Class

HKWorkout

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

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 13.0+ | macOS 13.0+ | visionOS 1.0+ | watchOS 2.0+

```
class HKWorkout
```

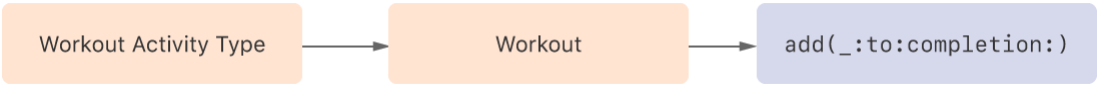
Mentioned in

- Dividing a HealthKit workout into activities
- About the HealthKit framework
- Creating a workout route

Overview

The `HKWorkout` class is a concrete subclass of the `HKSample` class; however, they behave somewhat differently than other sample types.

- You don't need a specific type identifier to create the `HKWorkoutType` instance. All workouts use the same type identifier.
- You must provide an `HKWorkoutActivityType` value for each workout. This value defines the type of activity performed during the workout.
- After saving the workout to the HealthKit store, you must associate additional samples with the workout (for example, active energy burned or distance samples). These samples provide fine-grained details. Use the `add(_:to:completion:)` method to associate them with the workout.



The workout records a summary of information about a single physical activity (for example, the duration, total distance, and total energy burned). It also acts as a container for other `HKSample` objects. You can associate any number of samples with a workout, adding details over the course of the workout. For example, you may want to break a single run into a number of shorter intervals, and then add samples to track the user's heart rate, energy burned, distance traveled, and steps taken for each interval. For more information, see Adding samples to a workout.

> **Note**
>
> If a workout has summary information, it also needs a set of associated samples that add up to the summary's total. See Adding samples to a workout.

HealthKit supports a wide range of activity types. For a complete list, see `HKWorkoutActivityType`.

Workouts are mostly immutable. You set their properties when you instantiate the workout, and they can't change. However, you can continue to add samples to the workouts.

Fill the Activity rings

Workouts can contribute to the Move and Exercise rings in the Activity app. To affect the rings, you must associate one or more active energy burned samples with the workout. Additionally:

- In watchOS. Use a workout session to track the user's activity. When the session has ended, create a workout object and the associated active energy burned samples. For more information, see `HKWorkoutSession`.

The system updates the Move ring based on the active energy burned samples. It updates the Exercise ring based on the amount of time the user spent actually exerting themselves during the workout session, as calculated by the watch's sensors.

- In iOS. No additional work is necessary. Workout objects automatically contribute to both the Move and Exercise rings. The Exercise ring increases by the workout's total duration, and the Move ring increases by the number of calories in the associated active energy burned samples. HealthKit also increases the Stand ring by one hour for each wall-clock hour that the workout overlaps.

Create and save workouts on the device that makes the most sense for your application—typically the device processing the user's workout.

Extend workouts

Like many HealthKit classes, the `HKWorkout` class should not be subclassed. You may extend workouts by adding metadata with custom keys as appropriate for your app.

For more information, see the methods `init(activityType:start:end:duration:totalEnergyBurned:totalDistance:metadata:)` and `init(activityType:start:end:workoutEvents:totalEnergyBurned:totalDistance:metadata:)`.

Topics

Creating workouts

~~`convenience init(activityType: HKWorkoutActivityType, start: Date, end: Date)`~~

Instantiates a new workout.

Deprecated

~~`convenience init(activityType: HKWorkoutActivityType, start: Date, end: Date, duration: TimeInterval, totalEnergyBurned: HKQuantity?, totalDistance: HKQuantity?, metadata: [String : Any]?)`~~

Instantiates a new workout that includes the energy burned, distance, and metadata for the workout.

Deprecated

~~`convenience init(activityType: HKWorkoutActivityType, start: Date, end: Date, workoutEvents: [HKWorkoutEvent]?, totalEnergyBurned: HKQuantity?, totalDistance: HKQuantity?, metadata: [String : Any]?)`~~

Instantiates a new workout whose duration is calculated based on the start and end dates and the provided workout events.

