#WWDC19

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

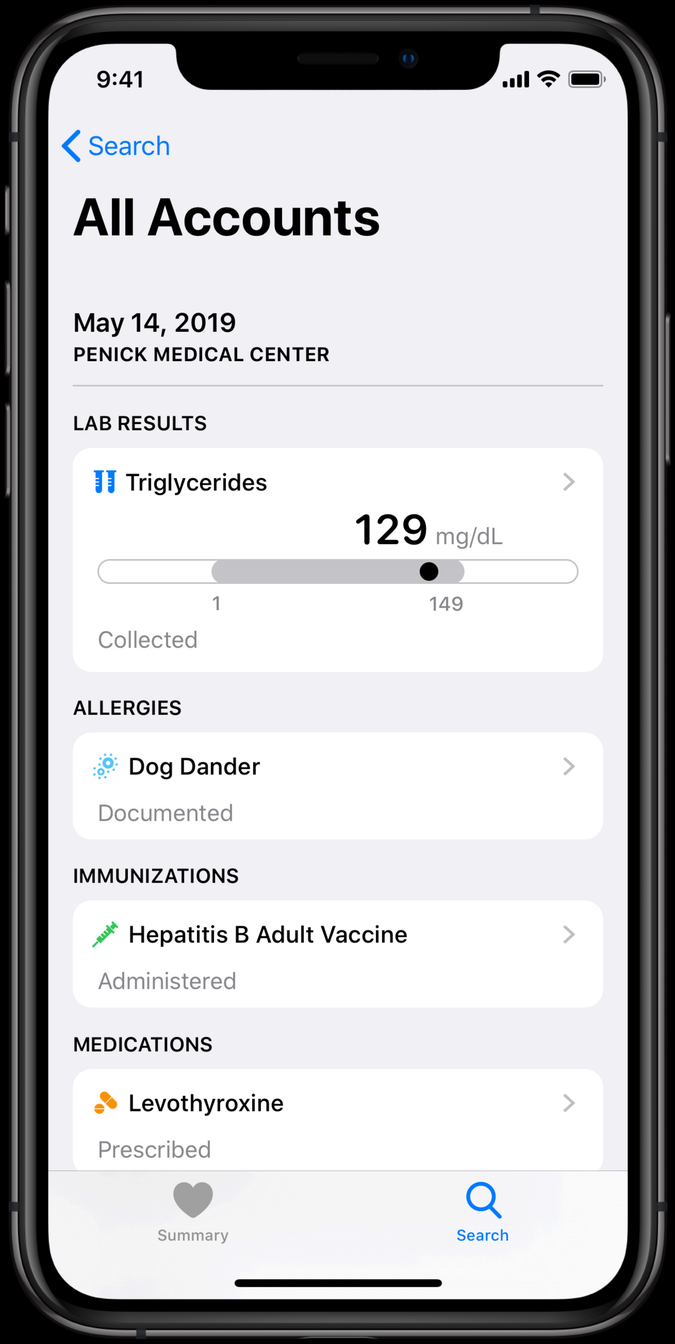
HealthKit Data Model

Reviewing samples

Measurements at a particular time

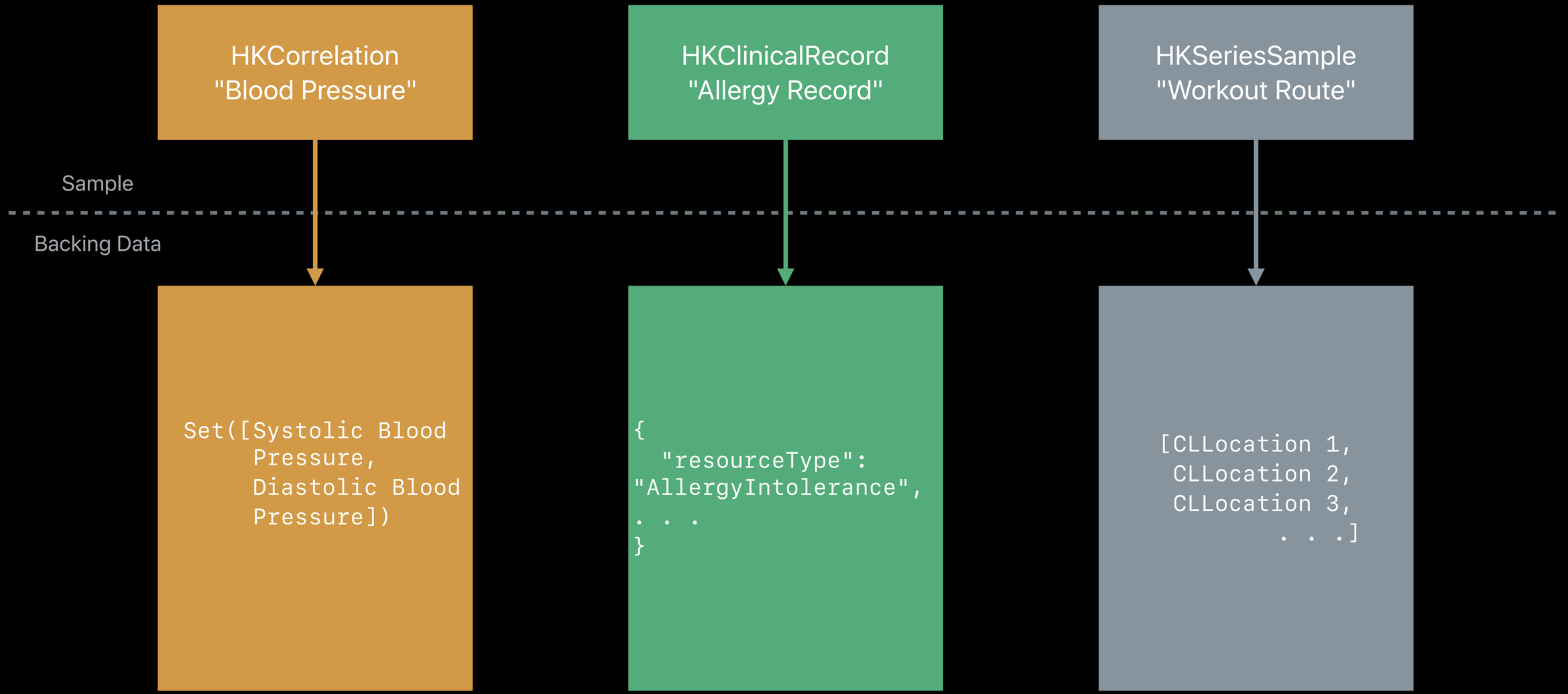
Occur over a time interval


29:31


9:41
Search
All Accounts
May 14, 2019
PENICK MEDICAL CENTER
LAB RESULTS
Triglycerides
129 mg/dL
1
149
Collected
ALLERGIES
Dog Dander
Documented
IMMUNIZATIONS
Hepatitis B Adult Vaccine
Administered
MEDICATIONS
Levothyroxine
Prescribed
Summary
Search

