Programming Guide: Samsung Digital Health

- Health Data

1.2.1
# Table of Contents

1. **OVERVIEW** ........................................................................................................................................... 3
   1.1. HEALTH DATA FRAMEWORK ........................................................................................................ 4
   1.2. ARCHITECTURE ................................................................................................................................. 4
      1.2.1. HEALTH DATA FRAMEWORK ............................................................................................... 5
      1.2.2. SAMSUNG DIGITAL HEALTH APPLICATION .................................................................... 5

2. **DEVELOPMENT ENVIRONMENT** ........................................................................................................ 11
   2.1. PREREQUISITES .............................................................................................................................. 11
   2.2. DOWNLOADING SAMSUNG DIGITAL HEALTH SDK .................................................................... 11
   2.3. S HEALTH BRANDING GUIDELINES ............................................................................................ 12
   2.4. DATA SYNCHRONIZATION DESIGN ............................................................................................ 12
   2.5. CREATING AN APPLICATION ....................................................................................................... 12
   2.6. TESTING YOUR APPLICATION ..................................................................................................... 12
      2.6.1. S HEALTH’S DEVELOPER MODE ....................................................................................... 13
      2.6.2. DATAVIEWER ........................................................................................................................ 14
      2.6.3. CHECKLIST FOR SAMSUNG DIGITAL HEALTH APP .................................................... 14
   2.7. PUBLISHING YOUR APPLICATION ............................................................................................... 15
   2.8. REGISTERING FOR PARTNER APPS ............................................................................................. 15

3. **DATAVIEWER** ................................................................................................................................... 16
   3.1. CHECKING SAVED DATA ............................................................................................................... 16
   3.2. ADDING NEW DATA ....................................................................................................................... 17
   3.3. CONNECTING TO S HEALTH ......................................................................................................... 17
   3.4. EXPORTING DATA ......................................................................................................................... 17
   3.5. SHOWING ON TRACKER TILE ....................................................................................................... 18

4. **HEALTH DATA FUNDAMENTALS** ..................................................................................................... 19
   4.1. IMPORTING LIBRARY ..................................................................................................................... 19
   4.2. HELLO SAMSUNG DIGITAL HEALTH .......................................................................................... 19

5. **HEALTH DATA FEATURES** .............................................................................................................. 24
   5.1. HEALTH DATA STORE ................................................................................................................... 24
      5.1.1. PREPARING DATA ACCESS .................................................................................................. 24
      5.1.2. HEALTH DATA QUERY ....................................................................................................... 25
      5.1.3. FILTER .................................................................................................................................. 26
      5.1.4. DATA NORMALIZATION ........................................................................................................ 26
   5.2. HEALTH DATA TYPE ..................................................................................................................... 27
   5.3. PRIVACY ....................................................................................................................................... 31

6. **SAMPLE APPLICATION - SIMPLEHEALTH** ......................................................................................... 33

7. **SAMPLE APPLICATION - FOODNOTE** ............................................................................................ 35

COPYRIGHT .................................................................................................................................................... 37
1. Overview

Samsung Digital Health SDK helps developers to synchronize health data with S Health 4.x safely and to create useful health applications. The health data store’s integrated health data in S Health can be shared with other S Health’s partner applications based on user’s consent. S Health supports Android devices with KitKat 4.4 including non-Samsung devices.

![Samsung Digital Health service](image)

**Figure 1: Samsung Digital Health service**

Refer to Table 1 for Samsung Digital Health SDK glossary.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>health data framework</td>
<td>It provides useful features to handle user’s health data. As it is included in S Health, applications for Samsung Digital Health works after installing S Health 4.x.</td>
</tr>
<tr>
<td>S Health</td>
<td>An application that helps monitor user’s activities and helps the user makes a healthier life with pedometer, exercise, heart rate, and etc. It can be downloaded from the application market like Google Play or Galaxy Apps. It’s written as <em>italic</em> in this document.</td>
</tr>
</tbody>
</table>

**Table 1: Glossary**

The SDK provides the following packages.

- Health Data package
- S Health Service package
This document contains descriptions for the SDK’s Health Data package. See the S Health Service package’s documents if you want to create a tracker on S Health.

1.1. Health Data Framework

The health data framework of Samsung Digital Health SDK provides the following features:

- Health Data Store
  - Handling the connection state
  - Inserting, reading, updating, deleting health data
  - Unified unit conversion
- Health Data Type
  - Platform-defined and custom data type
- Privacy
  - Granting permission based on user’s consent to read or write the specific data type

1.2. Architecture

The health data framework of Samsung Digital Health is designed to provide safe access for health data and a seamless health service to the user. Figure 2 is the architecture for the health data framework of Samsung Digital Health.

![Figure 2: Architecture for the health data framework of Samsung Digital Health](image)

The detailed description is presented below.
1.2.1. Health Data Framework

The health data framework keeps user’s health data of various data types safely. Health data from a specific source device that has various sensors such as pedometer, accelerometer, or heart rate sensor is inserted based on the unified data unit, read, updated, or deleted through the health data framework. The health data framework is included in S Health and applications can access the health data store’s data of S Health 4.x with the SDK.

1.2.2. Samsung Digital Health Application

A Samsung Digital Health application indicates an application that uses Samsung Digital Health SDK’s APIs. It can synchronize health data with S Health 4.x after approval as the S Health’s partner application. The SDK provides DataViewer that enables you to test your application before approval of the partner application.
Figure 3 shows the relationship between classes and interfaces of the health data package. Detailed descriptions for each class and interface are in the API Reference.
**HealthDataService.initialize()**

It initializes the health data service. APIs of the `com.samsung.android.sdk.healthdata` works properly after calling it without an exception.