Deprecated

~~`convenience init(activityType: HKWorkoutActivityType, start: Date, end: Date, duration: TimeInterval, totalEnergyBurned: HKQuantity?, totalDistance: HKQuantity?, device: HKDevice?, metadata: [String : Any]?)`~~

Instantiates a new workout activity that includes the device that produced the sample data.

Deprecated

~~convenience init(activityType: HKWorkoutActivityType, start: Date, end: Date, workoutEvents: [HKWorkoutEvent]?, totalEnergyBurned: HKQuantity?, totalDistance: HKQuantity?, device: HKDevice?, metadata: [String : Any]?)~~

Instantiates a workout that includes both workout events and the device that produced the sample data.

Deprecated

~~convenience init(activityType: HKWorkoutActivityType, start: Date, end: Date, workoutEvents: [HKWorkoutEvent]?, totalEnergyBurned: HKQuantity?, totalDistance: HKQuantity?, totalFlightsClimbed: HKQuantity?, device: HKDevice?, metadata: [String : Any]?)~~

Instantiates a workout using a variety of data, including the number of flights of stairs climbed.

Deprecated

~~convenience init(activityType: HKWorkoutActivityType, start: Date, end: Date, workoutEvents: [HKWorkoutEvent]?, totalEnergyBurned: HKQuantity?, totalDistance: HKQuantity?, totalSwimmingStrokeCount: HKQuantity?, device: HKDevice?, metadata: [String : Any]?)~~

Instantiates a workout using a variety of data, including the number of strokes while swimming.

Deprecated

Accessing workout data

`var duration: TimeInterval`

The workout's duration.

`var workoutActivityType: HKWorkoutActivityType`

The type of activity performed during the workout.

`var workoutActivities: [HKWorkoutActivity]`

`var workoutEvents: [HKWorkoutEvent]?`

An array of workout event objects.

`func statistics(for: HKQuantityType) -> HKStatistics?`

Returns the workout's statistics for the provided quantity type.

`var allStatistics: [HKQuantityType : HKStatistics]`

A dictionary that contains all the statistics for the workout.

~~var totalDistance: HKQuantity?~~

The total distance traveled during the workout.

Deprecated

~~var totalEnergyBurned: HKQuantity?~~

The total active energy burned during the workout.

Deprecated

~~var totalFlightsClimbed: HKQuantity?~~

The total number of flights of stairs climbed during the workout.

Deprecated

~~var totalSwimmingStrokeCount: HKQuantity?~~

The total stroke count for the workout.

Deprecated

Specifying sort identifiers

`let HKWorkoutSortIdentifierDuration: String`

A constant for sorting workouts based on their duration.

`let HKWorkoutSortIdentifierTotalDistance: String`

A constant for sorting workouts based on their total distance.

`let HKWorkoutSortIdentifierTotalEnergyBurned: String`

A constant for sorting workouts based on the total energy burned.

Specifying predicate key paths

`let HKPredicateKeyPathWorkoutType: String`

The key path for accessing the workout's type.

`let HKPredicateKeyPathWorkoutDuration: String`

The key path for accessing the workout's duration.

~~let HKPredicateKeyPathWorkoutTotalDistance: String~~

The key path for accessing the workout's total distance.

Deprecated

~~let HKPredicateKeyPathWorkoutTotalEnergyBurned: String~~

The key path for accessing the workout's total energy burned.

Deprecated

`let HKPredicateKeyPathWorkoutAverageQuantity: String`

The key path for accessing workouts with a matching average quantity.

`let HKPredicateKeyPathWorkoutMaximumQuantity: String`

The key path for accessing workouts with a matching maximum quantity.

`let HKPredicateKeyPathWorkoutMinimumQuantity: String`

The key path for accessing workouts with a matching minimum quantity.

`let HKPredicateKeyPathWorkoutSumQuantity: String`

The key path for accessing workouts with a matching sum.

Specifying metadata keys

Workout Metadata Keys

Constants that can be used to add metadata to workouts.

Instance Properties

`var workoutPlan: WorkoutPlan?`

Relationships

Inherits From

`HKSample`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSCoding`

NSObjectProtocol
NSSecureCoding
Sendable
SendableMetatype

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.
- Dividing a HealthKit workout into activities
  Partition multisport and interval workouts into activities that represent the different parts of the workout.

`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.