HealthKit Data Model

Models for Rich Data



HealthKit data model

New quantity series APIs

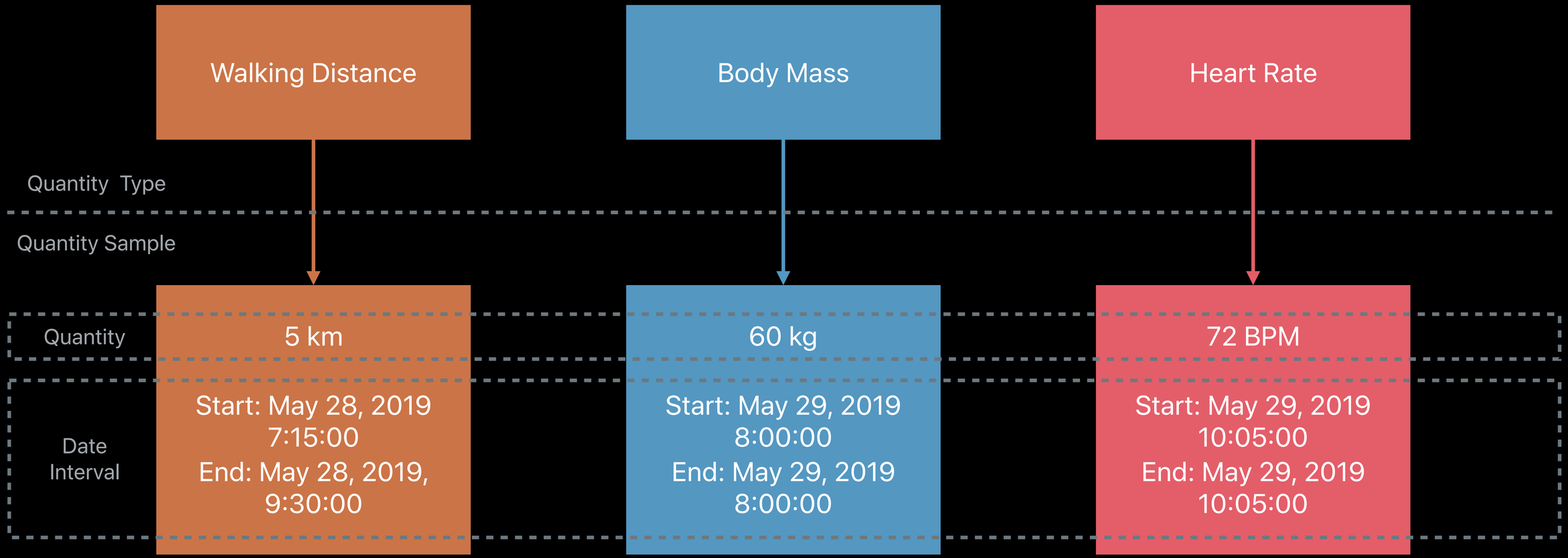
Beat-to-beat heart measurements

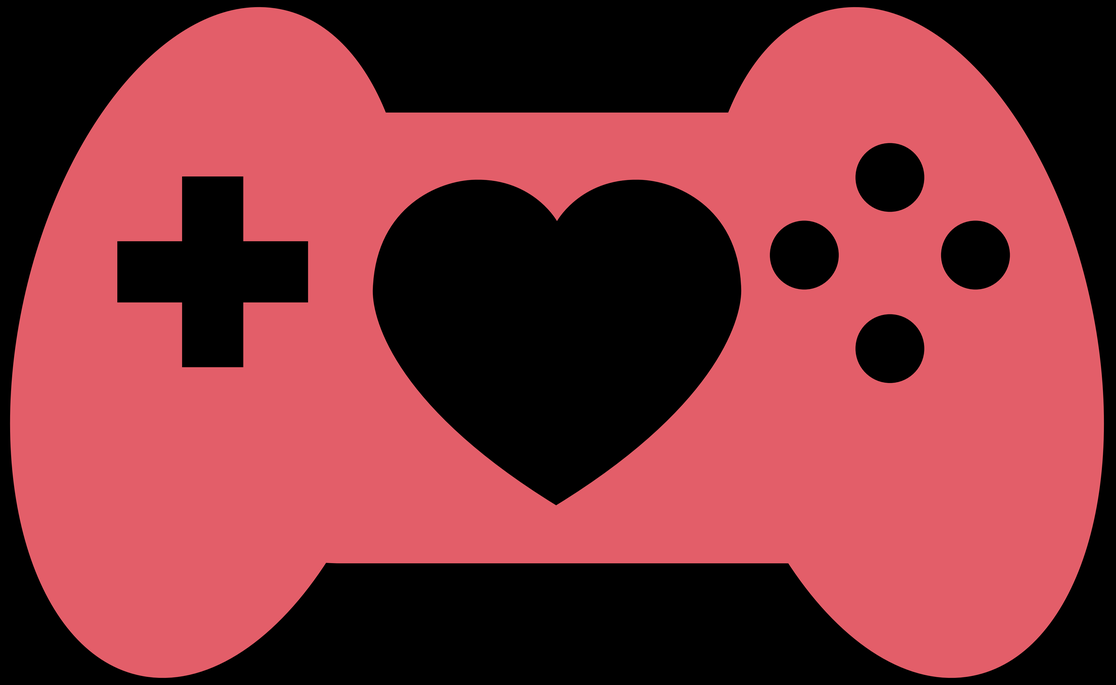
Heart rate events

Hearing health

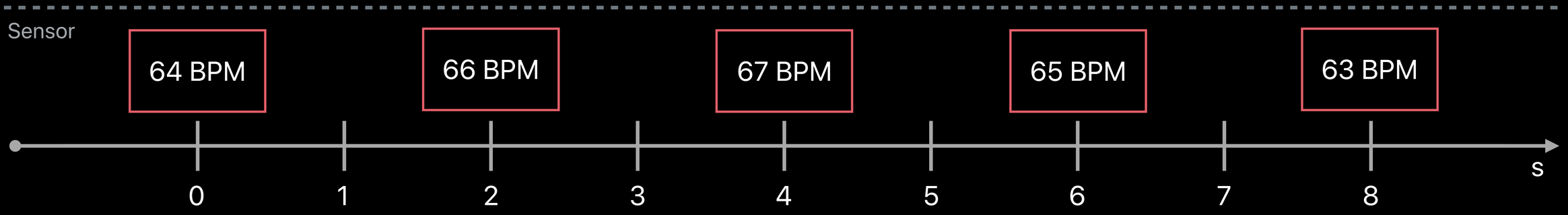
Quantity Data

Components of Quantity Data





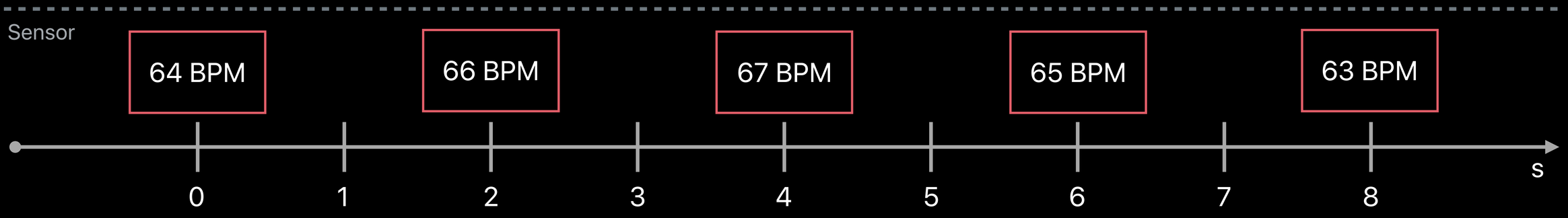
Representing Multiple Quantities



Representing Multiple Quantities

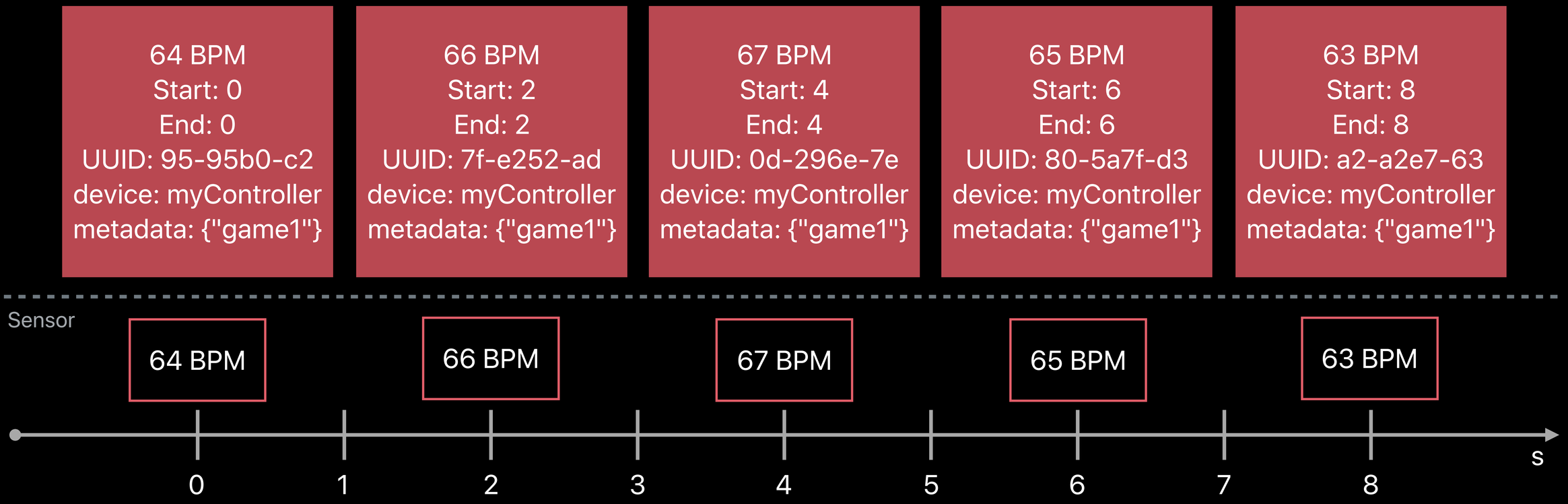
Single quantity sample

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



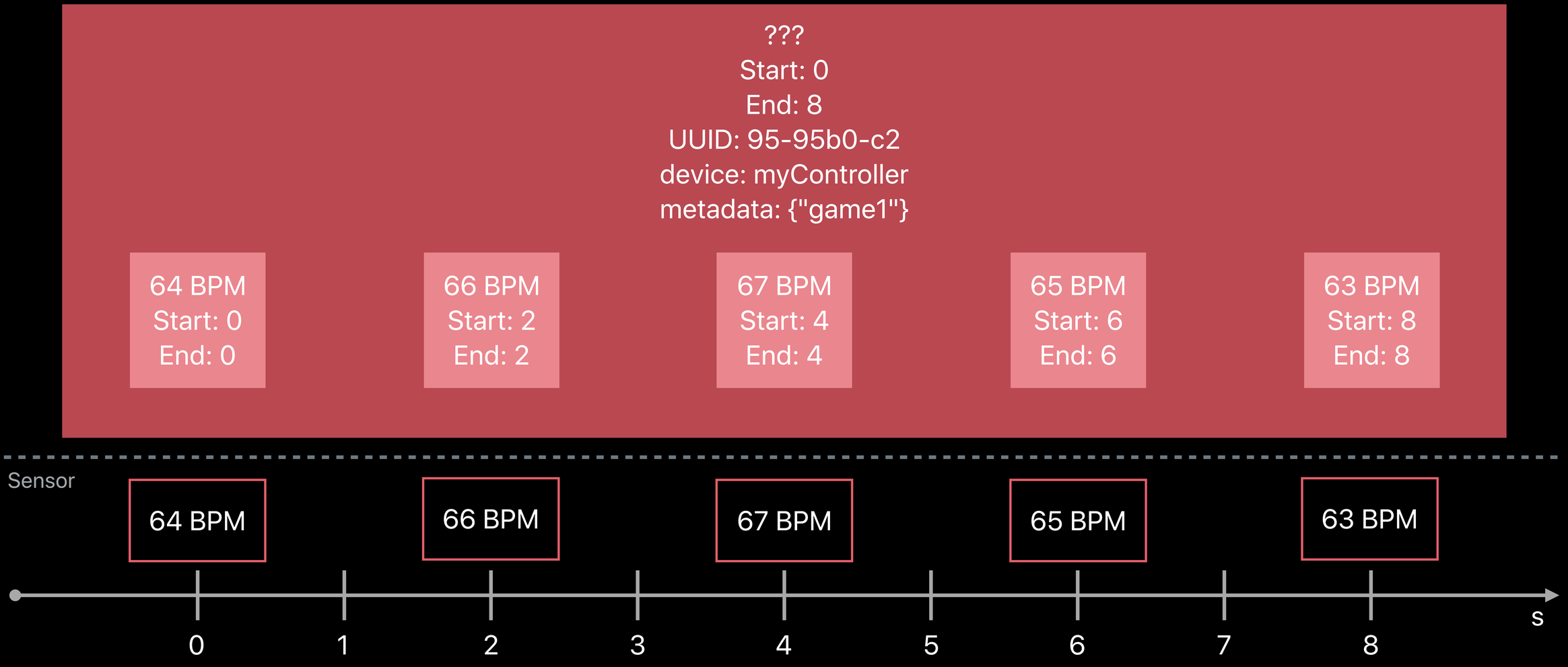
Representing Multiple Quantities

Multiple quantity samples



Representing Multiple Quantities

Quantity series



Aggregating Multiple Quantities

Aggregating Multiple Quantities

Aggregation Style
Cumulative
Discrete

Aggregating Multiple Quantities

Aggregation Style
Cumulative
Discrete

Aggregating Multiple Quantities

Aggregation Style	Example Types
Cumulative	Distance Calories Steps
Discrete	

Aggregating Multiple Quantities

Aggregation Style	Example Types	Statistics
Cumulative	Distance Calories Steps	Sum
Discrete		

Aggregating Multiple Quantities

Aggregation Style	Example Types	Statistics
Cumulative	Distance Calories Steps	Sum
Discrete		

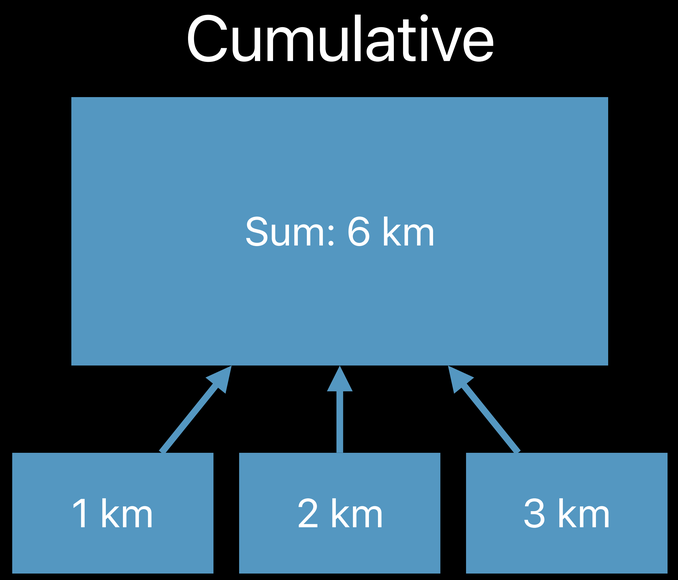
Aggregating Multiple Quantities

Aggregation Style	Example Types	Statistics
Cumulative	Distance Calories Steps	Sum
Discrete	Heart Rate Body Mass Height	

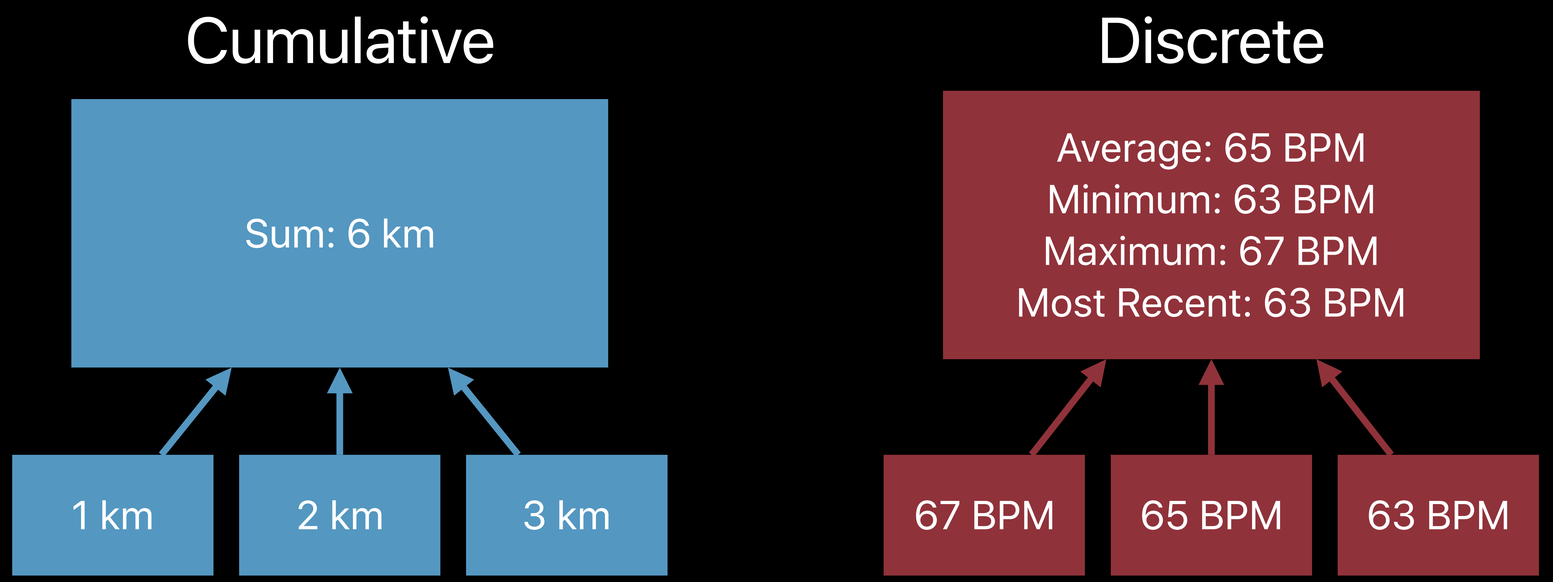
Aggregating Multiple Quantities

Aggregation Style	Example Types	Statistics
Cumulative	Distance Calories Steps	Sum
Discrete	Heart Rate Body Mass Height	Average Minimum Maximum Most Recent

Aggregating Multiple Quantities



Aggregating Multiple Quantities



```
// Aggregating Multiple Quantities

enum HKQuantityAggregationStyle : Int {
  case cumulative
  case discrete
}
```

```
// Aggregating Multiple Quantities

enum HKQuantityAggregationStyle : Int {
  case cumulative
  case discrete
}
```

```
// Aggregating Multiple Quantities

enum HKQuantityAggregationStyle : Int {
  case cumulative
  case discrete
}
```

```
// Aggregating Multiple Quantities

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

NEW

```
// Aggregating Multiple Quantities

enum HKQuantityAggregationStyle : Int {
  case cumulative
  case discrete
  case discreteArithmetic
  case discreteTemporallyWeighted
}
```

NEW