**HealthDataStore**

It handles the connection to the data storage of the device. It receives its connection result with `HealthDataStore.ConnectionListener`. Most requests require the connection to the health data store.

**HealthDataResolver**

The health data framework provides classes and interfaces to insert, read, update, or delete health data. `HealthDataResolver` is a central class to handle health data. It sends a data request with related request interfaces and gets a result through `HealthResultHolder`.

The query result can be received immediately with `HealthDataHolder.BaseResult`, `HealthDataResolver.ReadResult` or `HealthDataResolver.AggregateResult`. Or it can be received asynchronously with `HealthResultHolder.ResultListener`. See 5.1 for more information.
Application developers can use platform-defined data types that Samsung Digital Health provides. See 5.2 for more information. Especially `HealthConstants.Common` contains the following mandatory properties for health data:

- Unique ID of health data
- Created and updated time of health data
- Application package name
- Device that provides health data

`HealthConstants.Common` is the base class of predefined data types of Samsung Digital Health as in Figure 4.

**Figure 4: Class diagram for health data types**
User’s health data can be accessed with the user’s consent. Figure 5 shows relationship between classes and interfaces related to HealthPermissionManager. It requests permissions with HealthPermissionManager.PermissionKey that contains the required permission to read or write for the specific health data type. The permission result can be received synchronous or asynchronously. See 5.3 for more information.

Figure 5: Class diagram related to HealthPermissionManager
The Health Data library of Samsung Digital Health provides the following package:

- com.samsung.android.sdk.healthdata

Main interfaces and classes in the library are described in Table 2. See the API reference on details.

<table>
<thead>
<tr>
<th>Interface / Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HealthConnectionErrorResult</td>
<td>This class handles errors for connection failure to the health data store.</td>
</tr>
<tr>
<td>HealthConstants</td>
<td>This class defines constants of health data and contains interfaces for various kinds of health data such as the step count or exercise.</td>
</tr>
<tr>
<td>HealthData</td>
<td>This class is an object for a health data type, e.g., the blood pressure or weight. Quantitative and qualitative values can be specified for the specific health data type based on its data structure definition. It is used to manage health data with HealthDataResolver.</td>
</tr>
<tr>
<td>HealthDataObserver</td>
<td>This class defines an observer to handle health data changes.</td>
</tr>
<tr>
<td>HealthDataResolver</td>
<td>This class accesses health data to insert, read, update, and delete with the filter and aggregate functions.</td>
</tr>
<tr>
<td>HealthDataService</td>
<td>This class initializes the health data service.</td>
</tr>
<tr>
<td>HealthDataStore</td>
<td>This class handles the connection to the data store in the device.</td>
</tr>
<tr>
<td>HealthDataUnit</td>
<td>This class provides unified units for the health data store.</td>
</tr>
<tr>
<td>HealthDevice</td>
<td>This class contains detailed device information that provides health data.</td>
</tr>
<tr>
<td>HealthDeviceManager</td>
<td>This class manages devices related health data.</td>
</tr>
<tr>
<td>HealthPermissionManager</td>
<td>This class requests permission to read or write health data for the specific health data type.</td>
</tr>
<tr>
<td>HealthResultHolder</td>
<td>This interface represents the result of invoking method.</td>
</tr>
<tr>
<td>HealthUserProfile</td>
<td>This class provides user information.</td>
</tr>
</tbody>
</table>

Table 2: Interfaces and classes of com.samsung.android.sdk.healthdata
2. Development Environment

This chapter includes development environment for Samsung Digital Health applications. Check prerequisites first and follow all steps below.

2.1. Prerequisites

Check following prerequisites before downloading Samsung Digital Health SDK.

Android Version

Android 4.4 KitKat (API level 19) or above

Available Devices

Android smartphones including non-Samsung devices that are available S Health 4.x

S Health Version

A Samsung Digital Health application runs with S Health 4.0 or above.

2.2. Downloading Samsung Digital Health SDK

Samsung Digital Health SDK can be downloaded on the Samsung developer site and you can find the following content related to the Health Data package for the application development.

<table>
<thead>
<tr>
<th>Folder in SDK</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docs</td>
<td>API Reference</td>
</tr>
<tr>
<td></td>
<td>Describes Samsung Digital Health APIs</td>
</tr>
<tr>
<td></td>
<td>Programming Guide</td>
</tr>
<tr>
<td></td>
<td>Contains development information to create Samsung Digital Health applications</td>
</tr>
<tr>
<td>Libs</td>
<td>samsung-digital-health-healthdata-va.b.c.jar</td>
</tr>
<tr>
<td></td>
<td>Health data library of Samsung Digital Health</td>
</tr>
<tr>
<td>Samples</td>
<td>SimpleHealth</td>
</tr>
<tr>
<td></td>
<td>Sample application that demonstrates how to use Samsung Digital Health SDK</td>
</tr>
<tr>
<td></td>
<td>FoodNote</td>
</tr>
<tr>
<td></td>
<td>Sample application to check daily calorie intake</td>
</tr>
<tr>
<td>Tools</td>
<td>DataViewer.apk</td>
</tr>
<tr>
<td></td>
<td>A tool to test an application before getting an approval for the partner application of S Health. It works on the S Health’s developer mode and shows saved health data of the S Health’s health data store and can add data to S Health.</td>
</tr>
</tbody>
</table>

Table 3: Samsung Digital Health SDK
2.3. S Health Branding Guidelines

Samsung Digital Health provides user interface guidelines for its partner applications to unify the entry point for the connection to S Health and related interfaces. Read "S Health Branding Guidelines" and apply them to your application. An application that doesn’t keep the guidelines can be rejected for the partner application.

