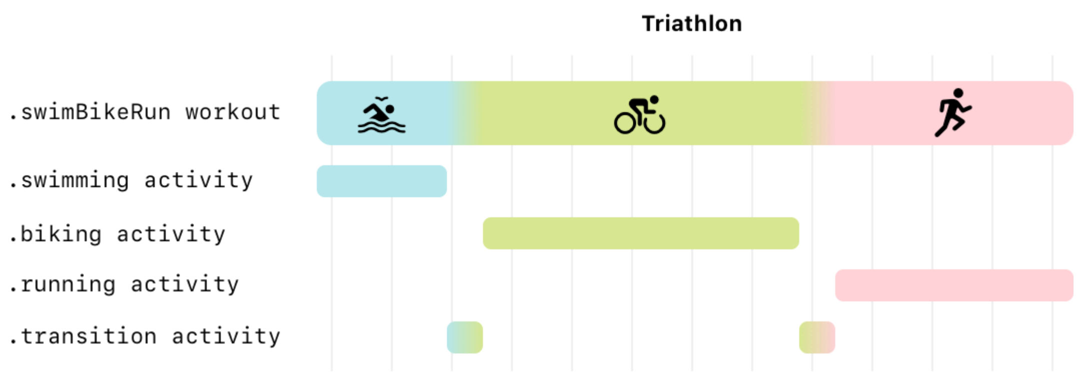
Article

Dividing a HealthKit workout into activities

Partition multisport and interval workouts into activities that represent the different parts of the workout.

Overview

Some workouts benefit from having the workout's duration broken into a set of discrete activities. For example, a multisport event, like a triathlon, has separate swim, bike, and run portions. Similarly, you can divide interval training into active and rest periods.



To model these activities in HealthKit, use the HKWorkoutActivity class to specify the different parts of a workout. You can then query for workouts that have matching activities, and analyze those activities independently from the rest of the workout.

HKWorkoutActivity instances have two main use cases:

Multisport workouts

For workouts with an HKWorkoutActivityType.swimBikeRun activity type, you can use HKWorkoutActivity instances to represent the HKWorkoutActivityType.swimming, HKWorkoutActivityType.cycling, and HKWorkoutActivityType.running portions of the workout. You can also use activities to mark the HKWorkoutActivityType.transition between these activities.

Interval training

Use HKWorkoutActivity instances to represent the active portions of the workout. You can create an interval workout for any HKWorkoutActivityType; however, all workout activities must use the same activity type as the containing workout. You can't mix activities within a single workout.

> Note
>
> All HKWorkout objects have at least one associated HKWorkoutActivity. If you don't explicitly add an activity to a workout, HealthKit adds an activity that matches the workout's activity type. This means you can use predicateForWorkoutActivities(workoutActivityType:) to filter for all the workouts that track an activity type. For example, filtering for workout activities that use HKWorkoutActivityType.running includes regular running workouts, running interval training, and HKWorkoutActivityType.swimBikeRun workouts that include a running activity.

Mark transitions

Activities can't overlap, but they also don't need to cover the entire HKWorkout sample's duration. You can leave the time between the active portions of the workout blank, or you can explicitly create workout activities for these intervals. For example, explicitly creating activities for the time between active portions of the workout helps you track statistics during those intervals.

For multisport workouts, you can explicitly mark the transitions between activities using an HKWorkoutActivityType.transition type. For interval training, you can use the same activity type as the workout for the rest periods; however, you may want to add custom metadata to indicate whether the activity is an active or resting interval.

How you choose to model the data depends on the needs of your app and the types of intervals you're tracking. If you're tracking a workout that alternates between a fast and a slow pace, using workout activities to explicitly track all intervals makes sense. If, however, the intervals alternate between exercising and resting (for example, during many high-intensity interval training exercises), you may want to leave the rest intervals blank.

Add workout activities to a workout session

To add workout activities to a workout session, start by creating the session and start collecting data using the workout builder.

```swift
// Create the workout configuration for a multisport workout.
let configuration = HKWorkoutConfiguration()
configuration.activityType = .swimBikeRun
configuration.locationType = .outdoor
```

```swift
// Create the workout session.
session = try HKWorkoutSession(healthStore: store,
                               configuration: configuration)

// Start the session and the workout builder.
let startDate = Date()
session.startActivity(with: startDate)
workoutBuilder = session.associatedWorkoutBuilder()

// Set the workout builder's data source.
workoutBuilder.dataSource =
HKLiveWorkoutDataSource(healthStore: store,
                        workoutConfiguration: configuration)

// Start collecting data.
try await workoutBuilder.beginCollection(at: startDate)
```

Next, when an activity begins, create a new configuration for the activity, and start the activity using the session's `beginNewActivity(configuration:date:metadata:)` method. The data source automatically begins collecting the default data types for the activity.

```swift
// Start the swimming activity.
let swimmingConfiguration = HKWorkoutConfiguration()
swimmingConfiguration.activityType = .swimming
swimmingConfiguration.locationType = .outdoor
swimmingConfiguration.swimmingLocationType = .openWater

session.beginNewActivity(configuration: swimmingConfiguration,
                         date: Date(),
                         metadata: nil)
```