```
// Aggregating Multiple Quantities

enum HKQuantityAggregationStyle : Int {
  case cumulative
  case discrete
  case discreteArithmetic
  case discreteTemporallyWeighted
  case discreteEquivalentContinuousLevel
}
```

NEW

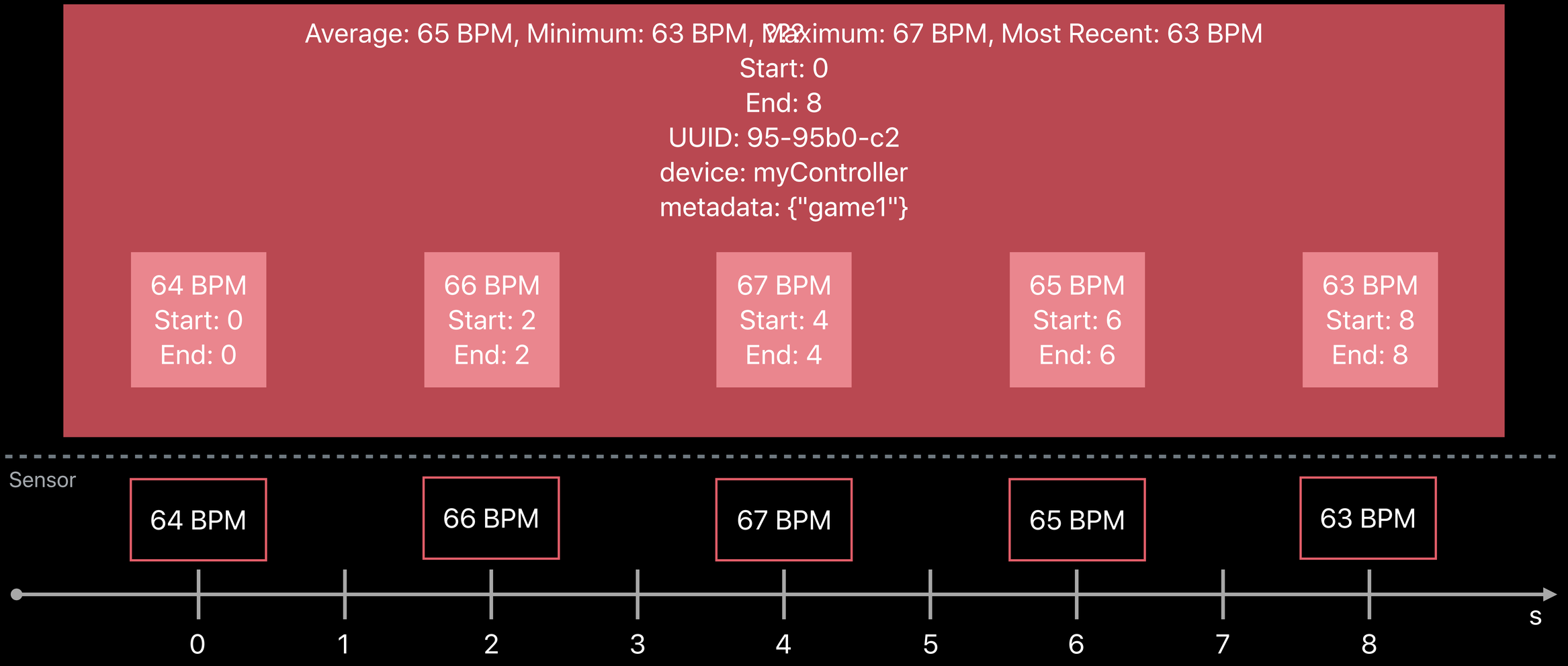
NEW

```
// 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 }
}
```

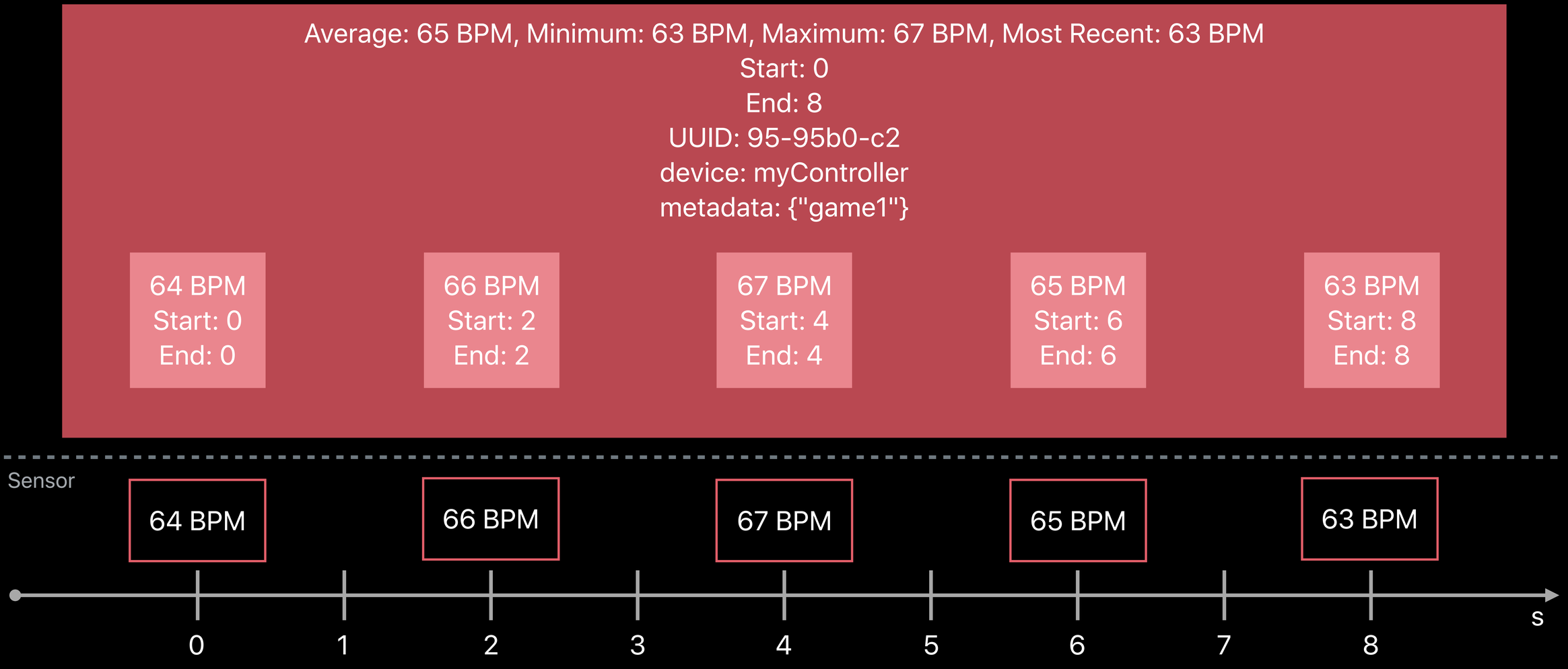
NEW

```
// 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



Building a Quantity Series



```
// 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)
```

```swift
// 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

Querying for Quantity Data

Total active calories burned

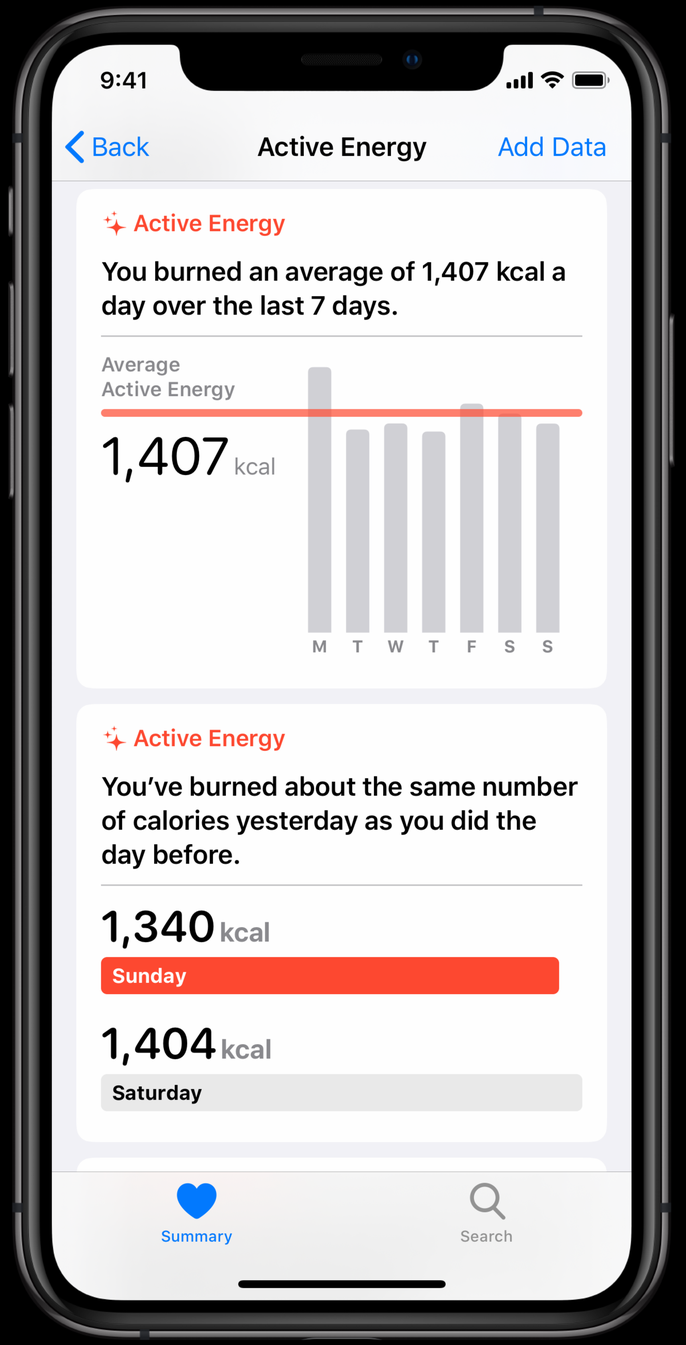
Data for charts

Querying for Quantity Data

Total active calories burned

Data for charts

Average calories burned per day



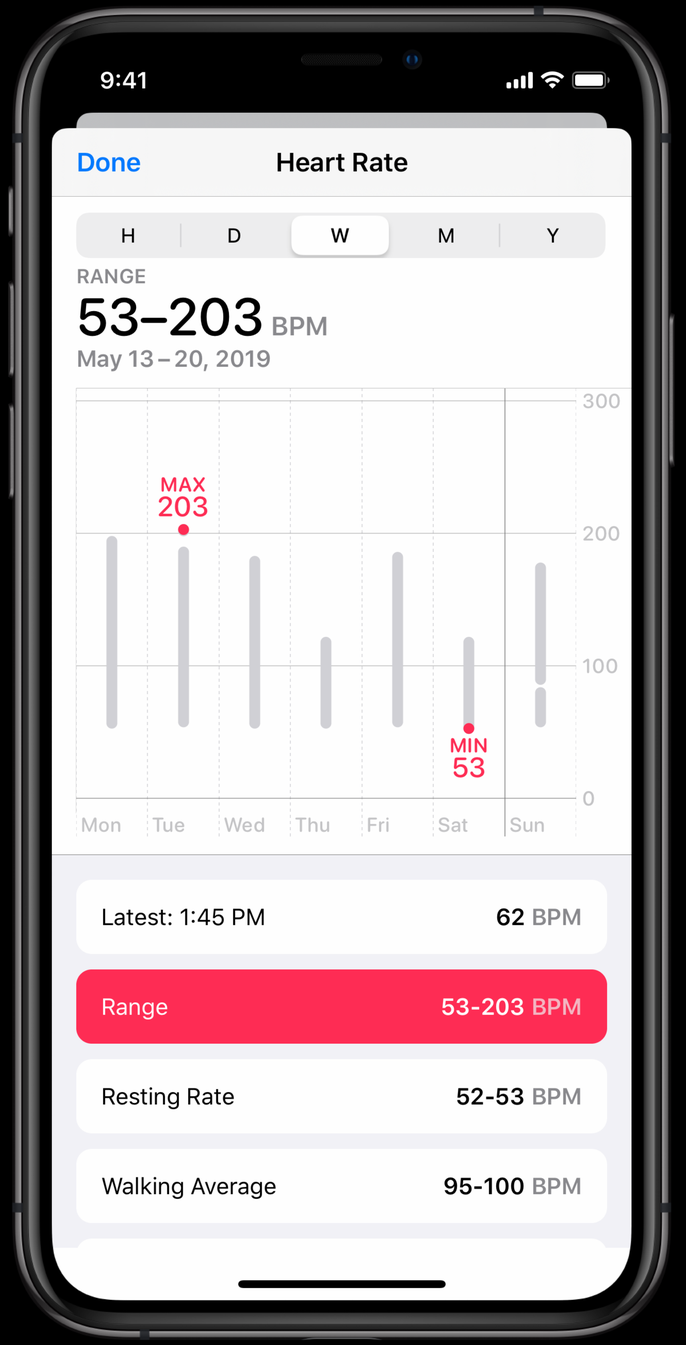
Querying for Quantity Data

Total active calories burned

Data for charts

Average calories burned per day

Min and max heart rate



Querying for Quantity Data

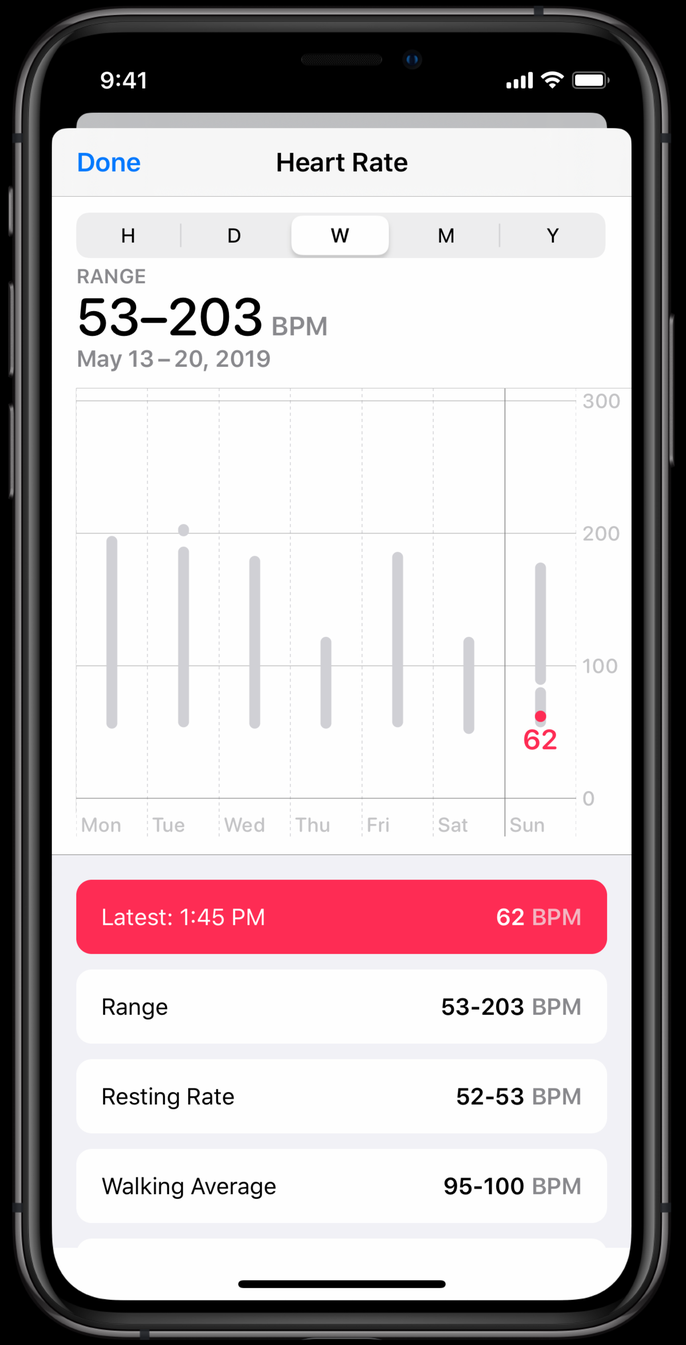
Total active calories burned

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

