Exploring New Data Representations in HealthKit

Luke Spicer, Health Software Engineer
Divya Koyyalagunta, Health Software Engineer
70,000

Apps in the Health and Fitness Category on the App Store
HealthKit data model
New quantity series APIs
Beat-to-beat heart measurements
Heart rate events
Hearing health
HealthKit data model

New quantity series APIs

Beat-to-beat heart measurements

Heart rate events

Hearing health
HealthKit data model

New quantity series APIs

Beat-to-beat heart measurements

Heart rate events

Hearing health
HealthKit data model
New quantity series APIs
Beat-to-beat heart measurements
Heart rate events
Hearing health
HealthKit data model
New quantity series APIs
Beat-to-beat heart measurements
Heart rate events
Hearing health
HealthKit data model
New quantity series APIs
Beat-to-beat heart measurements
Heart rate events
Hearing health
HealthKit data model

New quantity series APIs

Beat-to-beat heart measurements

Heart rate events

Hearing health
HealthKit Data Model
Reviewing samples

Measurements at a particular time
Occur over a time interval
All Accounts

May 14, 2019
PENICK MEDICAL CENTER

LAB RESULTS

Triglycerides
129 mg/dL

ALLERGIES

Dog Dander
Documented

IMMUNIZATIONS

Hepatitis B Adult Vaccine
Administered

MEDICATIONS

Levothyroxine
Prescribed
HealthKit Data Model
Models for Rich Data

HKCorrelation
"Blood Pressure"

HKClinicalRecord
"Allergy Record"

HKSeriesSample
"Workout Route"

Set([Systolic Blood Pressure, Diastolic Blood Pressure])

{ "resourceType": "AllergyIntolerance", ...

[CLLocation 1, CLLocationCoordinate2, CLLocationCoordinate3, ...]
HealthKit data model

New quantity series APIs

Beat-to-beat heart measurements

Heart rate events

Hearing health
Quantity Data
Components of Quantity Data

Walking Distance
- Quantity Type: Walking Distance
- Quantity Sample: 5 km
- Date Interval:
  - Start: May 28, 2019 7:15:00
  - End: May 28, 2019 9:30:00

Body Mass
- Quantity Type: Body Mass
- Quantity Sample: 60 kg
- Date Interval:
  - Start: May 29, 2019 8:00:00
  - End: May 29, 2019 8:00:00

Heart Rate
- Quantity Type: Heart Rate
- Quantity Sample: 72 BPM
- Date Interval:
  - Start: May 29, 2019 10:05:00
  - End: May 29, 2019 10:05:00
Representing Multiple Quantities

Sensor

64 BPM
66 BPM
67 BPM
65 BPM
63 BPM

0 1 2 3 4 5 6 7 8 s
Representing Multiple Quantities

Single quantity sample

65 BPM
Start: 0
End: 8
UUID: 95-95b0-c2
device: myController
metadata: {"game1"}
Representing Multiple Quantities
Multiple quantity samples

64 BPM
Start: 0
End: 0
UUID: 95-95b0-c2
device: myController
metadata: {"game1"}

66 BPM
Start: 2
End: 2
UUID: 7f-e252-ad
device: myController
metadata: {"game1"}

67 BPM
Start: 4
End: 4
UUID: 0d-296e-7e
device: myController
metadata: {"game1"}

65 BPM
Start: 6
End: 6
UUID: 80-5a7f-d3
device: myController
metadata: {"game1"}

63 BPM
Start: 8
End: 8
UUID: a2-a2e7-63
device: myController
metadata: {"game1"}
Representing Multiple Quantities

Quantity series

Start: 0  
End: 8  
UUID: 95-95b0-c2  
device: myController  
metadata: {"game1"}