2.4. Data Synchronization Design

The partner application of S Health needs to synchronize health data with S Health. The available data types are:

- Step count / Daily step count trend
- Exercise
- Sleep / SleepStage
- Food info / Food intake / Caffeine intake / Water intake
- Weight
- Health rate
- Body temperature
- Blood pressure
- Blood glucose
- Oxygen saturation
- HbA1c
- Electrocardiogram
- Ambient temperature and humidity
- UV exposure
- User profile (read-only)

You can read data above from S Health and write your application’s data to S Health except user profile.

Main data types that represent your application should be contained for data synchronization and synched data between your application and S Health needs to give a natural flow to users. E.g. if your application is a nutrition app, user thinks food intake data would be shared with S Health without doubt.

2.5. Creating an Application

If you checked steps of 2.1 ~ 2.4, it’s ready to start the application creation. See health data fundamentals and create your application.

2.6. Testing Your Application

Though you’ve created an application with Samsung Digital Health SDK, the app can work with S Health 4.x only after the approval for the S Health’s partner app. To register your application as the S Health’s partner app, see Registering for Partner Apps.
Your application needs to be tested before requesting for the partner app and can use the following environment.

- S Health's developer mode
- SDK's DataView

**Note**

Samsung Digital Health SDK doesn’t support the Emulator test. Android 4.4 KitKat (API level 19) device or above is required to test your application.

### 2.6.1. S Health’s Developer Mode

S Health 4.0 or above has the developer mode to test Samsung Digital Health applications that are not registered as partner apps. The S Health’s developer mode is not activated by default. You can use the developer mode as the following steps.

1) Select the action overflow of S Health on the top-right side.
2) Find Settings > About S Health in the action list.
3) Tap the version region quickly 10 times. The exact region needs to be tapped. Refer to the red and blue box in Figure 6.
4) If it succeeds, "*(Developer Mode)*" is shown in front of the version and it means the developer mode is activated. Now you can test your application on S Health.

Figure 6 shows how to turn on or off the developer mode in S Health. If you tap the version region quickly 10 times in the developer mode as the right figure, the developer mode is deactivated.
2.6.2. DataViewer

DataViewer shows saved health data in S Health and can create new test data for each data type. It works after the S Health’s developer mode is active. See DataViewer for more information.

![DataViewer screenshot](image)

Figure 7: DataViewer

2.6.3. Checklist for Samsung Digital Health App

Samsung Digital Health provides a checklist to test an application for the S Health’s partner app that contains basic test items. Download it [here](#) and check your application if it satisfies items on checklist as Figure 8. It helps to save your time for the registration of S Health’s partner applications.

![Checklist screenshot](image)

Figure 8: Checklist for Samsung Digital Health app
2.7. Publishing Your Application

If your application is ready after, package and publish it on the application market such as Google Play.

Your released application should run well even if it doesn’t become the S Health’s partner app yet. Check connection error handling to the health data store of S Health especially.

2.8. Registering for Partner Apps

The application works with S Health after the partner application registration. Because health data is closely connected to the privacy issue, Samsung checks violations in your application and registers your application as the S Health’s partner application.

Request the partner app on the developer site after sign-in.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only the approved partner application runs with S Health 4.x. Otherwise the application can be tested on the developer mode.</td>
</tr>
</tbody>
</table>
3. DataViewer

*DataViewer* is a very useful tool to check data synchronization with your application and *S Health*. You can find it in the downloaded SDK's /HealthData/Tools folder.

**Prerequisites**

- Activate the S Health’s [developer mode](#).
- Install *DataViewer* and open it.

*DataViewer* provides the SDK’s supported data type list and some useful menus. Figure 9 shows its main screen and menus.

![Figure 9: Main screen and menus of DataViewer](image)

3.1. Checking Saved Data

You can check saved data in *S Health* by selecting one of data types in Figure 10. Saved data are listed up with its saved time after allowing the selected data type’s permission. Each data contains detailed information for the data type’s properties and you can see it by selecting one of data.

Data’s detailed information describes its measured time and value including common information such as its created time, the source app’s package name, and source device.

Checking saved data with *DataViewer* is the exact way to test your application when the application writes measured data to *S Health*. Because inserted data to *S Health* cannot be shown on *S Health* with its UI policy. Make sure if the following information is saved well especially:
- Measured data value and its unit
- ‘start_time’ that indicates the measurement time and ‘time_offset’
- Other interesting information

![Figure 10: Checking saved data of food intake](image)

### 3.2. Adding New Data

You can add new data easily with `DataViewer` by selecting the “ADD” upper right button on the saved data list.

It is very helpful to test your application when it reads `S Health`’s new data. Check your application after filling values of mandatory properties and saving it. Mandatory properties are marked with 'not null'.

### 3.3. Connecting to S Health

`DataViewer` handles `S Health`’s all data types and each data type can be accessed by user’s content. Whenever you select any data type on the main screen’s data list, the permission UI is popped up as the middle one of Figure 10 to allow only one data type’s permission.