```
HKQuantitySeriesSampleQuery(quantitySample: HKQuantitySample,
                            quantityHandler: (HKQuantitySeriesSampleQuery,
                                              HKQuantity?,
                                              Date?,
                                              Bool,
                                              Error?) -> Void)
```

HKQuantitySeriesSampleQuery

NEW

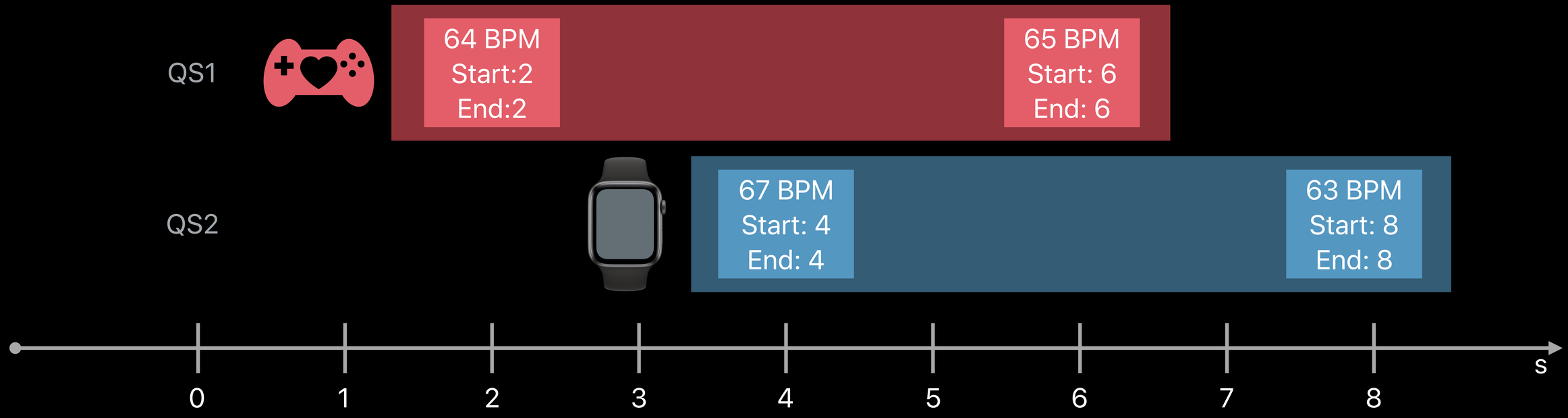
```
HKQuantitySeriesSampleQuery(quantityType: HKQuantityType,
                            predicate: NSPredicate?,
                            quantityHandler: (HKQuantitySeriesSampleQuery,
                             HKQuantity?,
                             Date?,
                             Bool,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

HKQuantitySeriesSampleQuery

NEW

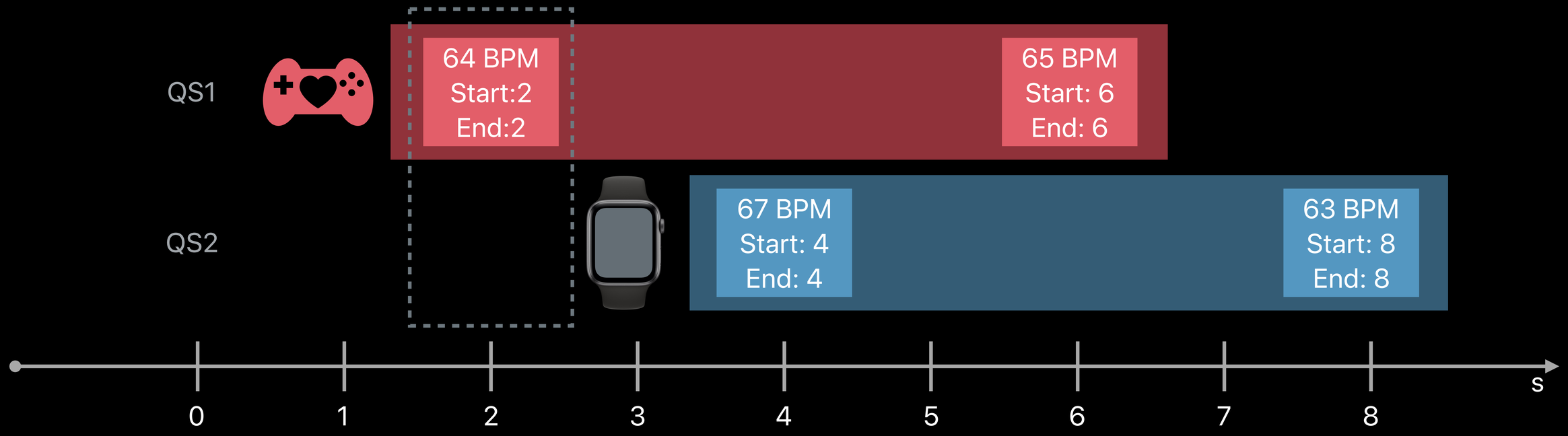
```
HKQuantitySeriesSampleQuery(quantityType: HKQuantityType,
                            predicate: NSPredicate?,
                            quantityHandler: (HKQuantitySeriesSampleQuery,
                                              HKQuantity?,
                                              DateInterval?,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

HKQuantitySeriesSampleQuery



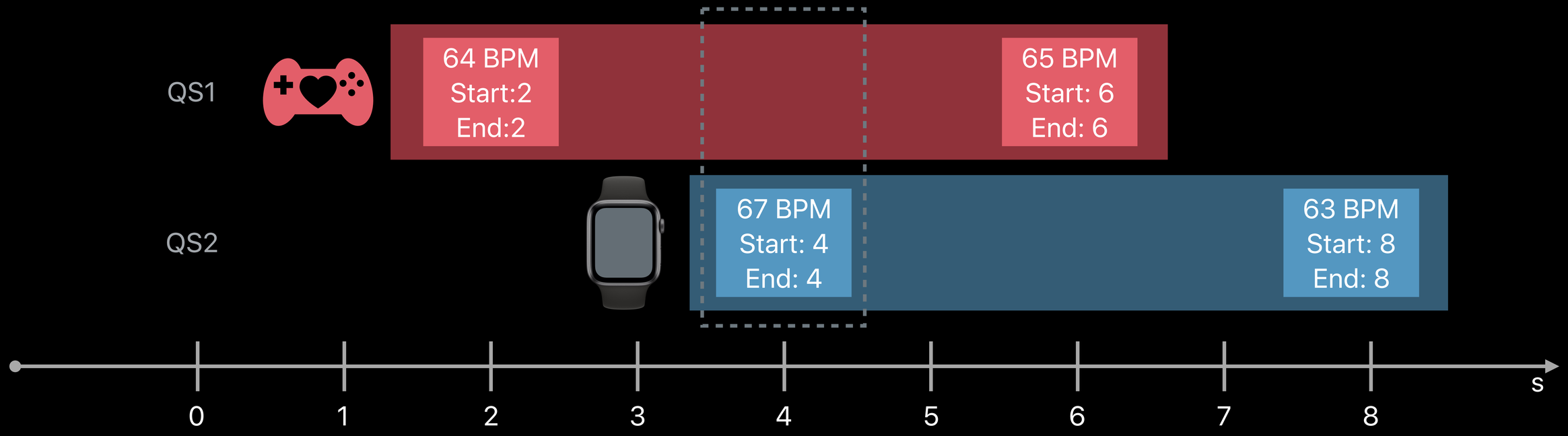
HKQuantitySeriesSampleQuery

```
quantity:   64 BPM
start:      2
end:        2
sample:     nil
done:       false
```



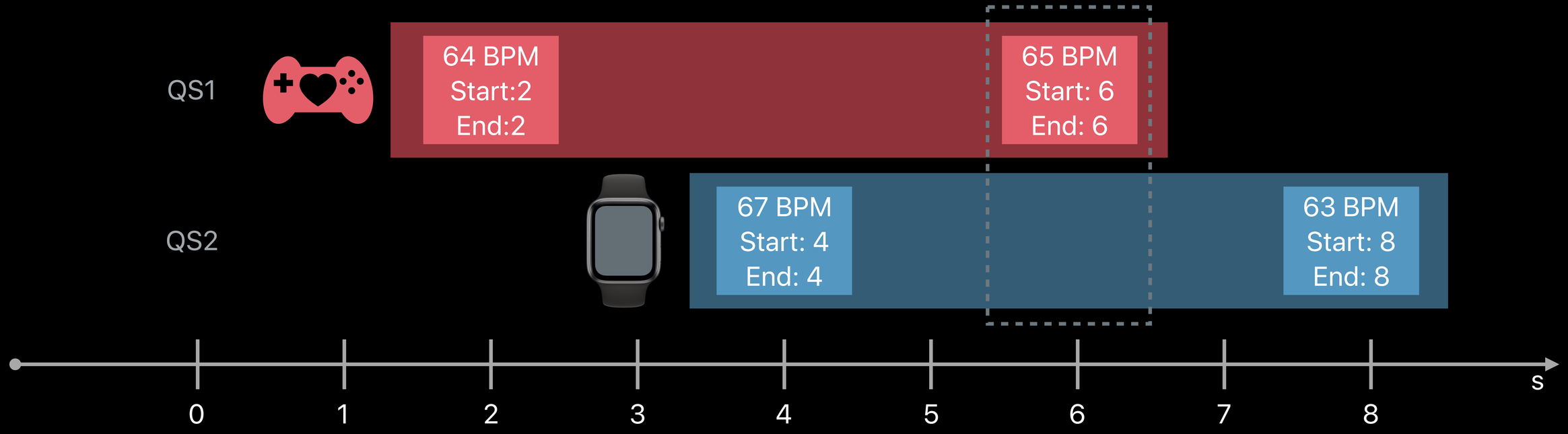
HKQuantitySeriesSampleQuery

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



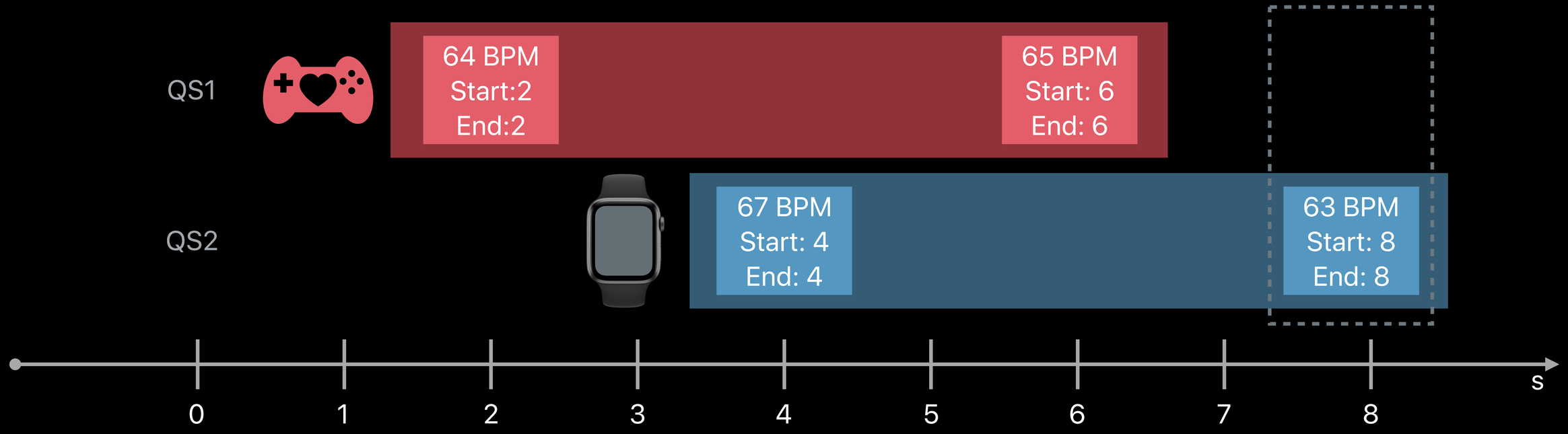
HKQuantitySeriesSampleQuery

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



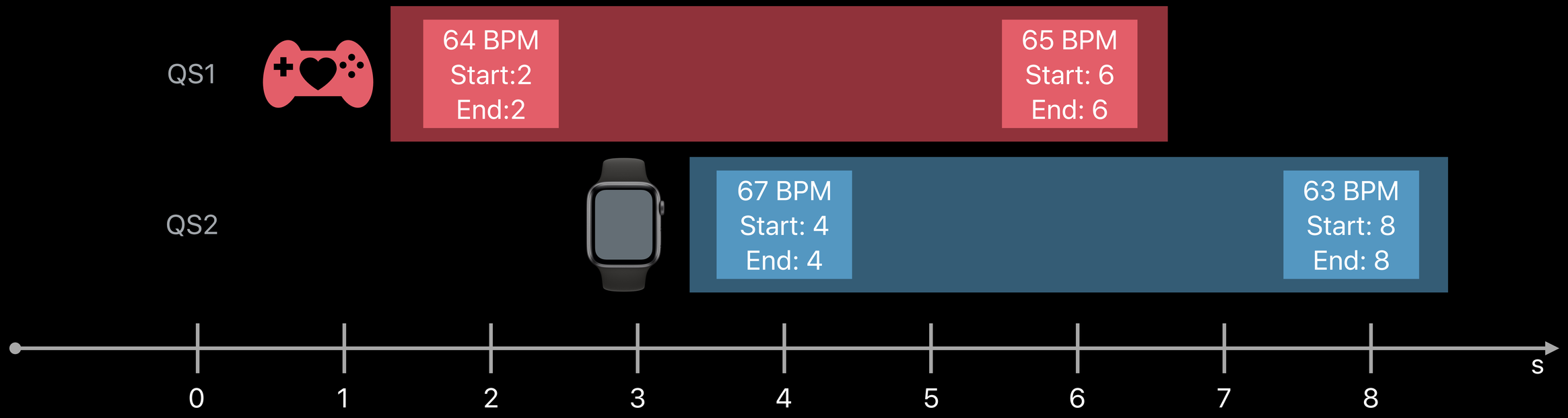
HKQuantitySeriesSampleQuery