When the activity ends, call the session's `endCurrentActivity(on:)` method. HealthKit also ends the current activity when you begin a new activity.

```swift
// End the activity.
session.endCurrentActivity(on: Date())
```

To explicitly track the intervals between activities, start a new activity using `HKWorkoutActivityType.transition`. End this transition when the next activity begins.

```
// Explicitly track the transition between activities.
let transitionConfiguration = HKWorkoutConfiguration()
transitionConfiguration.activityType = .transition
transitionConfiguration.locationType = .outdoor

session.beginNewActivity(configuration: transitionConfiguration,
                         date: Date(),
                         metadata: nil)
```

Finally, when the entire workout session ends, call the session's `end()` method. This also ends the current activity. Then call the workout builder's `finishWorkout(completion:)` method to save the workout to the HealthKit store. This method also returns an `HKWorkout` object, which you can use to display summary information about the workout.

```
// Ending the session also ends the current activity.
session.end()

// Finishing the workout saves the workout
// and returns an HKWorkout object that you can use to display summary data.
let workout = try await workoutBuilder.finishWorkout()

// Do something with the workout here.
print(workout as Any)
```

Enable and disable the collection of data

When you start a new workout activity, your data source automatically begins collecting relevant data from Apple Watch. To see the data types that the data source collects, check the data source's `typesToCollect` property.

When running an `HKWorkoutActivityType.swimBikeRun` workout session, HealthKit automatically changes the collected data types based on the current workout activity. For example, the data source collects data like `activeEnergyBurned`, `distanceSwimming`, and `swimmingStrokeCount`.

If you start an `HKWorkoutActivityType.running` activity, the system automatically updates the data source's `typesToCollect` property based on the new activity. For example, the data source automatically stops collecting `distanceSwimming` and `swimmingStrokeCount`, and starts collecting relevant data like `distanceWalkingRunning` and `runningStrideLength`.

Most of the time, you can use the default collected data types. However, if your app calculates and saves its own `HKSample` objects during the workout, you may want to manually enable and disable the collection of that data type, letting the data source automatically associate your samples with the workout.

To start collecting a data type, call the data source's `enableCollection(for:predicate:)` method.

```
// Enable the collection of respiratory rate.
guard let dataSource = session.associatedWorkoutBuilder().dataSource else {
    print("*** No data source found! ***")
    return }

let respiratoryRate = HKQuantityType(.respiratoryRate)
dataSource.enableCollection(for: respiratoryRate, predicate: nil)
```

HealthKit then associates any matching samples from your app with the workout activity. You can also disable the collection of a data type by calling `disableCollection(for:)`.

```
// Disable the collection of respiratory rate.
guard let dataSource = session.associatedWorkoutBuilder().dataSource else {
    print("*** No data source found! ***")
    return }

let respiratoryRate = HKQuantityType(.respiratoryRate)
dataSource.disableCollection(for: respiratoryRate)
```

Query for workout activities

To query for workouts with activities that match a specific predicate, start by creating a workout activity predicate using one of the `HKQuery` class's `predicateForWorkoutActivities` methods. Next, use `predicateForWorkouts(activityPredicate:)` to wrap the activity predicate inside a workout predicate. You can then use the workout predicate in your query.

```
// Create a predicate for an average heart rate of greater than 150 bpm.
let highHeartRate = HKQuantity(unit: .count(), doubleValue: 150.0)
let heartRateType = HKQuantityType(.heartRate)

let heartRatePredicate =
HKQuery.predicateForWorkoutActivities(operatorType: .greaterThan,
                                      quantityType: heartRateType,
                                      averageQuantity: highHeartRate)
```

```
// Wrap the workout activity predicate inside a workout predicate.
let workoutPredicate = HKQuery.predicateForWorkouts(activityPredicate: heartRatePred

let query = HKSampleQueryDescriptor(predicates: [.workout(workoutPredicate)],
                                    sortDescriptors: [])

let matchingWorkouts = try await query.result(for: store)

// Do something with the samples here.
print(matchingWorkouts)
```

This example returns all the workouts that have an activity with an average heart rate over 150 bpm. Use the workout's `workoutActivities` property to access the activities associated with a workout.

See Also

Samples

Adding samples to a workout

Create associated samples that add details to a workout.

Accessing condensed workout samples

Read series data from condensed workouts.

`class HKWorkout`

A workout sample that stores information about a single physical activity.

`class HKWorkoutActivity`

An object that describes an activity within a longer workout.

`class HKWorkoutBuilder`

A builder object that incrementally constructs a workout.

`class HKWorkoutType`

A type that identifies samples that store information about a workout.

`let HKWorkoutTypeIdentifier: String`

The workout type identifier.

`enum HKWorkoutActivityType`

The type of activity performed during a workout.

`enum HKWorkoutSessionType`

The type of session.

`class HKWorkoutEvent`

An object representing an important event during a workout.