDS);
samplePoint.getFieldValue(Field.FIELD_STEPS_DELTA).setIntValue(stepsDelta);

// Step 5: Save a DT_CONTINUOUS_STEPS_DELTA sampling point to the sampling dataset.
sampleSet.addSample(samplePoint);

// Step 6: Call the insert API to insert the sampling dataset into Health Kit.
Task<Void> insertTask = dataController.insert(sampleSet);

// Step 7: Calling the insert API to insert the sampling dataset is an asynchronous operation.
// Therefore, a listener needs to be registered to monitor whether the data insertion is successful or not.
insertTask.addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void result) {
        Log.w(TAG,  msg: "Success insert a SampleSet into HMS core");
        //showSampleSet(sampleSet);
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(Exception e) {
        String errorCode = e.getMessage();
        String errorMsg = HiHealthStatusCodes.getStatusCodeMessage(Integer.parseInt(errorCode));
        Log.w(TAG,  msg: errorCode + ": " + errorMsg);
    }
});
```



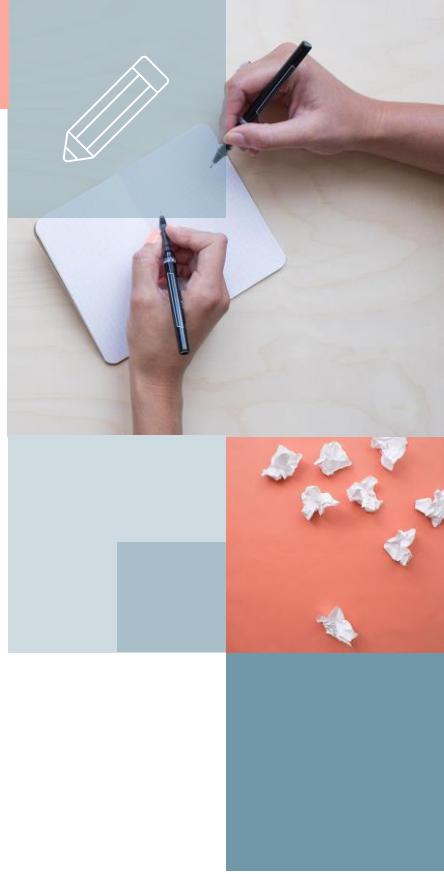
Data Controller: Read Data

```
private void getintensityData(DataController dataController) {


    SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd hh:mm:ss");
    Date startDate = null;
    Date endDate = null;
    try {
        startDate = dateFormat.parse( source: "2021-05-30 00:00:00");
        endDate = dateFormat.parse( source: "2021-05-31 23:59:00");
    } catch (ParseException exception) {
        logger( string: "Time parsing error");
    }


    ReadOptions readOptions = new ReadOptions.Builder()
            .read(DataType.DT_CONTINUOUS_EXERCISE_INTENSITY_V2)
            .setTimeRange(startDate.getTime(), endDate.getTime(), TimeUnit.MILLISECONDS)
            .build();

    Task<ReadReply> readReplyTask = dataController.read(readOptions);

    readReplyTask.addOnSuccessListener(new OnSuccessListener<ReadReply>() {
        @Override
        public void onSuccess(ReadReply readReply) {
            logger( string: "Success read a SampleSets from HMS core");
            for (SampleSet sampleSet : readReply.getSampleSets()) {
                showSampleSet(sampleSet);
            }
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(Exception e) {
            String errorCode = e.getMessage();
            String errorMsg = HiHealthStatusCodes.getStatusCodeMessage(Integer.parseInt(errorCode));
            logger( string: errorCode + ": " + errorMsg);
        }
    });
}
```



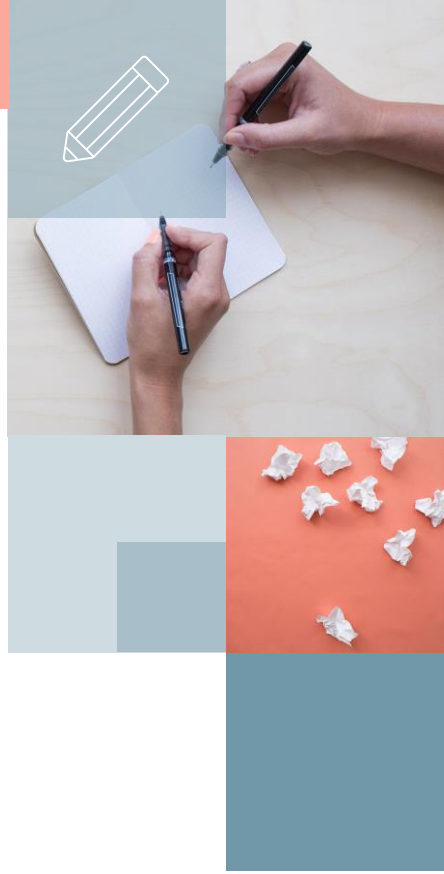
Activity Record