```
quantity:   63 BPM
start:      8
end:        8
sample:     nil
done:       true
```



HKQuantitySeriesSampleQuery

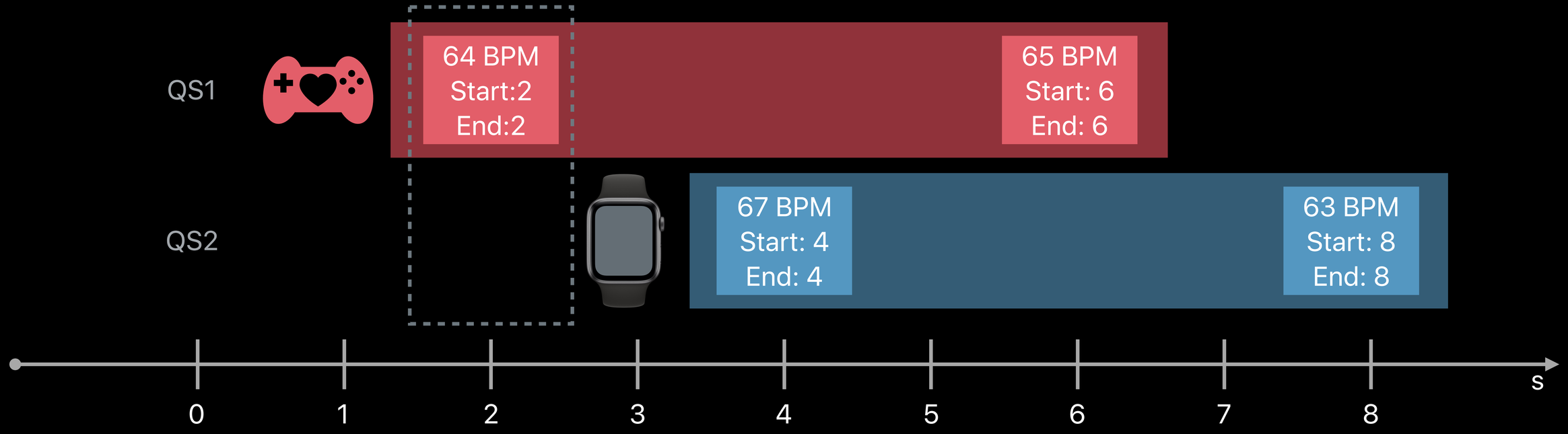
```
query.includeSample = true
```



HKQuantitySeriesSampleQuery

```
query.includeSample = true
```

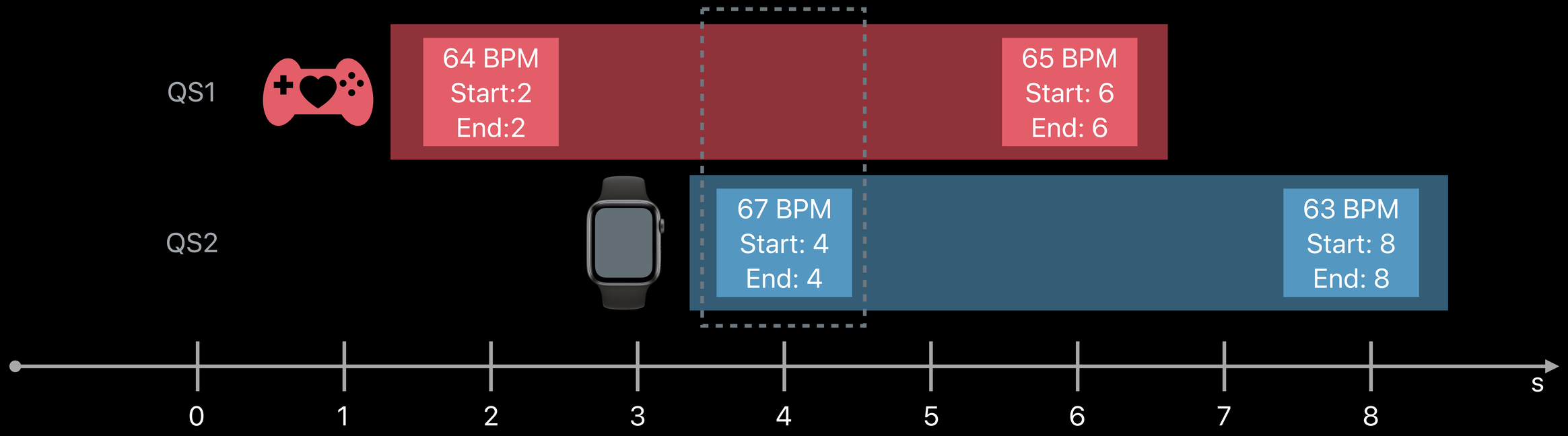
```
quantity:   64 BPM
start:      2
end:        2
sample:     QS1
done:       false
```



HKQuantitySeriesSampleQuery

```
query.includeSample = true
```

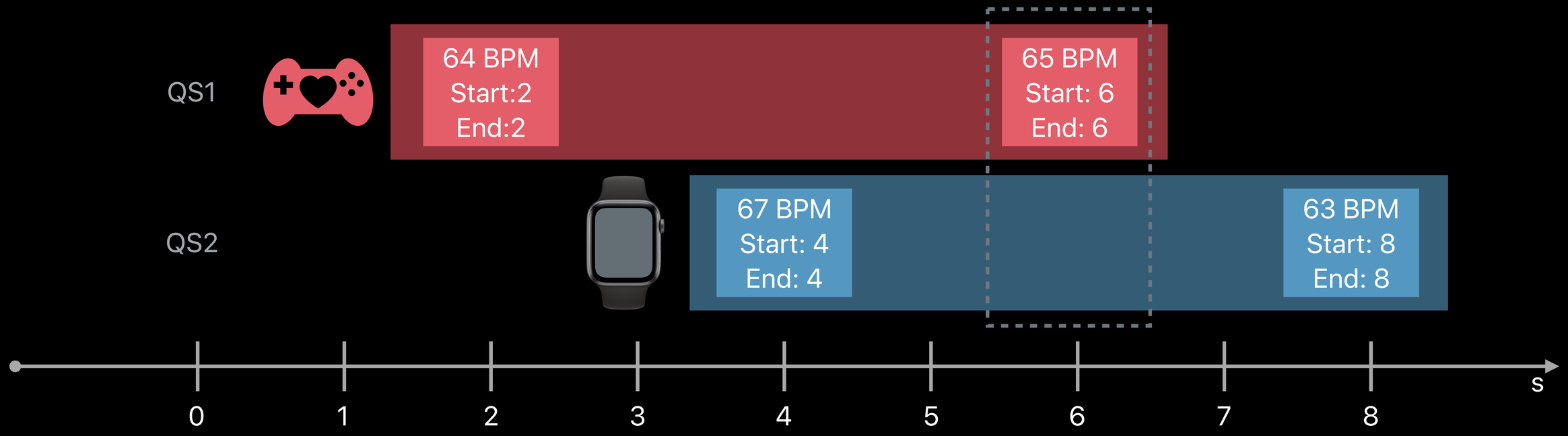
```
quantity:   67 BPM
start:      4
end:        4
sample:     QS2
done:       false
```



HKQuantitySeriesSampleQuery

```
query.includeSample = true
```

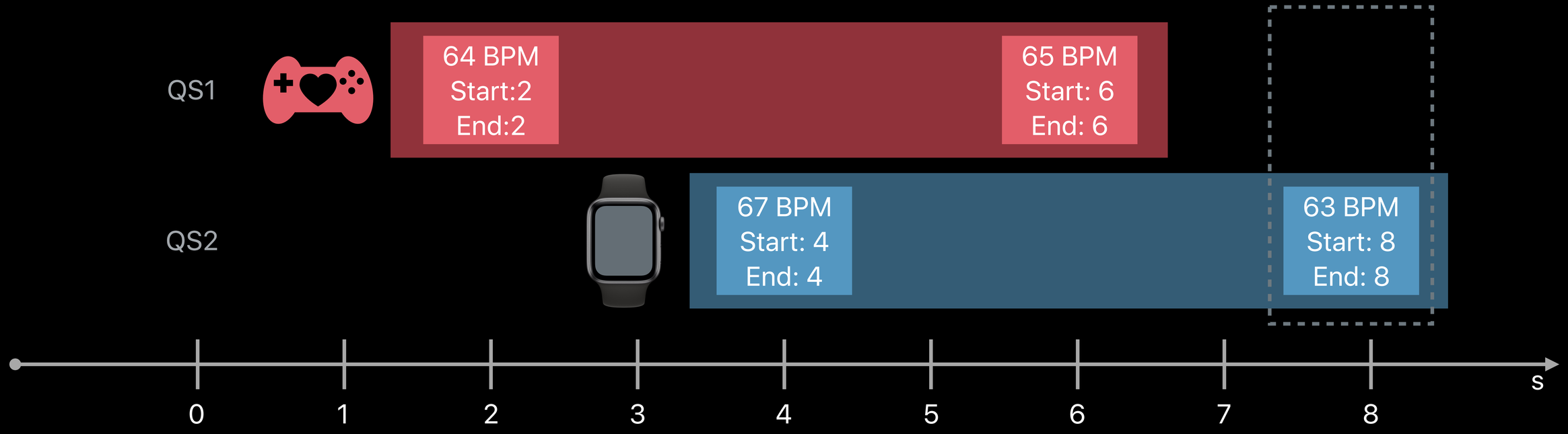
```
quantity:   65 BPM
start:      6
end:        6
sample:     QS1
done:       false
```



HKQuantitySeriesSampleQuery