64 BPM  
Start: 0  
End: 0

66 BPM  
Start: 2  
End: 2

67 BPM  
Start: 4  
End: 4

65 BPM  
Start: 6  
End: 6

63 BPM  
Start: 8  
End: 8

Sensor

64 BPM  
66 BPM  
67 BPM  
65 BPM  
63 BPM
Aggregating Multiple Quantities
## Aggregating Multiple Quantities

### Aggregation Style

<table>
<thead>
<tr>
<th>Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cumulative</td>
</tr>
<tr>
<td>Discrete</td>
</tr>
<tr>
<td>Aggregation Style</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Cumulative</td>
</tr>
<tr>
<td>Discrete</td>
</tr>
</tbody>
</table>
## Aggregating Multiple Quantities

<table>
<thead>
<tr>
<th>Aggregation Style</th>
<th>Example Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cumulative</td>
<td>Distance</td>
</tr>
<tr>
<td></td>
<td>Calories</td>
</tr>
<tr>
<td></td>
<td>Steps</td>
</tr>
<tr>
<td>Discrete</td>
<td></td>
</tr>
</tbody>
</table>
# Aggregating Multiple Quantities

<table>
<thead>
<tr>
<th>Aggregation Style</th>
<th>Example Types</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cumulative</td>
<td>Distance, Calories, Steps</td>
<td>Sum</td>
</tr>
<tr>
<td>Discrete</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Aggregating Multiple Quantities

<table>
<thead>
<tr>
<th>Aggregation Style</th>
<th>Example Types</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cumulative</td>
<td>Distance, Calories, Steps</td>
<td>Sum</td>
</tr>
<tr>
<td>Discrete</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aggregation Style</td>
<td>Example Types</td>
<td>Statistics</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Cumulative</td>
<td>Distance, Calories, Steps</td>
<td>Sum</td>
</tr>
<tr>
<td>Discrete</td>
<td>Heart Rate, Body Mass, Height</td>
<td></td>
</tr>
</tbody>
</table>
# Aggregating Multiple Quantities

<table>
<thead>
<tr>
<th>Aggregation Style</th>
<th>Example Types</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cumulative</td>
<td>Distance, Calories, Steps</td>
<td>Sum</td>
</tr>
<tr>
<td>Discrete</td>
<td>Heart Rate, Body Mass, Height</td>
<td>Average, Minimum, Maximum, Most Recent</td>
</tr>
</tbody>
</table>
Aggregating Multiple Quantities

Cumulative

Sum: 6 km

1 km  2 km  3 km
Aggregating Multiple Quantities

**Cumulative**
- Sum: 6 km
  - 1 km
  - 2 km
  - 3 km

**Discrete**
- Average: 65 BPM
- Minimum: 63 BPM
- Maximum: 67 BPM
- Most Recent: 63 BPM
  - 67 BPM
  - 65 BPM
  - 63 BPM
// Aggregating Multiple Quantities

enum HKQuantityAggregationStyle : Int {
    case cumulative
    case discrete
}

// Aggregating Multiple Quantities

enum HKQuantityAggregationStyle : Int {
    case cumulative
    case discrete
}

// Aggregating Multiple Quantities

eenum HKQuantityAggregationStyle : Int { 
    case cumulative
    case discrete
}
enum HKQuantityAggregationStyle : Int {
    case cumulative
    case discrete
    case discreteArithmetic
}

// Aggregating Multiple Quantities
enum HKQuantityAggregationStyle : Int {
    case cumulative
    case discrete
    case discreteArithmetic
    case discreteTemporallyWeighted
}

// Aggregating Multiple Quantities
// Aggregating Multiple Quantities

class HKQuantityAggregationStyle: Int {
    case cumulative
    case discrete
    case discreteArithmetic
    case discreteTemporallyWeighted
    case discreteEquivalentContinuousLevel
}
// New HKQuantitySample Subclasses

// HKCumulativeQuantitySample

class HKCumulativeQuantitySample : HKQuantitySample {
    var sumQuantity: HKQuantity { get }
}