```
HiHealthOptions hiHealthOptions = HiHealthOptions.builder()
        .addDataType(DataType.DT_CONTINUOUS_EXERCISE_INTENSITY_V2, HiHealthOptions.ACCESS_READ)
        .addDataType(DataType.DT_CONTINUOUS_ACTIVITY_SEGMENT, HiHealthOptions.ACCESS_READ)
        .addDataType(DataType.DT_CONTINUOUS_EXERCISE_INTENSITY_V2, HiHealthOptions.ACCESS_READ)
        .build();

AuthHuaweiId signInHuaweiId = HuaweiIdAuthManager.getExtendedAuthResult(hiHealthOptions);
DataController dataController = HuaweiHiHealth.getDataController(getApplicationContext(), signInHuaweiId);
final ActivityRecordsController activityRecordsController = HuaweiHiHealth.getActivityRecordsController(getApplicationContext(), signInHuaweiId);

 private void getActivityRecordData(ActivityRecordsController activityRecordsController) {

     // 1. Build the time range of the request object: start time and end time.
     Calendar cal = Calendar.getInstance();
     Date now = new Date();
     cal.setTime(now);
     long endTime = cal.getTimeInMillis();
     cal.add(Calendar.DAY_OF_YEAR, i1: -1);
     long startTime = cal.getTimeInMillis();

     // 2. Build the request body for reading activity records and set the request time range.
     ActivityRecordReadOptions readOption =
             new ActivityRecordReadOptions.Builder().setTimeInterval(startTime, endTime, TimeUnit.MILLISECONDS)
                     .readActivityRecordsFromAllApps()
                     .build();

     // 3. Call getActivityRecord to obtain the activity records on the Health platform based on the request body.
     Task<ActivityRecordReply> getTask = activityRecordsController.getActivityRecord(readOption);
     getTask.addOnSuccessListener(new OnSuccessListener<ActivityRecordReply>() {
         @Override
         public void onSuccess(ActivityRecordReply activityRecordReply) {
             Log.i("ActivityRecords", "Get ActivityRecord was successful!");

         }
     }).addOnFailureListener(new OnFailureListener() {
         @Override
         public void onFailure(Exception e) {
             String errorCode = e.getMessage();
             String errorMsg = HiHealthStatusCodes.getStatusCodeMessage(Integer.parseInt(errorCode));
             Log.i("ActivityRecordSample", errorCode + ": " + errorMsg);
         }
     });

 }
```



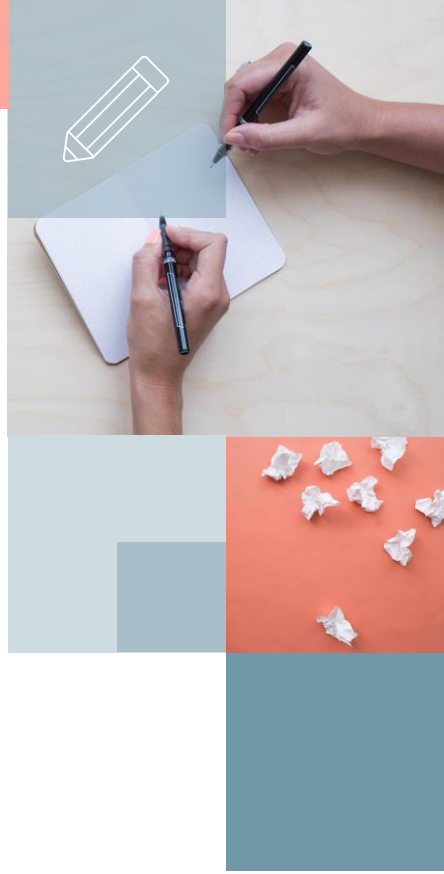
Subscribing Health Data

```
// Construct a 10,000-step target mode.
DataReportModel modelStepTarget = new DataReportModel(HiHealthPointType.DATA_POINT_STEP_SUM,
   CharacteristicConstant.ReportType.TARGET.getReportTypeValue(), 10000);