```
query.includeSample = true
```

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



Demo

Writing and reading quantity series

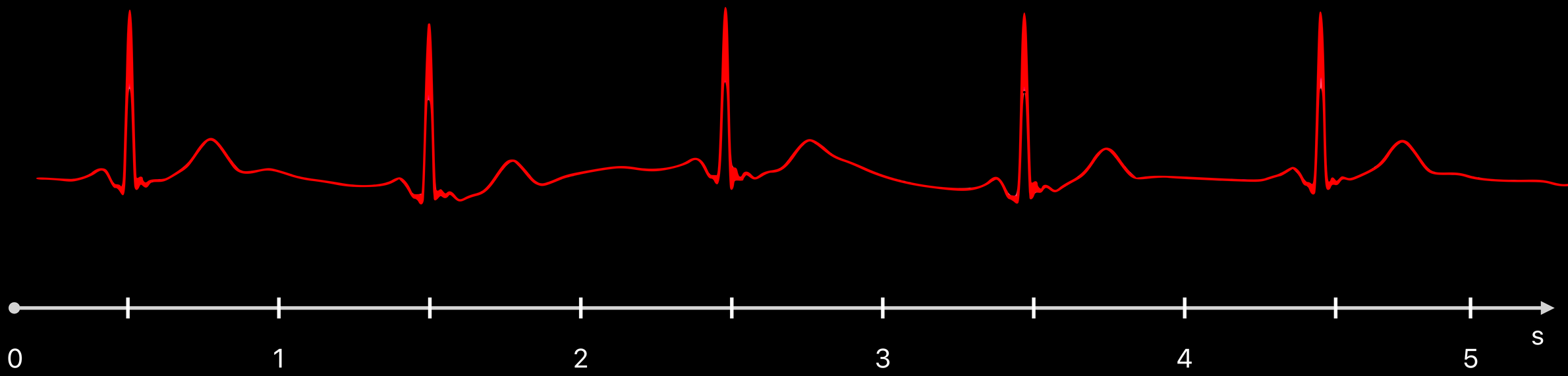
New Health Types

Heart and hearing health

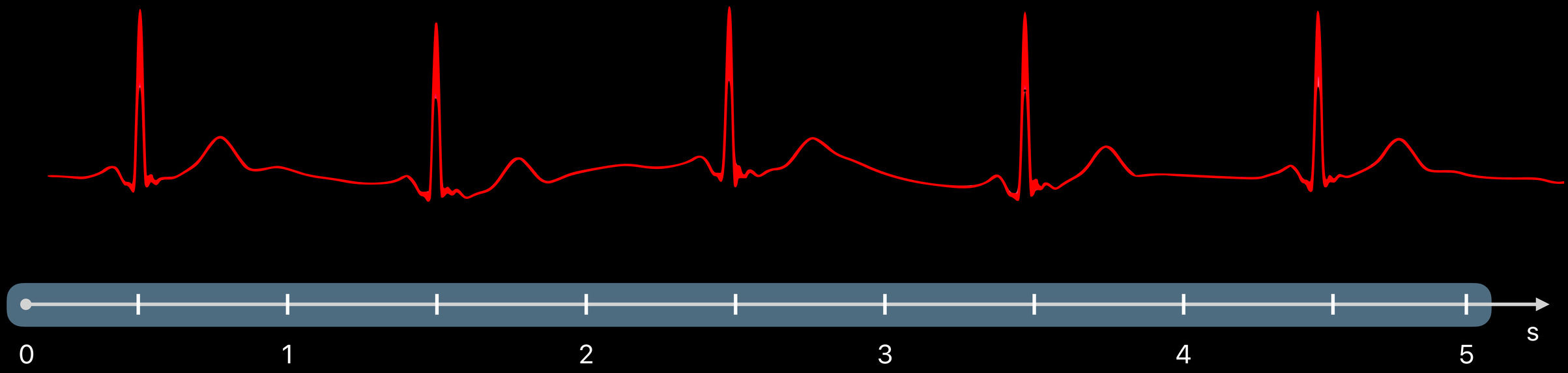
Divya Koyyalagunta, Health Software Engineer


10:09
Done
Heart Rate
H
D
W
M
Y
RANGE
53–203 BPM
May 13 – 20, 2019
300
200
MAX
203
MIN
53
Mon
Tue
Wed
Thu
Fri
Sat
Latest: 1:45 PM
Range
53
Resting Rate
Walking Average
Current
10:09
190
31
12AM
6
12PM
6
Heart Rate
77 BPM
53 BPM, 1m ago

Heart Rate

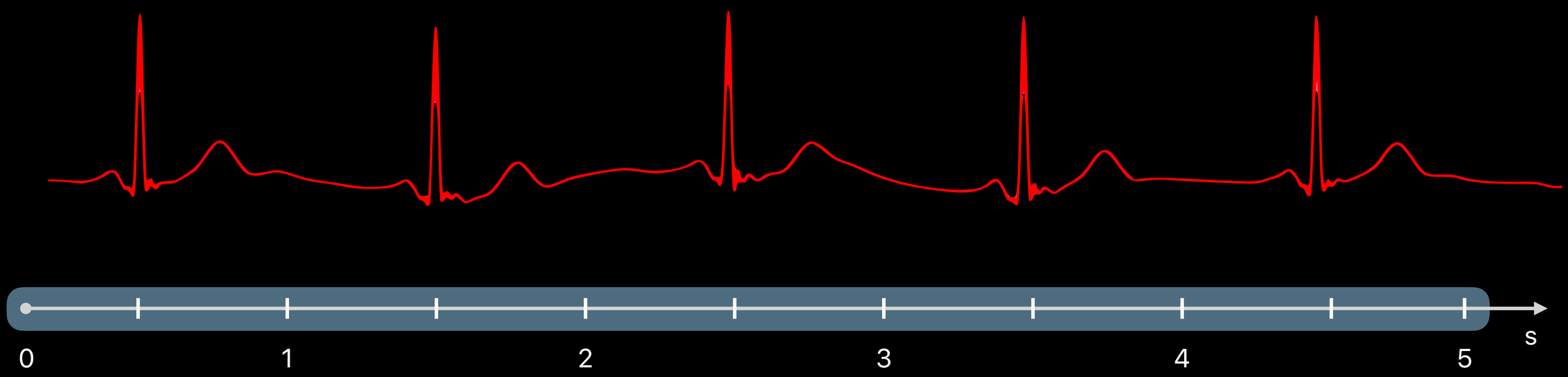


Heart Rate

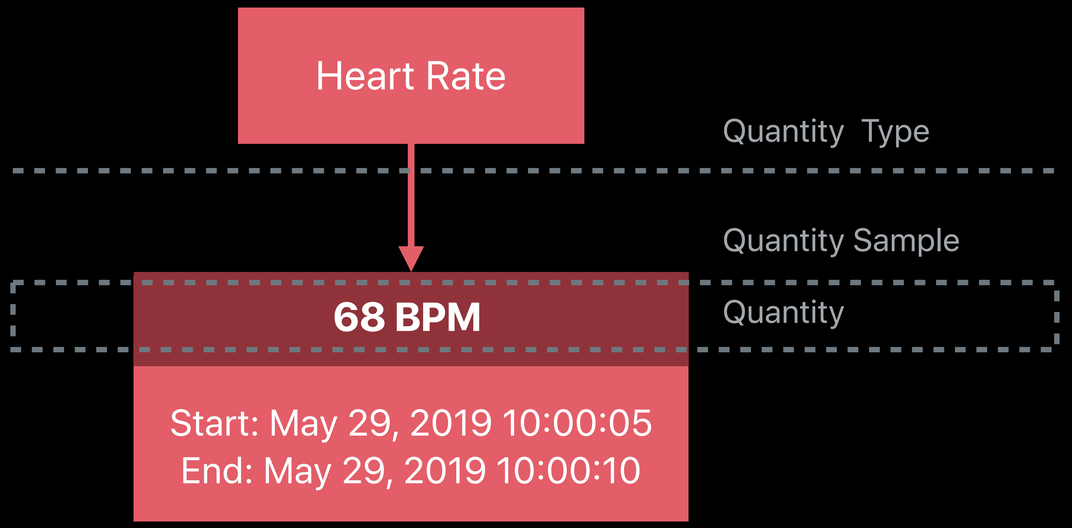


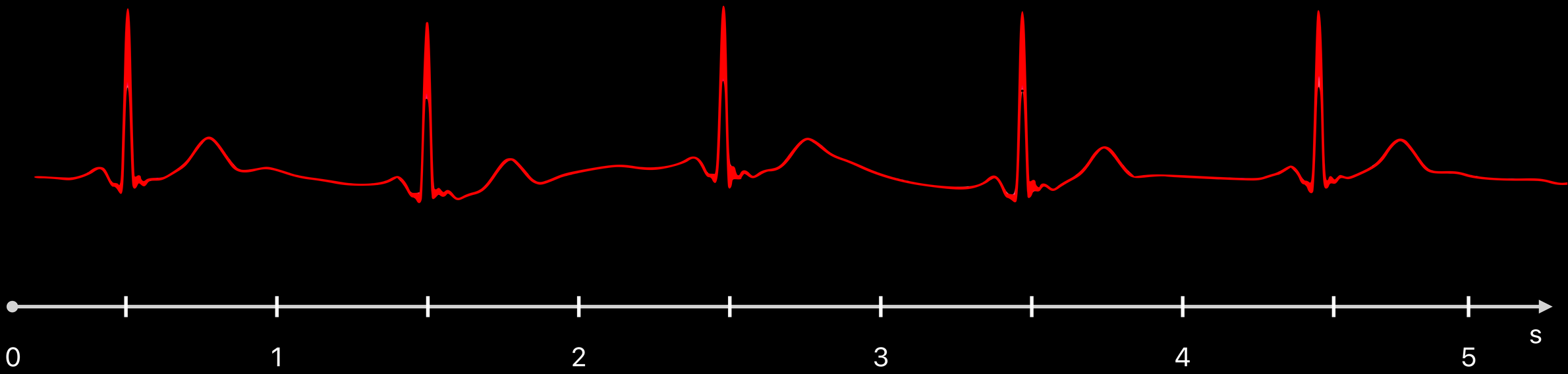
Heart Rate

68 BPM

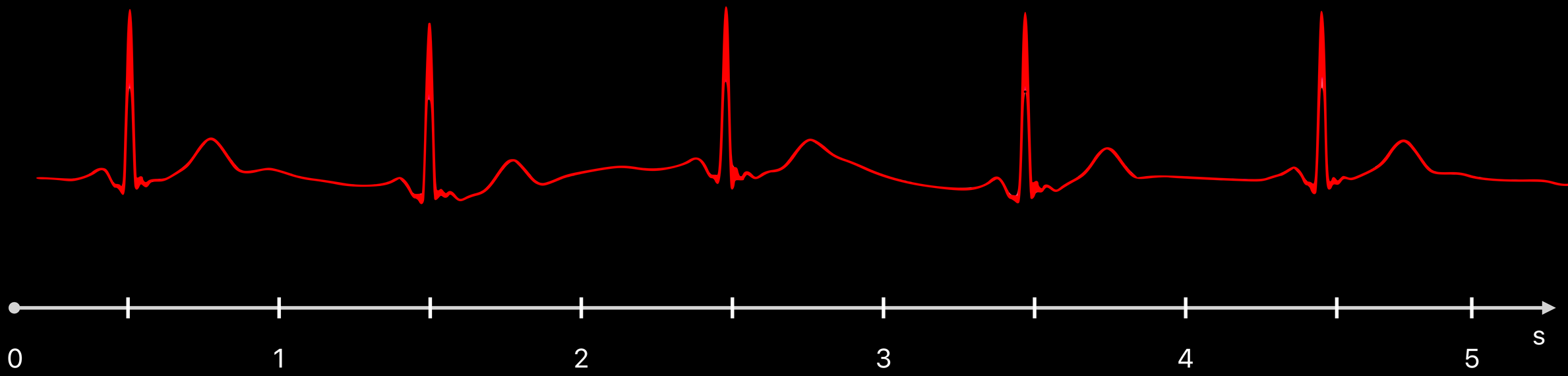


Heart Rate

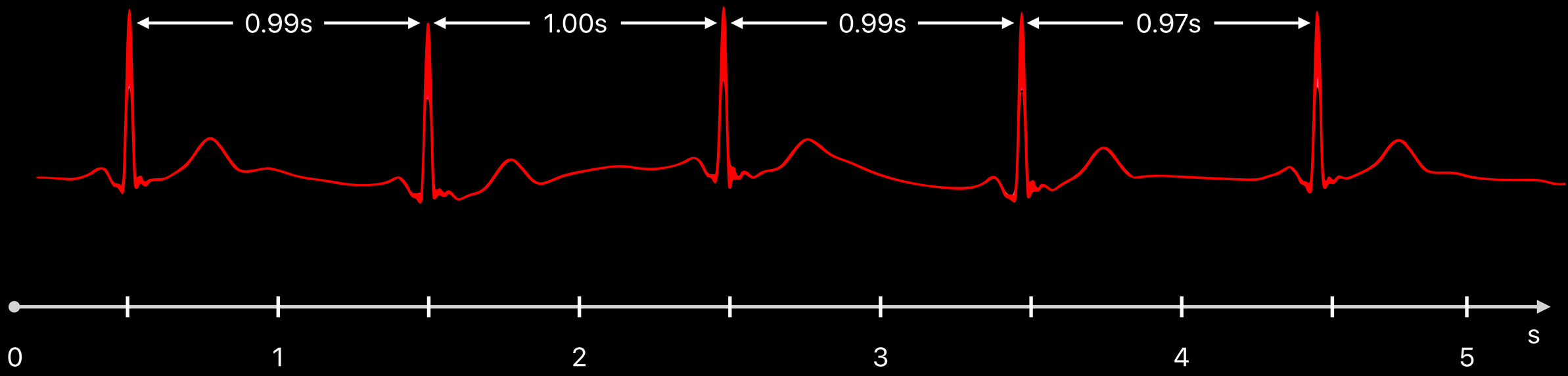

