healthR

Developer Documentation

Introduction

healthR is a service that allows healthcare organisations and digital health solution providers to access data from the best health apps and devices through a single API and data model.

This document provides an overview of the different ways the service can be used, along with information on how to use the API and the normalised data model.

Devices and Data Sources

The service currently supports the following data sources via Web APIs provided by device vendors:

- Fitbit
- Withings
- Google Fit
- iHealth
- Garmin
- Omron
- Qardio
- Polar
- Oura Ring
- Misfit
- Suunto Watches
- Strava
- Huami / AmazFit
- Huawei

We can also help you integrate data from Apple Health natively from your app into your backend / cloud platforms. Data from Apple Health resides on the device and needs to be integrated by calling the Apple Health APIs locally on the device.

Data Types

The healthR service allows your users to connect and authorise their wearables device account to your app. The service then polls the device account for new data at regular intervals. Data from the different devices is normalised into a simple data model that can be consumed via a HTTP API (with JSON payload) or directly via the data store.

Below is a list of the data categories available:

- Steps
- Calories Burnt (kCal)
- Calories Consumed (kCal)
- Exercise Duration (minutes)
- Deep Sleep (minutes)
- Light Sleep (minutes)
- Weight (kg)
- Fat Free Mass (kg)
- Fat Ratio (%)
- Fat Mass Weight (kg)

- Blood Pressure Systolic (mmHg)
- Blood Pressure Diastolic (mmHg)
- Heart Rate (bpm)
- Oxygen Saturation (SpO2)
- Blood Glucose (mg/dL)
- Pulse Wave Velocity
- Body Temperature (°C / °F)

Data Model

The normalised data model contains the following fields:

- User ID (String) – the ID the calling system has set to identify the user
- RecordDate (DateTime) - the date and time of the reading as reported by the device
- Category (String) - the type of data (one of the types defined above)
- Reading (Decimal) - the actual reading value
- Unit (String) - the units the value is represented in
- SNOMEDCode (String) – the corresponding SNOMED Code for the reading
- Device (String) - the device (manufacturer) where the reading originated from

Example output as JSON is below:


```
{
  "userID": "727828141",
  "recordDate": "2020-01-07 11:23",
  "category": "Blood Pressure Systolic",
  "reading": 118.0,
  "unit": "mmHg",
  "device": "Withings",
  "snomedCode": "163020007"
}
```


API

The healthR service is built using a multi-tenant model and the HTTP API has the following main functions:

1. Associate a user with your organisation / tenant
2. Initiate a connection with a device account using OAuth
3. Querying / retrieving data for users in your organisation / tenant
4. De-authorise a device from a user
5. Remove a user and all their data

The API uses GET and POST verbs to manage operations.

API Authentication

An API Key is used to authenticate an organisation / tenant. The key is provided by our admin team and should passed using the Authorization header in each request made to the API, as follows:

Authorization: Apikey 1234567890abcdef

Adding a User

Before being able to connect to a user's wearable account, you will need to connect the user against your tenant. In order to do this, you need to POST details of the user to the following API method:

```
POST https://platform.healthr.cloud/api/accounts
{
  "name": "Karen Frances",
  "gender": "Female",
  "externalID": "7386823",
  "dob": null,
  "country": null
}
```

Connecting a Device

In order to initiate the authorisation flow for a device account, POST details of the User and the type of device you wish to connect with. For example, in order to connect to a Fitbit account, you would make the following call:

```
POST https://platform.healthr.cloud/device/connect
{
  "externalID": "7386823",
  "device": "Fitbit",
  "ConnectionKey": "78343676dghsd67512fgh",
  "DirectDataAccess": false
}
```

The user will be redirected to the Fitbit Authorisation page where they will be asked to login and authorise their data to be used by the calling app (in this case healthR). **Note**: Any screens leading up to this point can be customised by the calling app to explain the process, however, once the call to connect to a device is made, the authorisation screens for the device account will differ depending on the device manufacturer and there is no ability to customise those (standard OAuth screens).

By default, once a user has connected their wearable account with healthR, the data will be automatically synchronised and stored in our platform ready to query. If you do not want this to happen and instead would like to query the data directly, then you can set the **DataAccessOnly** flag to true. This will disable the auto-sync and you will need to use the Direct Data Access API method in order to query data directly from source. Note: this will be slower than having the data stored on

the platform and you may be limited to the last 30 days worth of data (or whatever restrictions the wearable device manufacturers have in place with their APIs).