// HKDiscreteQuantitySample

class HKDiscreteQuantitySample : HKQuantitySample {
    var minimumQuantity: HKQuantity { get }
    var averageQuantity: HKQuantity { get }
    var maximumQuantity: HKQuantity { get }
    var mostRecentQuantity: HKQuantity { get }
    var mostRecentQuantityDateInterval: DateInterval { get }
}
// New Predicate Key Paths for Quantity Sample Subclasses

// HKCumulativeQuantitySample
public let HKPredicateKeyPathSum: String

// HKDiscreteQuantitySample
public let HKPredicateKeyPathMin: String
public let HKPredicateKeyPathAverage: String
public let HKPredicateKeyPathMax: String
public let HKPredicateKeyPathMostRecent: String
public let HKPredicateKeyPathMostRecentStartDate: String
public let HKPredicateKeyPathMostRecentEndDate: String
public let HKPredicateKeyPathMostRecentDuration: String
Aggregating Multiple Quantities

Average: 65 BPM, Minimum: 63 BPM, Maximum: 67 BPM, Most Recent: 63 BPM

Start: 0
End: 8
UUID: 95-95b0-c2
device: myController
metadata: {"game1"}

64 BPM
Start: 0
End: 0

66 BPM
Start: 2
End: 2

67 BPM
Start: 4
End: 4

65 BPM
Start: 6
End: 6

63 BPM
Start: 8
End: 8
Building a Quantity Series

Average: 65 BPM, Minimum: 63 BPM, Maximum: 67 BPM, Most Recent: 63 BPM

Start: 0
End: 8

UUID: 95-95b0-c2
device: myController
metadata: {"game1"}

Sensor

64 BPM
Start: 0
End: 0

66 BPM
Start: 2
End: 2

67 BPM
Start: 4
End: 4

65 BPM
Start: 6
End: 6

63 BPM
Start: 8
End: 8
// How to Use HKQuantitySeriesSampleBuilder

// Step 1: Request authorization to read and write heart rate type
let healthStore = HKHealthStore()
let dataTypes = Set([HKObjectType.quantityType(forIdentifier: .heartRate)!!])
healthStore.requestAuthorization(toShare: dataTypes, read: dataTypes) {
    (success, error) in
    // Handle error
    // ...
}

// Step 2: Create a HKQuantitySeriesSampleBuilder
let builder = HKQuantitySeriesSampleBuilder(healthStore: healthStore
    quantityType: heartRateQuantitytype,
    startDate: gameStartDate,
    device: gameDevice)
/ How to Use HKQuantitySeriesSampleBuilder

// Step 3: Add quantities to the builder
// while game is ongoing ...
do {
  try builder.insert(heartRateQuantity, for: heartRateDateInterval)
} catch {
  // Handle insert error
}

// Step 4: Finish the HKQuantitySeriesSampleBuilder
builder.finishSeries(metadata: metadata, endDate: gameEndDate) { (samples, finishError) in
  // Handle finish error or use the samples.
}
Querying for Quantity Data
Querying for Quantity Data

Total active calories burned

1,404 kcal
May 18, 3 AM – May 19, 3 AM

You burned an average of 1,407 kcal a day over the last 7 days.
Querying for Quantity Data

Total active calories burned

Data for charts
Querying for Quantity Data

Total active calories burned

Data for charts

Average calories burned per day
Querying for Quantity Data

Total active calories burned

Data for charts

Average calories burned per day

Min and max heart rate
Querying for Quantity Data

Total active calories burned
Data for charts
Average calories burned per day
Min and max heart rate
Most recent heart rate
Querying for Quantity Data
HKStatisticsCollectionQuery

- Produces multiples statistics
- Statistics available per source
- Provides updates
- Supports all aggregation styles
- Automatically includes quantity series data
Querying for Quantity Data

Querying for individual quantities
HKQuantitySeriesSampleQuery(quantitySample: HKQuantitySample,
quantityHandler: (HKQuantitySeriesSampleQuery,
HKQuantity?,
Date?,
Bool,
Error?) -> Void)
HKQuantitySeriesSampleQuery