If you want to allow access for several data types at once, use “Connect to S Health” of `DataViewer`. It shows the permission UI pop-up that contains permission of all data types.

### 3.4. Exporting Data

`DataViewer`’s data can be exported to the csv file by selecting “Export Data” on the `DataViewer`’s main screen.
3.5. Showing on Tracker Tile

*DataViewer* has its tracker and its tracker tile can be posted on *S Health* as other trackers. Interesting data type’s latest data displays on the posted tile by setting “Show on Tracker Tile” of *DataViewer*.

![Figure 11: DataViewer’s tracker tile](image)

*Figure 11: DataViewer’s tracker tile*
4. Health Data Fundamentals

The following sections give you fundamentals for developing a Samsung Digital Health application. Check development environment for Samsung Digital Health applications first.

4.1. Importing Library

Add the following library to the “libs” folder in your created application project.

- samsung-digital-health-healthdata-a.b.c.jar

4.2. Hello Samsung Digital Health

The snippets below show the basic tasks to use the Health Data package.

Health Data Service Initialization

The health data service can be initialized with:

- HealthDataService.initialize()
Health Data Store Connection

If the health data service initialization succeeds, connect to the health data store with HealthDataStore.

```java
// Create a HealthDataStore instance and set its listener
mStore = new HealthDataStore(this, mConnectionListener);
// Request the connection to the health data store
mStore.connectService();
```

You can end the health data store connection when the activity is destroyed.

```java
@Override
public void onDestroy() {
    mStore.disconnectService();
    super.onDestroy();
}
```

The connection result is sent to HealthDataStore.ConnectionListener. If it succeeds, acquiring data permission or querying data is available.

```java
private final HealthDataStore.ConnectionListener mConnectionListener = new HealthDataStore.ConnectionListener() {
    @Override
    public void onConnected() {
        Log.d(APP_TAG, "Health data service is connected.");
        HealthPermissionManager pmsManager = new HealthPermissionManager(mStore);
        try {
            // Check whether the permissions that this application needs are acquired
            // Request the permission for reading step counts if it is not acquired
            // Get the current step count and display it if data permission is required
            // ...
        } catch (Exception e) {
            Log.e(APP_TAG, e.getClass().getName() + " - " + e.getMessage());
            Log.e(APP_TAG, "Permission setting fails.");
        }
    }

    @Override
    public void onConnectionFailed(HealthConnectionErrorResult error) {
        Log.d(APP_TAG, "Health data service is not available.");
        showConnectionFailureDialog(error);
    }

    @Override
    public void onDisconnected() {
        Log.d(APP_TAG, "Health data service is disconnected."可谓);
    }
};
```

The connection to the health data store can fail and you can check its error result in onConnectionFailed(). If there is an error, an application checks whether the health framework provides a solution with hasResolution() and calls resolve().
If the health framework provides its solution, `resolve()` makes an application move to one of the following page without a dialog message:

- App market’s *S Health* page to install or update it
- Device’s Settings page to make *S Health* available
- User’s agreement page of *S Health*

An application needs to show a proper message for each error case and call `resolve()`.

```java
private void showConnectionFailureDialog(HealthConnectionErrorResult error) {
    AlertDialog.Builder alert = new AlertDialog.Builder(this);
    mConnError = error;
    String message = "Connection with S Health is not available";

    if (mConnError.hasResolution()) {
        switch(error.getErrorCode()) {
            case HealthConnectionErrorResult.PLATFORM_NOT_INSTALLED:
                message = "Please install S Health";
                break;
            case HealthConnectionErrorResult.OLD_VERSION_PLATFORM:
                message = "Please upgrade S Health";
                break;
            case HealthConnectionErrorResult.PLATFORM_DISABLED:
                message = "Please enable S Health";
                break;
            case HealthConnectionErrorResult.USER_AGREEMENT_NEEDED:
                message = "Please agree with S Health policy";
                break;
            default:
                message = "Please make S Health available";
                break;
        }
    }

    alert.setMessage(message);
    alert.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int id) {
            if (mConnError.hasResolution()) {
                mConnError.resolve(mInstance);
            }
        }
    });

    if (error.hasResolution()) {
        alert.setNegativeButton("Cancel", null);
    }

    alert.show();
}
```

See the [health data store](#) for more information.
Permission Request

The meta-data element for required data permission value in manifest works with the permission request API. If you want to request data permission for reading step count, write its value in your application project’s manifest as the following example.

The declared data permission in manifest is checked when S Health is initialized. See privacy check flow in privacy.

```xml
<application
    <meta-data
        android:name="com.samsung.android.health.permission.read"
        android:value="com.samsung.health.step_count" />
</application>
```

Create a permission key set and add a permission key for reading step count.

```java
public class MainActivity extends Activity {
    private Set<PermissionKey> mKeySet;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // ...
        mKeySet = new HashSet<PermissionKey>();
        mKeySet.add(new PermissionKey(HealthConstants.StepCount.HEALTH_DATA_TYPE, PermissionType.READ));
        HealthDataService healthDataService = new HealthDataService();
        // initialize and connect to health data store
    }
}
```

And call HealthPermissionManager.requestPermissions() with its listener.

```java
private final HealthDataStore.ConnectionListener mConnectionListener = new HealthDataStore.ConnectionListener() {
    @Override
    public void onConnected() {
        Log.d(APP_TAG, "Health data service is connected.");
        HealthPermissionManager pmsManager = new HealthPermissionManager(mStore);
        try {
            // Check whether the permissions that this application needs are acquired
            Map<PermissionKey, Boolean> resultMap = pmsManager.isPermissionAcquired(mKeySet);
            if (resultMap.containsValue(Boolean.FALSE)) {
                // Request the permission for reading step counts if it is not acquired
                pmsManager.requestPermissions(mKeySet, MainActivity.this).setResultListener(mPermissionListener);
            } else {
                // Get the current step count and display it
                // ...
            }
        } catch (Exception e) {
            Log.e(APP_TAG, e.getClass().getName() + " - " + e.getMessage());
            Log.e(APP_TAG, "Permission setting fails.");
        }
    }
};
```
If `requestPermissions()` is called successfully, the permission UI is popped up to the user.