Heart Rate
Quantity Type
Quantity Sample
68 BPM
Quantity
Start: May 29, 2019 10:00:05
End: May 29, 2019 10:00:10


0
1
2
3
4
5
s

Heart Rate Variability SDNN

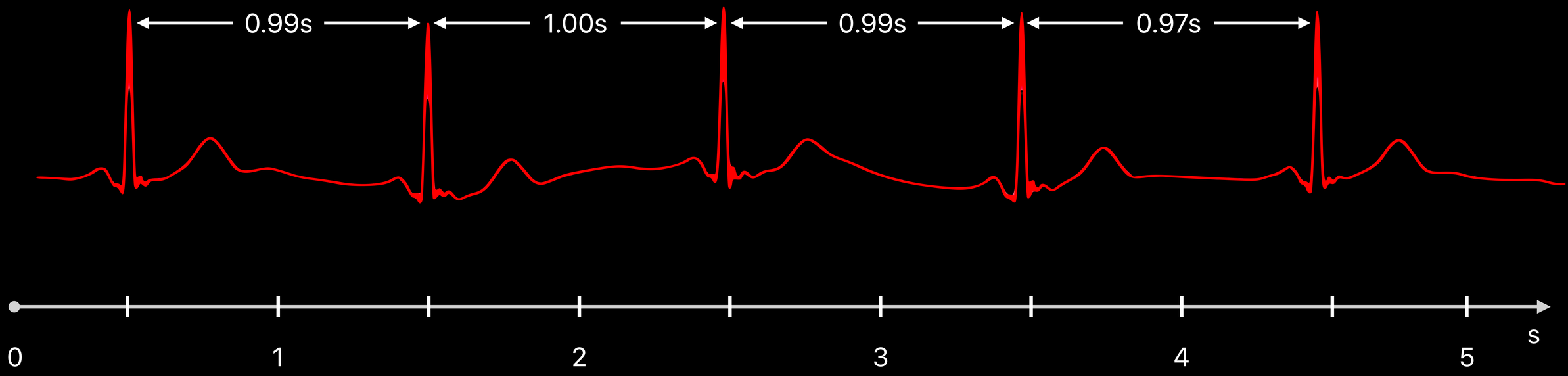


Heart Rate Variability SDNN

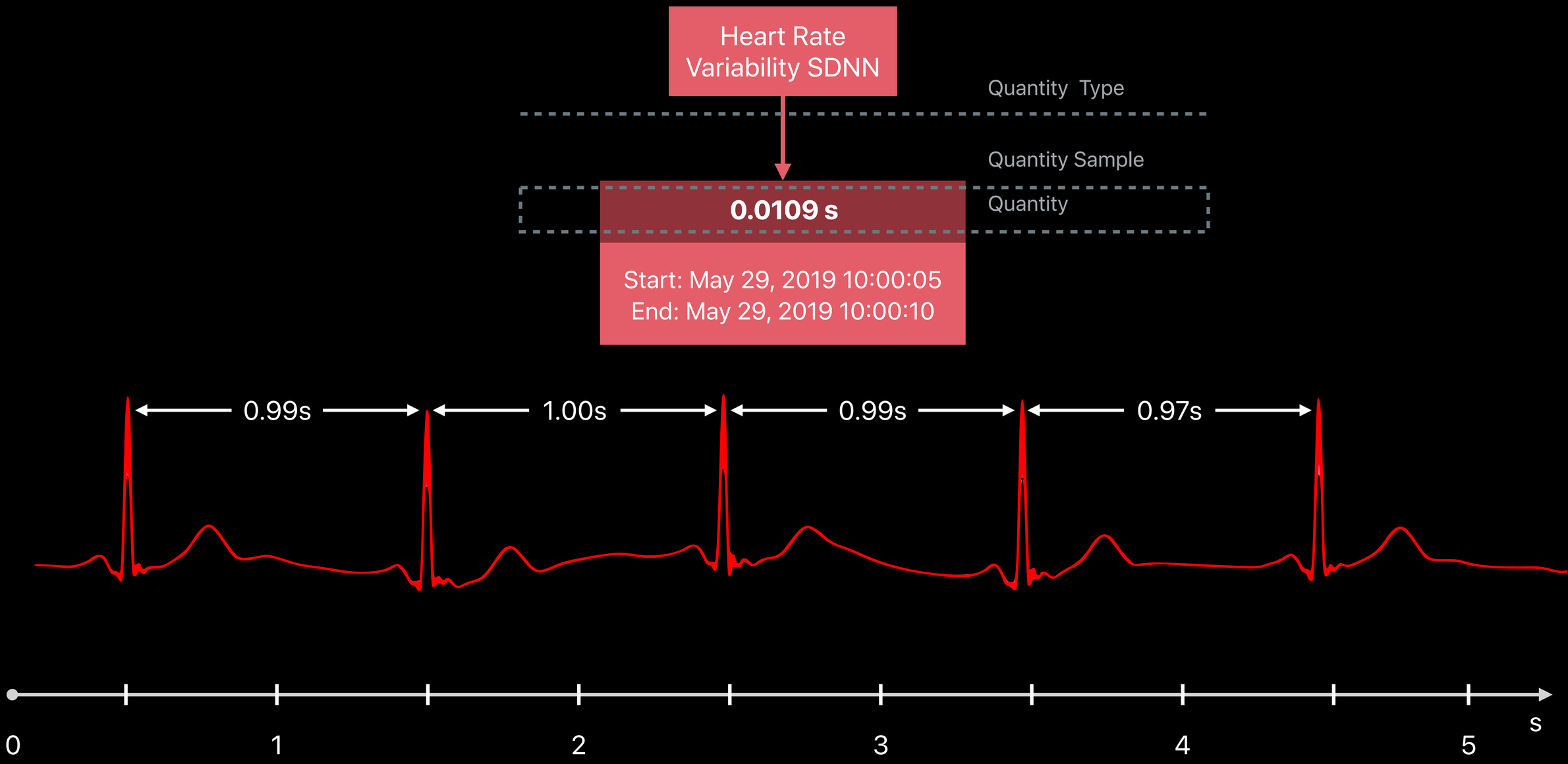


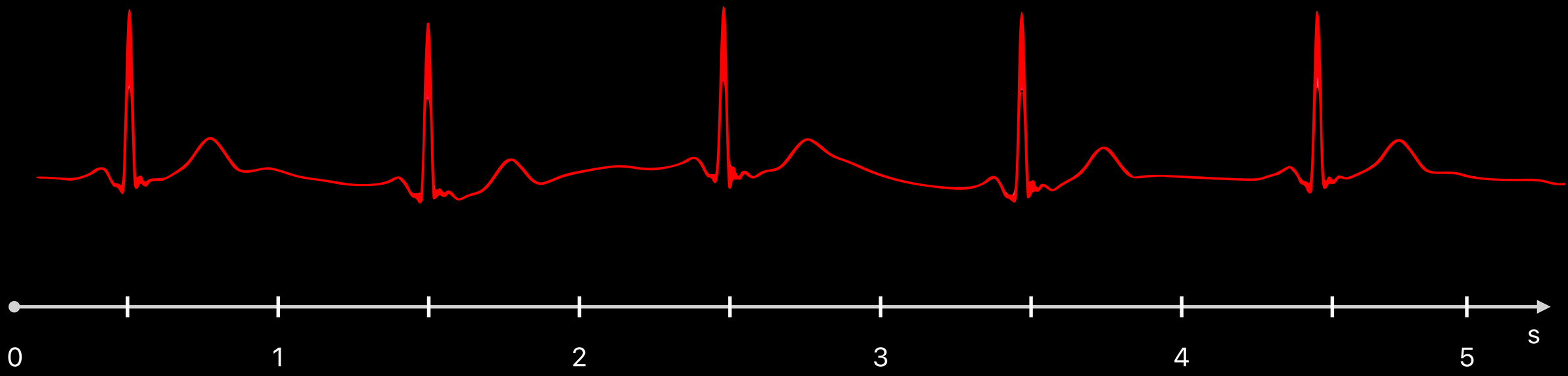
Heart Rate Variability SDNN

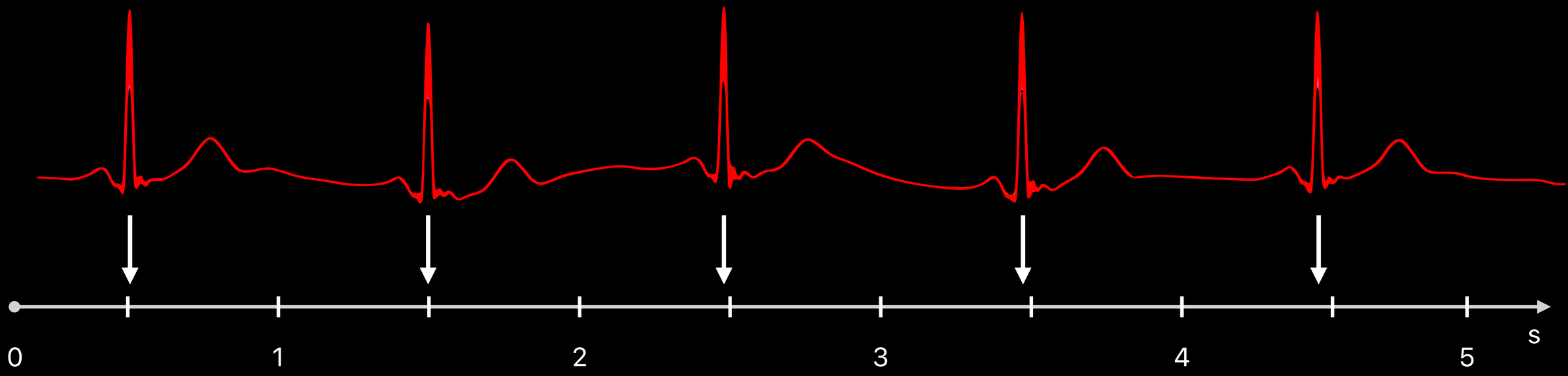
0.0109 s

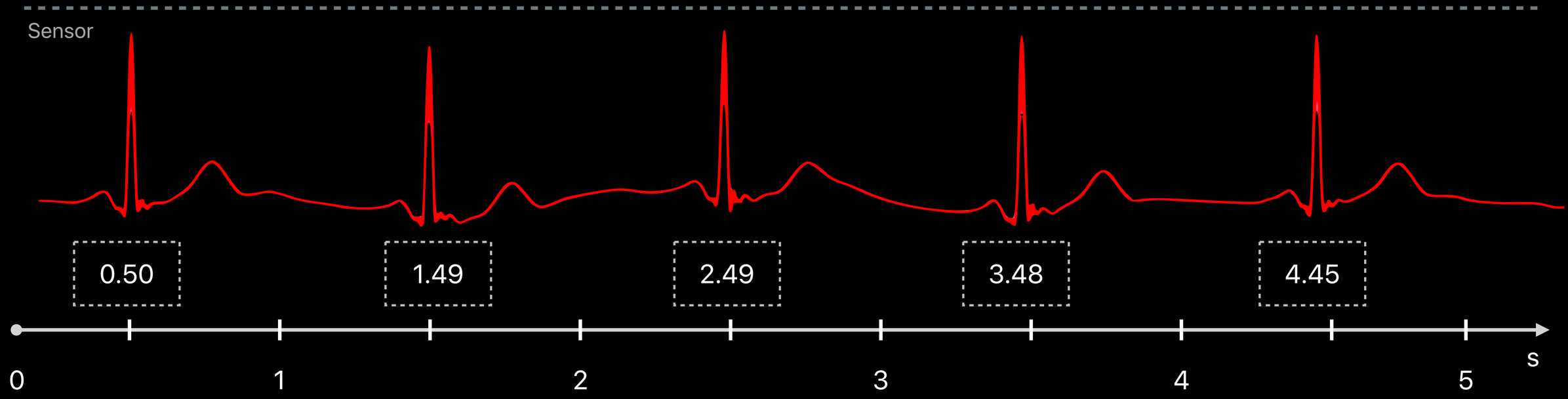


Heart Rate Variability SDNN




0
1
2
3
4
5
s


0
1
2
3
4
5
s


Sensor
0.50
1.49
2.49
3.48
4.45
0
1
2
3
4
5
s

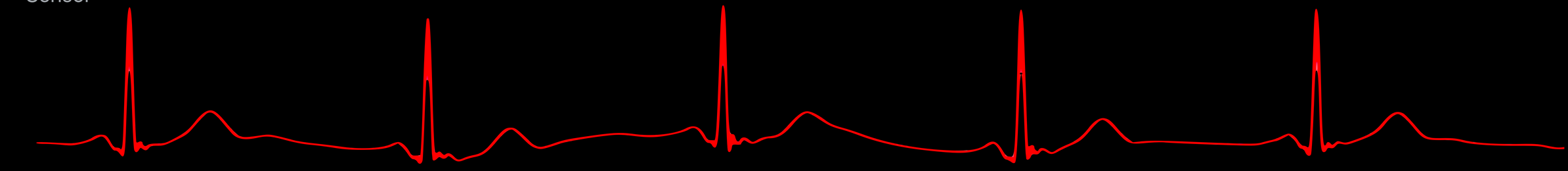
HKHeartbeatSeriesSample

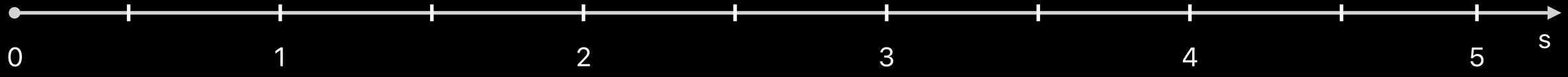
NEW

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

0.50	1.49	2.49	3.48	4.45

Sensor





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

NEW