HKQuantitySeriesSampleQuery(quantityType: HKQuantityType, predicate: NSPredicate?, quantityHandler: (HKQuantitySeriesSampleQuery, HKQuantity?, Date?, Bool, Error?) -> Void)
HKQuantitySeriesSampleQuery

HKQuantitySeriesSampleQuery(quantityType: HKQuantityType,
predicate: NSPredicate?,
quantityHandler: (HKQuantitySeriesSampleQuery,
HKQuantity?,
DateInterval?,
HKQuantitySample?,
Bool,
Error?) -> Void)
HKQuantitySeriesSampleQuery

- Quantity: 64 BPM
- Start: 2
- End: 2
- Sample: nil
- Done: false

**QS1**
- 64 BPM
  - Start: 2
  - End: 2

**QS2**
- 65 BPM
  - Start: 6
  - End: 6
- 67 BPM
  - Start: 4
  - End: 4
- 63 BPM
  - Start: 8
  - End: 8
HKQuantitySeriesSampleQuery

quantity: 67 BPM
start: 4
end: 4
sample: nil
done: false

QS1
- 64 BPM
  Start: 2
  End: 2

QS2
- 67 BPM
  Start: 4
  End: 4

- 65 BPM
  Start: 6
  End: 6

- 63 BPM
  Start: 8
  End: 8
HKQuantitySeriesSampleQuery

quantity: 65 BPM
start: 6
end: 6
sample: nil
done: false

QS1

64 BPM
Start: 2
End: 2

65 BPM
Start: 6
End: 6

QS2

67 BPM
Start: 4
End: 4

63 BPM
Start: 8
End: 8
HKQuantitySeriesSampleQuery

- **quantity:** 63 BPM
- **start:** 8
- **end:** 8
- **sample:** nil
- **done:** true

---

- **QS1**
  - **64 BPM**
    - **Start:** 2
    - **End:** 2
  - **65 BPM**
    - **Start:** 6
    - **End:** 6

- **QS2**
  - **67 BPM**
    - **Start:** 4
    - **End:** 4
  - **63 BPM**
    - **Start:** 8
    - **End:** 8

---

Time Scale:

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
HKQuantitySeriesSampleQuery

query.includeSample = true

QS1
- 64 BPM
  - Start: 2
  - End: 2
- 65 BPM
  - Start: 6
  - End: 6

QS2
- 67 BPM
  - Start: 4
  - End: 4
- 63 BPM
  - Start: 8
  - End: 8
HKQuantitySeriesSampleQuery

query.includeSample = true

quantity: 64 BPM
start: 2
end: 2
sample: QS1
done: false

64 BPM
Start: 2
End: 2

65 BPM
Start: 6
End: 6

67 BPM
Start: 4
End: 4

63 BPM
Start: 8
End: 8
query.includeSample = true

quantity: 67 BPM
start: 4
end: 4
sample: QS2
done: false

QS1
64 BPM
Start: 2
End: 2

QS2
67 BPM
Start: 4
End: 4

65 BPM
Start: 6
End: 6

63 BPM
Start: 8
End: 8
HKQuantitySeriesSampleQuery

query.includeSample = true

quantity: 65 BPM
start: 6
end: 6
sample: QS1
done: false

QS1

64 BPM
Start: 2
End: 2

65 BPM
Start: 6
End: 6

QS2

67 BPM
Start: 4
End: 4

63 BPM
Start: 8
End: 8

s
HKQuantitySeriesSampleQuery

query.includeSample = true

quantity: 63 BPM
start: 8
end: 8
sample: QS2
done: true

QS1
- 64 BPM
  Start: 2
  End: 2
- 65 BPM
  Start: 6
  End: 6

QS2
- 63 BPM
  Start: 8
  End: 8
- 67 BPM
  Start: 4
  End: 4
Demo
Writing and reading quantity series
New Health Types
Heart and hearing health