![Permission UI for reading step count](image)

**Figure 12: Permission UI for reading step count**

User’s permission information is saved by selecting “DONE” after the user allows each data permission. And it is received through `HealthResultHolder.ResultListener`.

```java
private final HealthResultHolder.ResultListener<PermissionResult> mPermissionListener =
    new HealthResultHolder.ResultListener<PermissionResult>() {
        @Override
        public void onResult(PermissionResult result) {
            Log.d(APP_TAG, "Permission callback is received.");
            Map<PermissionKey, Boolean> resultMap = result.getResultMap();
            if (resultMap.containsValue(Boolean.FALSE)) {
                // Requesting permission fails
            } else {
                // Get the current step count and display it
            }
        }
    };
```

See the [health permission manager](#) for more information.
5. Health Data Features

Key features of Samsung Digital Health SDK are:

- Health data store
- Health data types
- Privacy

5.1. Health Data Store

Samsung Digital Health SDK provides the health data store to access health data with user’s consent. Data in the health data store can be shared with other Samsung Digital Health applications also.

![Figure 13: S Health data store](image)

5.1.1. Preparing Data Access

An application needs to initialize the health data service, connect to the health data store, and acquire data permission to access the health data store’s data. Figure 14 shows a basic flow and main APIs for working with S Health.

![Figure 14 Flow for data access](image)
In the connection failure case, all exceptions should be handled clearly with a proper message to user. It helps to prevent unexpected operations on your application. For detailed code, see:

- [Health service initialization snippet](#)
- [Health data connection snippet](#)
- [Acquiring data permission snippet](#)

### 5.1.2. Health Data Query

HealthDataResolver helps to access the health data store’s data with APIs of Figure 15.

**Figure 15: Health data query**

All queries except inserting data can be requested with the filter to set the target range for a query. And a data query is sent asynchronously or synchronously.

Asynchronous query is used usually and you can select one of methods for your need. See the following examples for more information.

- [Asynchronous data query](#)
- [Synchronous data query](#)

The query result is retrieved as Table 4.

<table>
<thead>
<tr>
<th>Request</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert()</td>
<td>BaseResult</td>
</tr>
<tr>
<td>update()</td>
<td></td>
</tr>
<tr>
<td>delete()</td>
<td></td>
</tr>
<tr>
<td>read()</td>
<td>ReadResult</td>
</tr>
<tr>
<td>aggregate()</td>
<td>AggregateResult</td>
</tr>
</tbody>
</table>

**Table 4: Data request and its result**

See data query snippets in [API Reference](#).
5.1.3. Filter

`HealthDataResolver.Filter` is very useful to clear the data range for reading, updating, and aggregating health data. Multiple filters can be combined with:

- `and()`
- `or()`

See more descriptions in [API Reference](#).

5.1.4. Data Normalization

The health data is stored based on the normalized unit as defined in International System of Unit (SI) in Table 5.

<table>
<thead>
<tr>
<th>Item</th>
<th>Unit</th>
<th>Item</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>cm</td>
<td>Calorie</td>
<td>kcal</td>
</tr>
<tr>
<td>Weight</td>
<td>kg</td>
<td>Speed</td>
<td>m/s</td>
</tr>
<tr>
<td>Temperature</td>
<td>Celsius</td>
<td>Distance</td>
<td>meter</td>
</tr>
<tr>
<td>Blood glucose</td>
<td>mmol/L</td>
<td>Time</td>
<td>millisecond</td>
</tr>
<tr>
<td>Blood pressure</td>
<td>mmHg</td>
<td>Water intake</td>
<td>ml</td>
</tr>
<tr>
<td>HbA1c</td>
<td>%</td>
<td>Caffeine intake</td>
<td>mg</td>
</tr>
<tr>
<td>SpO₂</td>
<td>%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: International System of Unit

It enables your application to read health data without the specific unit. It means that you need to be careful to unify the data units when you insert health data to the health data store. `HealthDataUnit` helps to convert data value between units.

- **Converting unit**

If there is no required unit in `HealthDataUnit`, you can define and register a new unit as the following example.