Querying User Data

Once a connection to a device has been made, the healthR service will fetch data in the background and store this in the healthR database. This data can be queried by date, category, user. At least one parameter for the query must be provided.

```
POST https://platform.healthr.cloud/api/records/query
{
  "dateFrom": "2019-12-01",
  "dateTo": "2020-01-07",
  "externalID": "7386823",
  "category": null
}
```

Multiple categories can be requested by specifying them in a comma separated list, for example:

```
{
  ...,
  "category": "Steps,Heart Rate"
}
```

Resulting Data

The output from the query will be a list of the Wearable Record object. An example can be seen below:

```
{
  "records": [
            {
              "accountID": "727828141",
              "recordDate": "2020-01-07 11:23",
              "category": "Blood Pressure Systolic",
              "reading": 118.0,
              "unit": "mmHg",
              "device": "Withings",
              "snomedCode": "163020007",
              "title": "Some additional title",
              "data": "Some additional text data"
            }
```

```
        ]
    }
```

Direct Data Access

If during the connection process you specified Direct Data Access only (therefore not storing data in healthR for querying purposes), you will need to use the following API call to access data directly from source (manufacturers API). All of the parameters specified in the request below are mandatory and you can only request data on a per user basis (you can't request data across a number of users as you can using Query method above).

```
POST https://platform.healthr.cloud/api/records/direct
```

```
{
  "dateFrom": "2019-12-01",
  "dateTo": "2020-01-07",
  "externalID": "7386823"
}
```

Resulting Data

The resulting dataset is the same as the one returned in Query method above.

Adding Longitudinal Data / Custom Device Data

healthR supports the ability for apps / services / devices to push data to the HealthStream platform to create a longitudinal record for an individual. This allows apps to use the platform as their main data store for all health-related data for an individual. It's also allows new wearable device vendors to use the platform as their main data store for data generated from their devices.

```
POST https://platform.healthr.cloud/api/records/create
```

```
{
  "accountID": "727828141",
  "recordDate": "2020-01-07 11:23",
  "category": "Blood Pressure Systolic",
  "reading": 118.0,
  "unit": "mmHg",
  "device": "Withings",
  "snomedCode": "163020007",
  "title": "Some additional title",
  "data": "Some additional text data"
}
```

Where there is text-based data that needs to be stored rather than numerical readings, the Title and Data fields can be used to store long text such as consultation notes, summaries or general information. Structured data / documents (JSON / XML) can also be stored in the Data field.

Deauthorise Device for User

Client apps can allow a user to de-authorise a device and prevent any further data from being retrieved by calling the following API method:

```
POST https://platform.healthr.cloud/api/account/deauthorise

{

  "device": "Google Fit",

  "externalID": "7386823",

}
```

Deployment Models

The healthR service is available as SaaS offering as well as being able to be deployed on premise / client infrastructure. To consume the service using a SaaS model, our admin team will setup your tenant and provide you with the API Key required to start integrating wearable data. For the on-premise version, there are several customisations that can be applied from the normalised data model to the database used to store all wearable data.

On-premise Deployment

For the on-premise deployment, the client is responsible for getting API Keys / OAuth 2.0 credentials for each wearable they wish their users to connect & authorise. This option allows for the user experience to be consistent with client brand reflected throughout the OAuth process for each device. This option also allows for the wearable data to be queried directly from the data store for analytics.

Vision API

healthR now supports **Optical Character Recognition (OCR)** and **Medical Concept detection** via our new Vision API. You can now upload images containing text (prescriptions, medical notes, readings from non-connected devices) and the API will return the text from the image along with all associated medical concepts (classification of text).

In order to get the text and medical concepts detected from an image, simply POST a base64 string of an image as below.

```
POST https://platform.healthr.cloud/api/vision/detect

{

  "ImageString":
"/9j/4AAQSkZJRgABAQEBLAEsAAD/4Q7RRXhpZgAATU0AKgAAAAgACAEOAAIAAAD8AAA
AfgESAAMAAA..."

}
```

Resulting Data

The output from the query will be a list of the Classified Terms along with their associated SNOMED codes. An example can be seen below:

```
{
  "results": [
            {
              "text": "Ibuprofen",
              "medicalConcepts": [
                        {
                              Text: Medication
                              Code: 1029288102
                        }
                  ]
            },
            ...
      ]
}
```