Divya Koyyalagunta, Health Software Engineer
Heart Rate
Heart Rate

68 BPM
Heart Rate

Quantity
Type
Sample

Heart Rate

68 BPM

Start: May 29, 2019 10:00:05
End: May 29, 2019 10:00:10
Heart Rate Variability SDNN

![Heart Rate Variability SDNN Graph]
Heart Rate Variability SDNN

- 0.99s
- 1.00s
- 0.99s
- 0.97s
Heart Rate Variability SDNN

0.0109 s
Heart Rate Variability SDNN

Start: May 29, 2019 10:00:05
End: May 29, 2019 10:00:10

0.0109 s
HKHeartbeatSeriesSample

Start: 0
End: 5
UUID: 95-95b0-c2
device: myController
metadata: {"game1"}

0.50
1.49
2.49
3.48
4.45
Heartbeat Series vs. Quantity Series

No values or units like HKQuantities
Heartbeat Series vs. Quantity Series

No values or units like HKQuantities

Series of time stamps
class HKHeartbeatSeriesSample : HKSeriesSample
class HKHeartbeatSeriesSample : HKSeriesSample

class HKHeartbeatSeriesBuilder : HKSeriesBuilder
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 1: Request authorization to read and write heartbeat series type
let healthStore = HKHealthStore()

let dataTypes: Set<HKObjectType> = Set(
    [HKSeriesType.heartbeat(),
     HKQuantityType.quantityType(forIdentifier: .heartRateVariabilitySDNN)!!])

healthStore.requestAuthorization(toShare: dataTypes, read: dataTypes) { (success, error) in
    // Handle error
}
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 1: Request authorization to read and write heartbeat series type
let healthStore = HKHealthStore()

let dataTypes: Set<HKObjectType> = Set([HKSeriesType.heartbeat(), HKQuantityType.quantityType(forIdentifier: .heartRateVariabilitySDNN)!])

healthStore.requestAuthorization(toShare: dataTypes, read: dataTypes) { (success, error) in
  // Handle error
}
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 1: Request authorization to read and write heartbeat series type
let healthStore = HKHealthStore()

let dataTypes: Set<HKObjectType> = Set(
    [HKSeriesType.heartbeat(),
     HKQuantityType.quantityType(forIdentifier: .heartRateVariabilitySDNN)!])

healthStore.requestAuthorization(toShare: dataTypes, read: dataTypes) { (success, error) in
    // Handle error
}
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 2: Initialize builder, add heartbeats as they come in
let builder = HKHeartbeatSeriesBuilder(healthStore: healthStore,
                                         device: gameDevice,
                                         start: gameStartDate)

// while game is ongoing ...
builder.addHeartbeatWithTimeInterval(sinceSeriesStartDate: 0.5, precededByGap: NO) {
    (success, error) in
    // Handle error
}
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 2: Initialize builder, add heartbeats as they come in
let builder = HKHeartbeatSeriesBuilder(healthStore: healthStore,
    device: gameDevice,
    start: gameStartDate)

// while game is ongoing ...
builder.addHeartbeatWithTimeInterval(sinceSeriesStartDate: 0.5, precededByGap: NO) {
    (success, error) in
    // Handle error
}

// Handle error
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 2: Initialize builder, add heartbeats as they come in
let builder = HKHeartbeatSeriesBuilder(healthStore: healthStore,
                                       device: gameDevice,
                                       start: gameStartDate)

// while game is ongoing ...
builder.addHeartbeatWithTimeInterval(sinceSeriesStartDate: 0.5, precededByGap: NO) {
    (success, error) in
    // Handle error
}"
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 2: Initialize builder, add heartbeats as they come in
let builder = HKHeartbeatSeriesBuilder(healthStore: healthStore,
device: gameDevice,
start: gameStartDate)