- **Registering custom unit**
5.2. Health Data Type

Samsung Digital Health provides useful predefined and custom data types. Predefined data types of Samsung Digital Health SDK are in Table 6.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HealthConstants.Exercise</td>
<td>Whole workout that user does for enhancing fitness and health</td>
</tr>
<tr>
<td>HealthConstants.Sleep</td>
<td>Information about user sleep</td>
</tr>
<tr>
<td>HealthConstants.SleepStage</td>
<td>Sleep stage information for the specified sleep data</td>
</tr>
<tr>
<td>HealthConstants.FoodInfo</td>
<td>Set of particular food information such as nutrition and calorie</td>
</tr>
<tr>
<td>HealthConstants.FoodIntake</td>
<td>Records about how much user takes food and calories. Link to FoodInfo to show nutritional information on S Health.</td>
</tr>
<tr>
<td>HealthConstants.CaffeinIntake</td>
<td>Amount of caffeine intake</td>
</tr>
<tr>
<td>HealthConstants.WaterIntake</td>
<td>Records about how much and many times user takes water</td>
</tr>
<tr>
<td>HealthConstants.Weight</td>
<td>Historical records of user's weight</td>
</tr>
<tr>
<td>HealthConstants.HeartRate</td>
<td>Measured heart rate values represented in beats per minute</td>
</tr>
<tr>
<td>HealthConstants.BodyTemperature</td>
<td>Body temperature of user, known as normothermia or eutherma</td>
</tr>
<tr>
<td>HealthConstants.BloodPressure</td>
<td>The pressure exerted by circulating blood upon walls of blood vessels</td>
</tr>
<tr>
<td>HealthConstants.BloodGlucose</td>
<td>The amount of glucose (sugar) present in blood</td>
</tr>
<tr>
<td>HealthConstants.OxygenSaturation</td>
<td>The oxygen saturation value of user</td>
</tr>
<tr>
<td>HealthConstants.HbA1c</td>
<td>Glycated hemoglobin data</td>
</tr>
<tr>
<td>HealthConstants.Electrocardiogram</td>
<td>Electrical activity of the heart</td>
</tr>
<tr>
<td>HealthConstants.AmbientTemperature</td>
<td>Ambient temperature and humidity data around device</td>
</tr>
<tr>
<td>HealthConstants.UvExposure</td>
<td>Degree of exposure to ultraviolet(UV) radiated from the sun</td>
</tr>
<tr>
<td>HealthUserProfile</td>
<td>S Health's user profile such as the date of birth or gender. It is provided with read-only.</td>
</tr>
</tbody>
</table>

Table 6: Predefined data types

And you can use daily step count trend if you need older data than one month as the custom data type.
All data types have the following mandatory properties in Table 7 and each data type’s detailed properties are written in the API Reference.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUID</td>
<td>ID of health data. Assigned by the system when new data is created in S Health.</td>
</tr>
<tr>
<td>CREATED_TIME</td>
<td>UTC time when data is created. Assigned by the system when new data is created in S Health.</td>
</tr>
<tr>
<td>UPDATED_TIME</td>
<td>UTC time when data is updated. Assigned by the system when new data is created in S Health.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>Application package name which provides data. Assigned by the system when new data is created in S Health.</td>
</tr>
<tr>
<td>DEVICE_UUID</td>
<td>Device ID which provides health data</td>
</tr>
</tbody>
</table>

**Table 7: Mandatory properties of health data**

The manifest for the data type is composed of meta-data to present characteristics of the health data type. It is defined as the XML schema. Table 8 shows each element description for the manifest.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;manifest&gt;</td>
<td>Base element for the data type. Its attributes are shown below.</td>
</tr>
<tr>
<td></td>
<td>- &quot;id&quot;: Data type ID</td>
</tr>
<tr>
<td></td>
<td>- &quot;import&quot;: Indicates another &quot;id&quot; of another manifest to extend</td>
</tr>
<tr>
<td></td>
<td>- &quot;version&quot;: Indicates the version of the manifest. It’s numeric. If there are manifests with different version values, the higher version works.</td>
</tr>
<tr>
<td>&lt;publisher&gt;</td>
<td>Publisher name. Its attribute is shown below.</td>
</tr>
<tr>
<td></td>
<td>- &quot;name&quot;: Name of publisher</td>
</tr>
<tr>
<td>&lt;documentation&gt;</td>
<td>Documentation section defines informational text. It has the following sub elements.</td>
</tr>
<tr>
<td></td>
<td>- &lt;title&gt;</td>
</tr>
<tr>
<td></td>
<td>- &lt;description&gt;</td>
</tr>
<tr>
<td>&lt;title&gt;</td>
<td>Title of the data type with the specific language code for localization. Its attribute is shown below.</td>
</tr>
<tr>
<td></td>
<td>- &quot;lang&quot;: It is composed of the following values. E.g. lang=&quot;en_US&quot;</td>
</tr>
<tr>
<td></td>
<td>- The language code with two-letter lowercase as defined by ISO 639-1.</td>
</tr>
<tr>
<td></td>
<td>- The country code with two-letter uppercase as defined by ISO 3166-1.</td>
</tr>
<tr>
<td></td>
<td>The value of this element without the &quot;lang&quot; language attribute becomes the default title.</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>Description for the data type. Its attribute is as follows.</td>
</tr>
<tr>
<td></td>
<td>- &quot;lang&quot;: It is composed of the following values. E.g. lang=&quot;en_US&quot;</td>
</tr>
</tbody>
</table>
- The language code with two-letter lowercase as defined by ISO 639-1.
- The country code with two-letter uppercase as defined by ISO 3166-1.

The value of this element without the "lang" language attribute becomes the default title.

### <visibility>
Sharing scope of the data type. Its attribute is shown below.
- "public"
  - "yes" (default) if the data type is shared to any application, or "no"
  - Specify <owner-app-list> as a sub element if it's not public.

### <owner-app-list>
If the visibility is not public, you can list the allowed applications to access or extend this data type. It has a <package> sub element.