```
// HKHeartbeatSeriesSample
class HKHeartbeatSeriesSample : HKSeriesSample
```

NEW

```
// HKHeartbeatSeriesSample
class HKHeartbeatSeriesSample : HKSeriesSample

// HKHeartbeatSeriesBuilder
class HKHeartbeatSeriesBuilder : HKSeriesBuilder
```

```swift
// 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
}
```

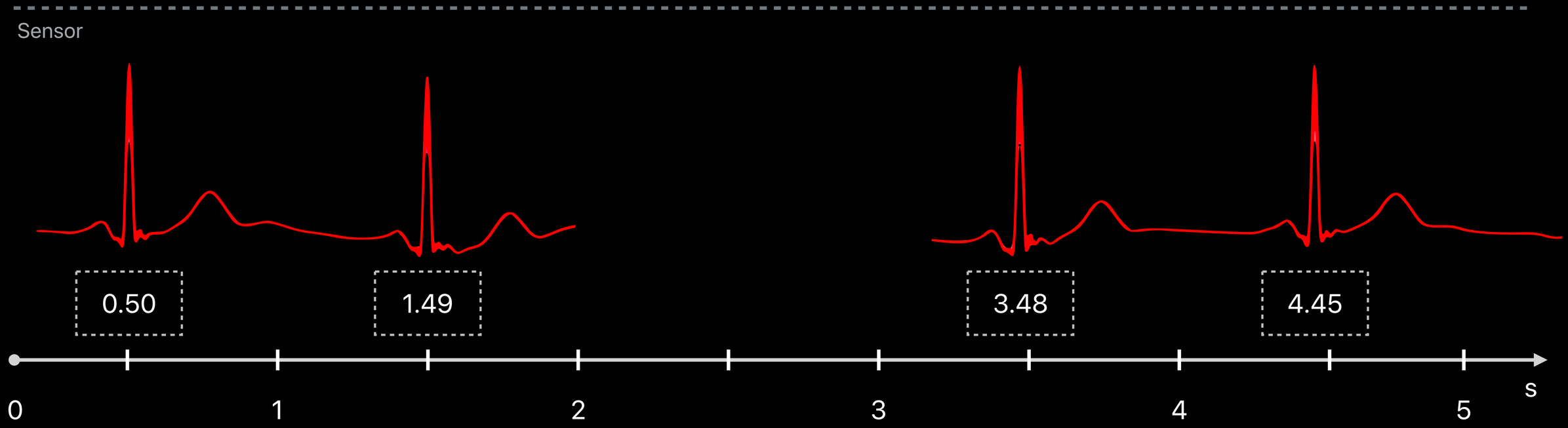
```swift
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// 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
}
```

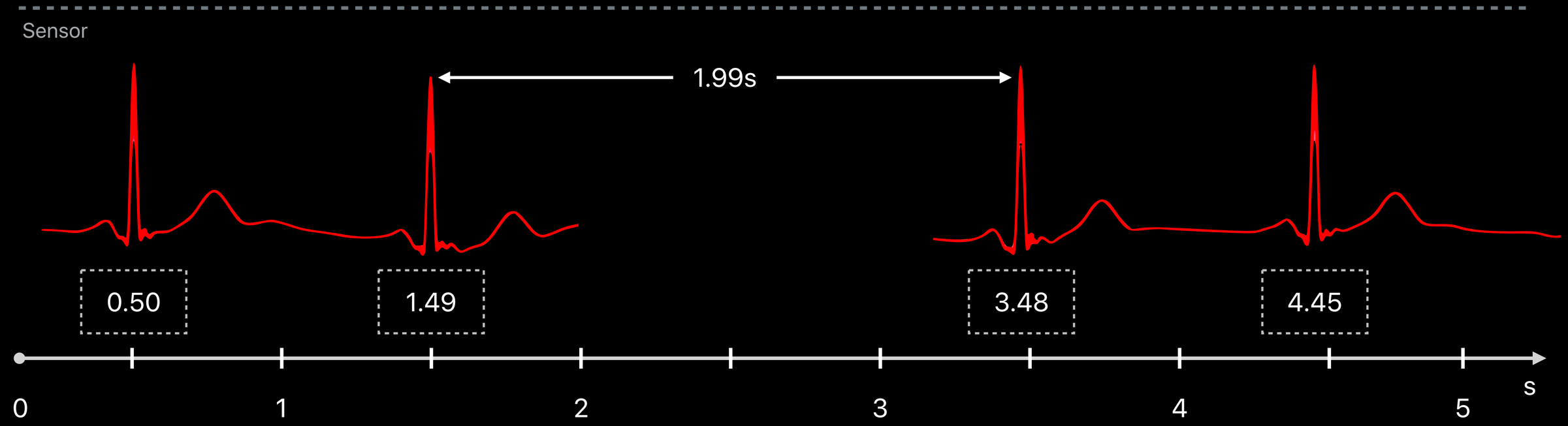
```swift
// Building HKHeartbeatSeriesSamples with HKHeartbeatSeriesBuilder

// 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
}
```

```
// 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
}
```


Sensor
0.50
1.49
3.48
4.45
0
1
2
3
4
5
s


Sensor
1.99s
0.50
1.49
3.48
4.45
0
1
2
3
4
5
s

```
// 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
}
```

NEW

```
// 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 = …
```

```swift
// 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
}
```

```swift
// 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

Heart Rate Events

Heart Rate Events

Heart Rate Events

HKCategoryTypeIdentifiers

NEW

`.lowHeartRateEvent`

`.highHeartRateEvent`

`.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

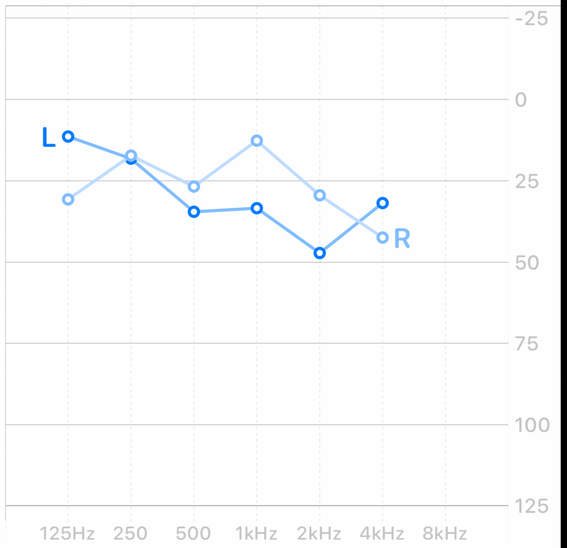
9:41

Hearing

Audiogram

Mild Impairment

May 19, 2019 10:30 AM



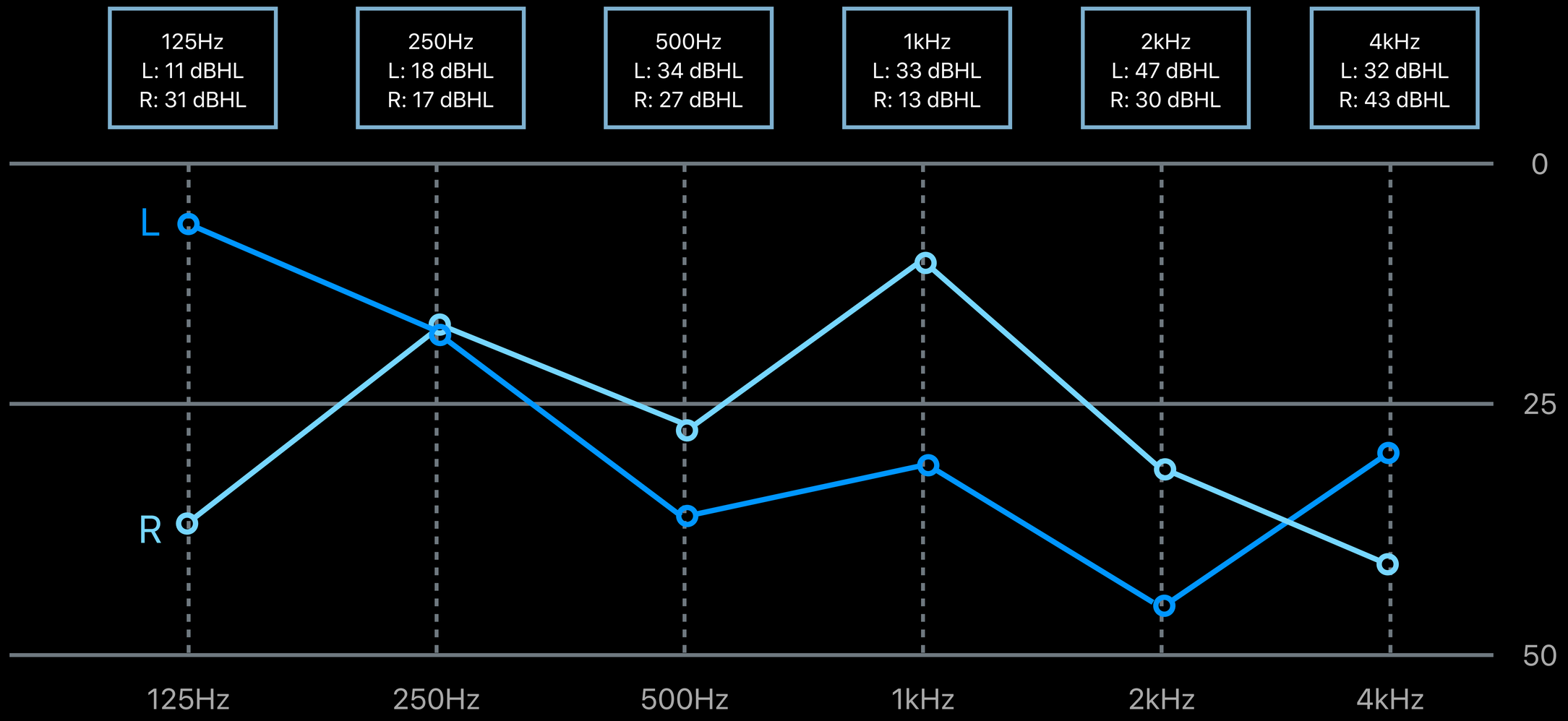
Left Average	37 dBHL
Right Average	28 dBHL

About Audiogram

An audiogram displays the results of a

Summary

Search

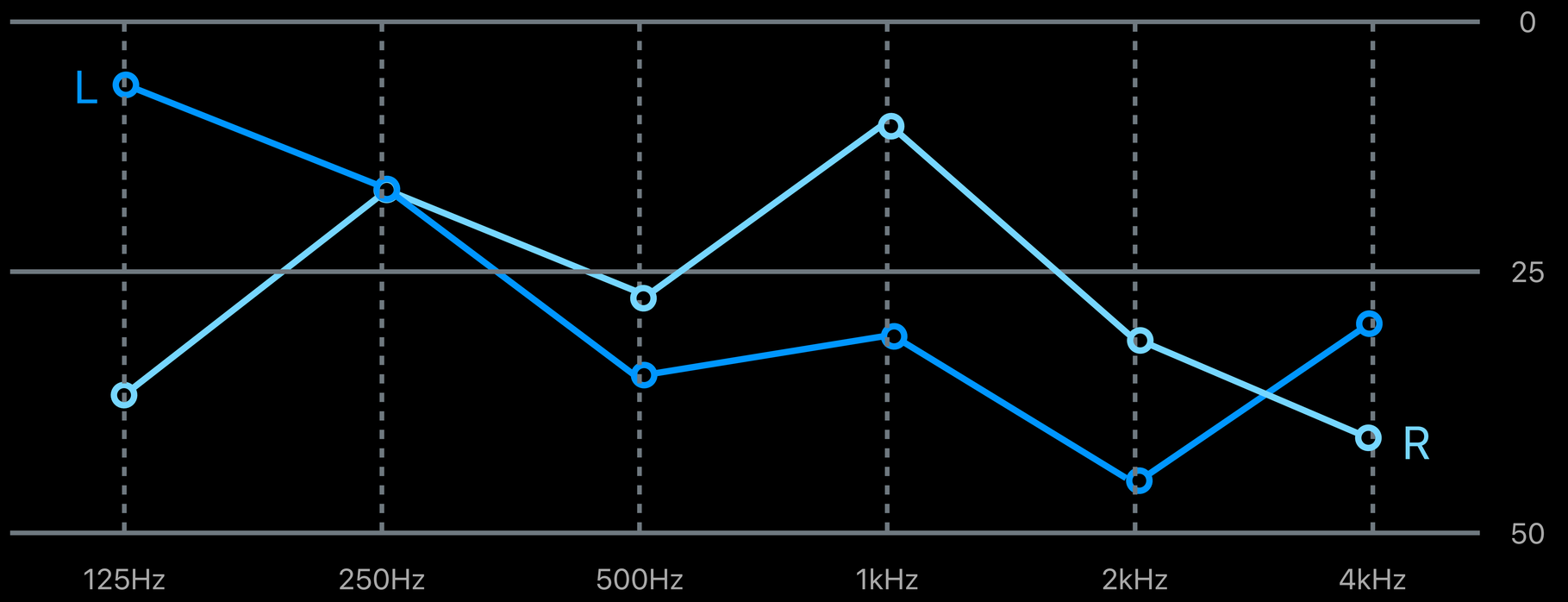

125Hz
L: 11 dBHL
R: 31 dBHL
250Hz
L: 18 dBHL
R: 17 dBHL
500Hz
L: 34 dBHL
R: 27 dBHL
1kHz
L: 33 dBHL
R: 13 dBHL
2kHz
L: 47 dBHL
R: 30 dBHL
4kHz
L: 32 dBHL
R: 43 dBHL
0
25
50
L
R
125Hz
250Hz
500Hz
1kHz
2kHz
4kHz

HKAudiogramSample

NEW

05/19/2019 10:30:00
UUID: 95-95b0-c2
device: myController
metadata: {"game1"}

125Hz	250Hz	500Hz	1kHz	2kHz	4kHz
L: 11 dBHL	L: 18 dBHL	L: 34 dBHL	L: 33 dBHL	L: 47 dBHL	L: 32 dBHL
R: 31 dBHL	R: 17 dBHL	R: 27 dBHL	R: 13 dBHL	R: 30 dBHL	R: 43 dBHL



```
// 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

NEW

`.headphoneAudioExposure`

Audio Exposure

HKQuantityTypeIdentifiers

NEW


```
.environmentalAudioExposure
```

NEW

`.audioExposureEvent`

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

Health and Fitness Technologies Lab

Wednesday, 3:00


WWDC19