

[HealthKit](#) / HKWorkoutBuilder

Class

HKWorkoutBuilder

A builder object that incrementally constructs a workout.

iOS 12.0+ | iPadOS 12.0+ | Mac Catalyst 13.1+ | macOS | visionOS 1.0+ | watchOS 5.0+

```
class HKWorkoutBuilder
```

Overview

Incrementally collect samples and events associated with a workout. When the workout ends, call `finishWorkout(completion:)` to create an [HKWorkout](#) sample and save it to the HealthKit store.

For watchOS, use an [HKWorkoutSession](#) and an [HKLiveWorkoutBuilder](#) instead.

Topics

Creating the builder

```
init(healthStore: HKHealthStore, configuration: HKWorkoutConfiguration, device: HKDevice?)
```

Returns a new workout builder object that is not connected to a workout session or other data source.

```
var device: HKDevice?
```

The device associated with the workout.

```
var workoutConfiguration: HKWorkoutConfiguration
```

The configuration information for the workout.

Starting the workout

```
func beginCollection(withStart: Date, completion: (Bool, (any Error)?)  
-> Void)
```

Sets the workout's start date and begins building the workout.

```
var startDate: Date?
```

The workout's start date and time.

```
func elapsedTime(at: Date) -> TimeInterval
```

Calculates the duration of the workout at the specified time.

Associating samples with the workout

```
func add([HKSample], completion: (Bool, (any Error)?) -> Void)
```

Adds a sample to be associated with the workout.

```
func seriesBuilder(for: HKSeriesType) -> HKSeriesBuilder?
```

Returns the series builder for the specified type, creating a new builder, if necessary.

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

Returns the statistics calculated for matching samples added to the workout.

Adding metadata to the workout

```
func addMetadata([String : Any], completion: (Bool, (any Error)?) ->  
Void)
```

Adds metadata to be saved with the workout.

```
var metadata: [String : Any]
```

The metadata the builder saves with the workout.

Adding events to the workout

```
func addWorkoutEvents([HKWorkoutEvent], completion: (Bool, (any Error)  
)?) -> Void)
```

Adds a workout event to the builder.

```
var workoutEvents: [HKWorkoutEvent]
```

The list of events added to the workout.

Managing workout activities

```
func addWorkoutActivity(HKWorkoutActivity, completion: (Bool, (any Error)?) -> Void)
```

Adds a workout activity to the workout builder.

```
func updateActivity(uuid: UUID, adding: [String : Any], completion: (Bool, (any Error)?) -> Void)
```

Adds metadata to a workout activity that you've already added to the workout builder.

```
func updateActivity(uuid: UUID, end: Date, completion: (Bool, (any Error)?) -> Void)
```

Sets the end date for a workout activity that you've already added to the workout builder.

```
var workoutActivities: [HKWorkoutActivity]
```

Ending the workout

```
func endCollection(withEnd: Date, completion: (Bool, (any Error)?) -> Void)
```

Stops the collection of data, sets the workout's end date, and deactivates the workout builder.

```
var endDate: Date?
```

The workout's end date and time.

```
func finishWorkout(completion: (HKWorkout?, (any Error)?) -> Void)
```

Creates the workout, using the samples and events added to the builder, and saves it to the HealthKit store.

```
func discardWorkout()
```

Stops the collection of data and discards the current results without saving the workout.

Accessing workout statistics

```
var allStatistics: [HKQuantityType : HKStatistics]
```

A dictionary that contains all the statistics for the workout builder.

Relationships

Inherits From

`NSObject`

Inherited By

`HKLiveWorkoutBuilder`

Conforms To

`CVarArg`

`CustomDebugStringConvertible`

`CustomStringConvertible`

`Equatable`

`Hashable`

`NSObjectProtocol`

`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 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 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.