### <package>
Package name of the allowed application.

### <policy>
Data policy. Its attributes are shown below.
- "lifetime": Data’s lifetime in months. E.g. lifetime="72" (it means 6 years.)
- "measurement": Measurement type which is one of the following values.
  - "discrete" for discretely measured data without the end time.
  - "session" for measured data with the end time which has a session.
  - "no": Not measured data
- "privacy": The privacy level which is one of the following values.
  - "insensitive": Data encryption is not required.
  - "sensitive": Data encryption is required. It’s the default value.
  - "medical": Data encryption and listing allowed countries with <medical-country> are required.

### <medical-country>
Need to write all allowed countries if the "privacy" attribute is "medical". It has the following attribute and allowed country code values.
- "allowed"
  - "yes": If all allowed counties are written.
  - "no": If all prohibited countries are written.
- Allowed or prohibited country codes with two-letter uppercase as defined by ISO 3166-1.
  - Use comma "," as the delimiter between country codes.
  - E.g. <policy lifetime="72" measurement="discrete" privacy="medical"> <medical-country allowed="no">AO, GB, IE</medical-country> </policy>

### <property>
Defines detailed information for each property of this data type including the default value, validation range, title and description. Its attributes are shown below.
- "name"
- "type"
  - One of "int", "long", "float", "double", "text", and "blob"
- "mandatory"
  - The default value is "no".
  - "yes" if the property is a mandatory property for the data type. Be careful not to make the
property as "yes" if you add a new property to update your application.

- If the property is defined as mandatory, the value for the property has to be specified when new data is inserted.
  - "unique"
  - The default value is "no".
  - "yes" if the data value needs to be unique

<table>
<thead>
<tr>
<th>&lt;default-value&gt;</th>
<th>Default value of the property to be filled when a new health data is inserted without value specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;validation&gt;</td>
<td>Validation range</td>
</tr>
<tr>
<td></td>
<td>- &quot;min&quot;</td>
</tr>
<tr>
<td></td>
<td>- The minimum value if the type is numeric</td>
</tr>
<tr>
<td></td>
<td>- &quot;max&quot;</td>
</tr>
<tr>
<td></td>
<td>- The maximum value if the type is numeric</td>
</tr>
<tr>
<td></td>
<td>- &quot;minLength&quot;</td>
</tr>
<tr>
<td></td>
<td>- The minimum length if the type is text</td>
</tr>
<tr>
<td></td>
<td>- &quot;maxLength&quot;</td>
</tr>
<tr>
<td></td>
<td>- The maximum length if the type is text</td>
</tr>
<tr>
<td>&lt;documentation&gt;</td>
<td>Description for the property of the data type.</td>
</tr>
</tbody>
</table>