// while game is ongoing ...
builder.addHeartbeatWithTimeInterval(sinceSeriesStartDate: 3.48, precededByGap: YES) {
    (success, error) in
    // Handle error
}
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 4: Add metadata and finish the series
let metadata: [String: Any] = [
    "gameName": "WWDC Heart Adventure"
]
builder.addMetadata(metadata) {
    (success, error) in
    // Handle error
}

builder.finishSeries() {
    (heartbeatSeriesSample, error) in
    // If no error, heartbeatSeriesSample was created and saved in the database by the builder
}
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// Step 4: Add metadata and finish the series

let metadata: [String: Any] = [
    "gameName": "WWDC Heart Adventure"
]

builder.addMetadata(metadata) {
    (success, error) in
    // Handle error
}

builder.finishSeries() {
    (heartbeatSeriesSample, error) in
    // If no error, heartbeatSeriesSample was created and saved in the database by the builder
}
// HKHeartbeatSeriesSample
class HKHeartbeatSeriesSample : HKSeriesSample

// HKHeartbeatSeriesBuilder
class HKHeartbeatSeriesBuilder : HKSeriesBuilder

// HKHeartbeatSeriesQuery
class HKHeartbeatSeriesQuery : HKQuery
// Querying for Beat-to-Beat Measurements with HKHeartbeatSeriesQuery

// Step 1: Assuming authorization is granted, run a query to fetch an HKHeartbeatSeriesSample
let heartbeatSeriesSample = …
// Querying for Beat-to-Beat Measurements with HKHeartbeatSeriesQuery

// Step 1: Assuming authorization is granted, run a query to fetch an HKHeartbeatSeriesSample
let heartbeatSeriesSample = ... 

// Step 2: Given an HKHeartbeatSeriesSample, fetch the associated heartbeat data
let query = HKHeartbeatSeriesQuery(heartbeatSeries: heartbeatSeriesSample) {
    (query, timeSinceSeriesStart, precededByGap, done, error) in
    guard error == nil else {
        // Handle error
    }
    // Read timeSinceSeriesStart and precededByGap
}
// Querying for Beat-to-Beat Measurements with HKHeartbeatSeriesQuery

// Step 1: Assuming authorization is granted, run a query to fetch an HKHeartbeatSeriesSample
let heartbeatSeriesSample = …

// Step 2: Given an HKHeartbeatSeriesSample, fetch the associated heartbeat data
let query = HKHeartbeatSeriesQuery(heartbeatSeries: heartbeatSeriesSample) {
    (query, timeSinceSeriesStart, precededByGap, done, error) in
    guard error == nil else {
        // Handle error
    }
    // Read timeSinceSeriesStart and precededByGap
}

// Step 3: Run the query
healthStore.execute(query)
HealthKit data model
New quantity series APIs
Beat-to-beat heart measurements
Heart rate events
Hearing health
Heart Rate Events

Low Heart Rate
Your heart rate fell below 40 BPM for 10 minutes starting at 9:59 AM.
Heart Rate Events

Low Heart Rate
Your heart rate fell below 40 BPM for 10 minutes starting at 9:59 AM.

High Heart Rate
Your heart rate rose above 120 BPM while you seemed to be inactive for 10 minutes starting at 9:59 AM.
Heart Rate Events

Low Heart Rate
Your heart rate fell below 40 BPM for 10 minutes starting at 9:59 AM.

High Heart Rate
Your heart rate rose above 120 BPM while you seemed to be inactive for 10 minutes starting at 9:59 AM.