// Result callback.
HiRealTimeCallback hiRealTimeCallback = new HiRealTimeCallback() {
    @Override
    public void onResult(int errCode, String message) {
        if (errCode == HiHealthError.SUCCESS) {
            Log.i(TAG, "success");
        }
    }

    @Override
    public void onDataChanged(Bundle bundle) {
        // Notify the subscriber when the total number of steps reaches 10,000 (the actual number of steps
        Log.i(TAG, String.valueOf(bundle.getInt(HiHealthKitConstant.BUNDLE_KEY_STEP)));
    }
};

// Subscribe to step count.
HiHealthDataStore.registerDataAutoReport(context, modelStepTarget, hiRealTimeCallback);

// Cancel the subscription.
HiHealthDataStore.unregisterDataAutoReport(context, modelStepTarget, hiRealTimeCallback);
```



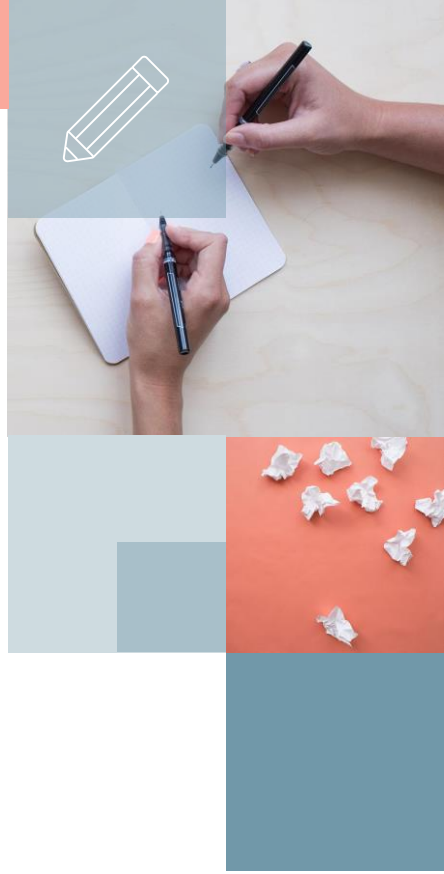
Awareness Kit

```
Awareness.getCaptureClient(this).getBehavior()
        // Callback listener for execution success.
        .addOnSuccessListener(new OnSuccessListener<BehaviorResponse>() {
            @Override
            public void onSuccess(BehaviorResponse behaviorResponse) {
                BehaviorStatus behaviorStatus = behaviorResponse.getBehaviorStatus();
                DetectedBehavior mostLikelyBehavior = behaviorStatus.getMostLikelyBehavior();
                String str = "Most likely behavior type is " + mostLikelyBehavior.getType() +
                        ",the confidence is " + mostLikelyBehavior.getConfidence();
                Log.i(TAG, str);
            }
        })
        // Callback listener for execution failure.
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(Exception e) {
                Log.e(TAG, "get behavior failed", e);
            }
        });
```



Appendix

- **Huawei Health Kit:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/health-introduce-0000001053684429
- **Awareness Kit:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/service-introduction-0000001050031140
- **Huawei Training Videos:** https://developer.huawei.com/consumer/en/training/result?type1=10159055101351300 7&searchTxt=Huawei%20health
- **Huawei Health REST APIs:** https://developer.huawei.com/consumer/en/doc/development/HMSCore-Guides/overview-restful-api-0000001050071695
- **Code Labs:** https://developer.huawei.com/consumer/en/codelabsPortal/carddetails/MyHealth



Thanks!

Any questions?

You can mail me at: anil.ghimire@huawei.com