**Table 8: Manifest definition**

An example below shows the manifest definition for the health data type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns="http://schemas.samsung.com/2014/10/iot"
  id="com.samsung.health.blood_glucose"
  version="1">
  <publisher name="Samsung Electronics Co., Ltd."/>
  <documentation>
    <title>Blood Glucose</title>
    <title lang="es_US">Glucosa en la sangre</title>
  </documentation>
  <policy lifetime="72" measurement="discrete" privacy="medical">
    !-- AO (Angola), GB (United Kingdom), IE (Ireland), FR (France), NL (Netherlands),
    BE (Belgium), LU (Luxembourg), AT (Austria), SK (Slovak), GR (Greece), CY (Cyprus),
    SI (Slovenia), DZ (Algeria), IR (Iran), CA (Canada), KR (Korea) -->
    <medical-country allowed="no">AO, GB, IE, FR, NL, BE, LU, AT, SK, GR, CY, SI, DZ, IR, CA, KR</medical-country>
  </policy>
  <property name="datauuid" mandatory="yes" type="text" unique="yes">
    <validation minLength="10" maxLength="36"/>
    <documentation>
      <title>Unique ID</title>
      <description>Unique ID</description>
    </documentation>
  </property>
  ...
  <property name="start_time" mandatory="yes" type="long">
```
5.3. Privacy

Privacy is a key for handling health data. Samsung Digital Health enables your application to access health data based on user’s consent. An application needs to declare proper permissions for handling required health data types and to handle SecureException when the application cannot gain user consent because user can withdraw consent at any time. See Permission Manager for permission declaration and request.

\[ S\ Health \] reads declared permissions of the application manifest file when it initialize and you can check them in \[ S\ Health > Settings > [App] \] as Figure 16.

On the other hand, an application needs to make a permission setting menu separately. The application calls a permission request API with required permission keys and the health data framework pops the permission UI up on the application. The flow can be preceded after user approves or denies the data type usage. User may change permission for each data type at any time.

In this time, items of the permission UI and \[ S\ Health\]’s setting menu can be different. Make same:

- Declared permissions in manifest
- Permission keys for the permission request API.

User consent is limited to the device. Even if multiple devices use the same Samsung account, the application has to acquire user consent on each device independently.

![Diagram of permission setting](image)

**Figure 16: Privacy and user permission**
6. Sample Application - SimpleHealth

The Health Data package provides a 'SimpleHealth' sample application to show how to read the today’s step count including its permission request. Check the sample application by importing its project with Android Studio (or Eclipse).

Prerequisites

Prerequisites to run SimpleHealth are:

- Prepare an Android device that supports Android 4.4KitKat (API level 19) or above. (Samsung and non-Samsung devices are available both.)
- Install S Health 4.x on the device.
- Turn on the developer mode.
- Import SimpleHealth with Android Studio (or Eclipse)
- Run SimpleHealth on the device.

Overview

SimpleHealth demonstrates how to use Samsung Digital Health’s APIs for retrieving the today’s step count with HealthConstants.StepCount health data and requesting permission.

Figure 17 shows its screenshots.

Figure 17: SimpleHealth
Especially, the health data store connection and exception handling would be helpful to your application to prepare unexpected error.

Table 9 describes source and resource files of SimpleHealth.

<table>
<thead>
<tr>
<th>Source / Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndroidManifest.xml</td>
<td>Declaring required permissions with the &lt;meta-data&gt; element.</td>
</tr>
<tr>
<td>src/com/Samsung/android/simplehealth</td>
<td></td>
</tr>
<tr>
<td>MainActivity.java</td>
<td>SimpleHealth’s main activity.</td>
</tr>
<tr>
<td></td>
<td>- Connecting to the health data store and handling exceptions through ConnectionListener</td>
</tr>
<tr>
<td></td>
<td>- Checking permission to read HealthConstants.StepCount and requesting permission if required.</td>
</tr>
<tr>
<td></td>
<td>- Creating an option menu for setting permission.</td>
</tr>
<tr>
<td></td>
<td>- Showing today’s step count.</td>
</tr>
<tr>
<td>StepCountReporter.java</td>
<td>- Reading today’s the total step count asynchronously.</td>
</tr>
<tr>
<td></td>
<td>- Adding an observer to get notification for the step count’s change.</td>
</tr>
</tbody>
</table>

**Table 9: SimpleHealth’s source description**

See detailed implementation in the imported SimpleHealth’s project through Android Studio (or Eclipse).
7. Sample Application - FoodNote

The Health Data package provides a 'FoodNote' sample application to show how to manage food intake data with S Health including the permission request. Check the sample application by importing its project with Android Studio (or Eclipse).

Prerequisites

Prerequisites to run FoodNote are:

- Prepare an Android device that supports Android 4.4KitKat (API level 19) or above. (Samsung and non-Samsung devices are available both.)
- Install S Health 4.x on the device.
- Turn on the developer mode.
- Import FoodNote with Android Studio (or Eclipse)
- Run FoodNote on the device.

Overview

FoodNote demonstrates how to use Samsung Digital Health's APIs for checking the selected day's calorie intake with HealthConstants.FoodIntake and HealthConstants.FoodInfo including permission request.

Figure 18 shows its screenshots.

Figure 18: FoodNote
Especially, the health data store connection and exception handling would be helpful to your application to prepare unexpected error.

Table 10 describes source and resource files of *FoodNote*.

<table>
<thead>
<tr>
<th>Source / Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndroidManifest.xml</td>
<td>Declaring required permissions with the <code>&lt;meta-data&gt;</code> element.</td>
</tr>
<tr>
<td>src/com/Samsung/android/app/foodnote</td>
<td></td>
</tr>
<tr>
<td>AppConstants.java</td>
<td>Getting the meal type name.</td>
</tr>
<tr>
<td>ChooseFoodActivity.java</td>
<td>Loading the food list of the <em>FoodInfoTable</em> class and showing them that user can choose.</td>
</tr>
<tr>
<td>FoodDataHelper.java</td>
<td>- Inserting food intake data</td>
</tr>
<tr>
<td></td>
<td>- Deleting food intake data</td>
</tr>
<tr>
<td></td>
<td>- Reading food intake data of the selected day and the meal type</td>
</tr>
<tr>
<td>FoodInfoTable.java</td>
<td>Defining sample food information.</td>
</tr>
<tr>
<td>MainActivity.java</td>
<td><em>FoodNote</em>’s main activity.</td>
</tr>
<tr>
<td></td>
<td>- Connecting to the health data store and handling exceptions through ConnectionListener.</td>
</tr>
<tr>
<td></td>
<td>- Checking permission for HealthConstants.FoodInfo and HealthConstants.FoodIntake, and requesting permission if required.</td>
</tr>
<tr>
<td></td>
<td>- Creating an option menu for setting permission.</td>
</tr>
<tr>
<td></td>
<td>- Showing the selected day’s calorie intake.</td>
</tr>
<tr>
<td></td>
<td>- Adding an observer to get notification for the calories change.</td>
</tr>
<tr>
<td>MealStoreActivity.java</td>
<td>- Showing a list of food intake items for the selected meal that is chosen on <em>MainActivity</em>.</td>
</tr>
<tr>
<td></td>
<td>- Deleting a food intake item in the list.</td>
</tr>
</tbody>
</table>

**Table 10: *FoodNote*’s source description**
Copyright

Copyright © 2016 Samsung Electronics Co. Ltd. All Rights Reserved.

Though every care has been taken to ensure the accuracy of this document, Samsung Electronics Co., Ltd. cannot accept responsibility for any errors or omissions or for any loss occurred to any person, whether legal or natural, from acting, or refraining from action, as a result of the information contained herein. Information in this document is subject to change at any time without obligation to notify any person of such changes.

Samsung Electronics Co. Ltd. may have patents or patent pending applications, trademarks copyrights or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give the recipient or reader any license to these patents, trademarks copyrights or other intellectual property rights.

No part of this document may be communicated, distributed, reproduced or transmitted in any form or by any means, electronic or mechanical or otherwise, for any purpose, without the prior written permission of Samsung Electronics Co. Ltd.

The document is subject to revision without further notice.

All brand names and product names mentioned in this document are trademarks or registered trademarks of their respective owners.

For more information, please visit http://developer.samsung.com/