Your heart has shown signs of an irregular rhythm suggestive of atrial fibrillation.
If you have not been diagnosed with AFib by a physician, you...
Heart Rate Events
HKCategoryTypeIdentifiers

1. lowHeartRateEvent
2. highHeartRateEvent
3. irregularHeartRhythmEvent
HealthKit data model
New quantity series APIs
Beat-to-beat heart measurements
Heart rate events
Hearing health
Pure-Tone Testing

Quietest sound you can hear at different frequencies
Pure-Tone Testing

Quietest sound you can hear at different frequencies

Assesses hearing impairment
Pure-Tone Testing

Quietest sound you can hear at different frequencies

Assesses hearing impairment

Results displayed in graphs called audiograms
Mild Impairment

May 19, 2019  10:30 AM

About Audiogram

An audiogram displays the results of a hearing test.
// Creating an HKAudiogramSample

// Step 1: Create an HKAudiogramSensitivityPoint with the left and right ear sensitivity values associated with a single frequency
let frequency = HKQuantity(unit: HKUnit.hertz(), doubleValue: 250.0)

let leftSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 34.0)
let rightSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 27.0)

let sensitivityPoint = HKAudiogramSensitivityPoint(frequency: frequency,
                                                   leftEarSensitivity: leftSensitivity,
                                                   rightEarSensitivity: rightSensitivity)
// Creating an HKAudiogramSample

// Step 1: Create an HKAudiogramSensitivityPoint with the left and right ear sensitivity values associated with a single frequency

let frequency = HKQuantity(unit: HKUnit.hertz(), doubleValue: 250.0)

let leftSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 34.0)
let rightSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 27.0)

let sensitivityPoint = HKAudiogramSensitivityPoint(frequency: frequency, 
leftEarSensitivity: leftSensitivity,
rightEarSensitivity: rightSensitivity)
// Creating an HKAudiogramSample

// Step 1: Create an HKAudiogramSensitivityPoint with the left and right ear sensitivity values associated with a single frequency
let frequency = HKQuantity(unit: HKUnit.hertz(), doubleValue: 250.0)

let leftSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 34.0)
let rightSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 27.0)

let sensitivityPoint = HKAudiogramSensitivityPoint(frequency: frequency,
                                                  leftEarSensitivity: leftSensitivity,
                                                  rightEarSensitivity: rightSensitivity)
// Creating an HKAudiogramSample

// Step 1: Create an HKAudiogramSensitivityPoint with the left and right ear sensitivity values associated with a single frequency
let frequency = HKQuantity(unit: HKUnit.hertz(), doubleValue: 250.0)
let leftSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 34.0)
let rightSensitivity = HKQuantity(unit: HKUnit.decibelHearingLevel(), doubleValue: 27.0)

let sensitivityPoint = HKAudiogramSensitivityPoint(frequency: frequency, leftEarSensitivity: leftSensitivity, rightEarSensitivity: rightSensitivity)
// Creating an HKAudiogramSample

// Step 2: Create an HKAudiogramSample with an array of sensitivity points

let audiogramSample = HKAudiogramSample(sensitivityPoints: [sensitivityPoint],
                                        start: startDate,
                                        end: endDate,
                                        metadata: nil)
// Creating an HKAudiogramSample

// Step 2: Create an HKAudiogramSample with an array of sensitivity points
let audiogramSample = HKAudiogramSample(sensitivityPoints: [sensitivityPoint],
                                        start: startDate,
                                        end: endDate,
                                        metadata: nil)

// Step 3: Save to HKHealthStore
healthStore.save(object: audiogramSample) { (success, error) in
    // Handle error
}
Audio Exposure
HKQuantityTypeIdentifiers

Headphone Audio Levels

Average: 96 dB

Loud
May 14 – 21, 2019

Show All Filters

Highlights

Show All

Headphone Audio Levels
Audio Exposure
HKQuantityTypeIdentifiers
Noise
110 dB
Loud
Repeated, long-term exposure to sounds at this level can damage your
HealthKit data model
New quantity series APIs
Beat-to-beat heart measurements
Heart rate events
Hearing health
Summary

Efficiently store large numbers of HKQuantities
Summary

Efficiently store large numbers of HKQuantities

New data representations for heart and hearing health
More Information

developer.apple.com/wwdc19/218