

Framework

Core Motion

Process accelerometer, gyroscope, pedometer, and environment-related events.

iOS 4.0+ iPadOS 4.0+ Mac Catalyst 13.0+ macOS 10.15+ visionOS 1.0+ watchOS 2.0+

Overview

Core Motion reports motion- and environment-related data from the available onboard hardware of iOS, iPadOS, watchOS, and visionOS devices. This hardware includes the device's accelerometers and gyroscopes, and, when available, the pedometer, magnetometer, and barometer. Use this data in your app as input for user interactions, fitness tracking, health-related matters, and more. For example, a game might use accelerometer and gyroscope input to control onscreen game behavior.

The services of this framework provide access to motion data either as raw or processed values, and many services provide both types of values. Raw values reflect the unmodified data from the hardware, while processed values eliminate forms of bias that might adversely affect your usage of the data. For example, a processed accelerometer value reflects only the acceleration caused by the user and not the acceleration caused by gravity.

Not all services are available on all devices, and some services might be unavailable even on devices with the required hardware. For example, many Core Motion services are available to visionOS apps, but those services aren't available to compatible iPad and iPhone apps running in visionOS. Before you try to use any motion-related services, check the availability of those services using a `CMMotionManager` object.

Important

An iOS app must include usage description keys in its `Info.plist` file for the types of data it needs. If these keys aren't present, the app crashes when you try to access the corresponding service. To access motion and fitness data, include `NSMotionUsageDescription`. To access the fall-detection service, include `NSFallDetectionUsageDescription`.

Topics

Essentials



Core Motion updates

Learn about important changes to Core Motion.

`class CMMotionManager`

The object for starting and managing motion services.

Device motion

Access acceleration, attitude, rotation, and magnetic field data that is adjusted for gravity and other forms of bias.



Getting processed device-motion data

Retrieve motion data that the system processed to remove environmental bias, such as the effects of gravity.

`class CMDeviceMotion`

Encapsulated measurements of the attitude, rotation rate, and acceleration of a device.

`class CMAttitude`

The device's orientation relative to a known frame of reference at a point in time.

`struct CMAttitudeReferenceFrame`

Constants that indicate the frame of reference for attitude-related motion data.

`class CMHeadphoneMotionManager`

An object that starts and manages headphone motion services.

Accelerometers

Access accelerometer data for all three axes of the device.

 Getting raw accelerometer events

Retrieve data from the onboard accelerometers.

```
class CMAccelerometerData
```

A data sample from the device's three accelerometers.

```
class CMRecordedAccelerometerData
```

A single piece of accelerometer data that was recorded by the device.

```
class CMSensorRecorder
```

An object that gathers and retrieves accelerometer data from a device.

```
class CMSensorDataList
```

A list of the accelerometer data recorded by the system.

Gyroscopes

Access the raw gyroscope data.

 Getting raw gyroscope events

Retrieve data from the onboard gyroscopes.

```
class CMGyroData
```

A single measurement of the device's rotation rate.

Magnetometer

Access raw magnetometer data.

```
class CMMagnetometerData
```

Measurements of the Earth's magnetic field relative to the device.

Altitude data

Access altitude data based on barometric sensor information.

```
class CMAltimeter
```

An object that initiates the delivery of altitude-related changes.

```
class CMAbsoluteAltitudeData
```

Data that records a change in absolute altitude.

```
class CMAltitudeData
```

Data for a recorded change in altitude.

Ambient pressure

```
class CMRecordedPressureData
```

A recorded measurement of pressure data.

```
class CMAmbientPressureData
```

A measurement of the ambient pressure and temperature.

Water submersion

 Accessing submersion data

Use a water-submersion manager to receive water pressure, temperature, and depth data on Apple Watch Ultra.

```
class CMWaterSubmersionManager
```

An object for managing the collection of pressure and temperature data during submersion.

```
protocol CMWaterSubmersionManagerDelegate
```

A delegate that receives updates about ambient pressure, water pressure, water temperature, and submersion events.

```
class CMWaterSubmersionEvent
```

An event indicating that the device's submersion state has changed.

`class CMWaterSubmersionMeasurement`

An update that contains data about the pressure and depth.

`class CMWaterTemperature`

An update that contains data about the water temperature.

Activity

`class CMMotionActivityManager`

An object that manages access to the motion data stored by the device.

`class CMHeadphoneActivityManager`

An object that starts and manages headphone activity services.

`class CMMotionActivity`

The data for a single motion update event.

`{}` Getting motion-activity data from headphones

Configure your app to listen for motion-activity changes from headphones.

Pedometer and fitness

Access step-counting data from the built-in motion processor.

`class CMPedometer`

An object for fetching the system-generated live walking data.

`class CMPedometerData`

Information about the distance traveled by a user on foot.

`class CMPedometerEvent`

A change in the user's pedestrian activity.

~~`class CMStepCounter`~~

The number of steps the user has taken with the device.

~~Deprecated~~

`class CMOdometerData`

A class that represents odometer data for workouts.

```
class CMHighFrequencyHeartRateData
```

A class that represents heart rate data collected at 1 Hz.

Movement disorder

- Getting movement disorder symptom data
Retrieve data from the Apple Watch's movement disorder manager.
- Adhering to the movement disorder data collection requirements
Ensure that your users understand and have control over the data your app collects.
- Movement disorder algorithm changelog
A chronological log of notable changes to the movement disorder algorithm.

```
class CMMovementDisorderManager
```

A manager for recording and querying movement disorder data.

```
class CMTremorResult
```

A result object that contains data about the presence and strength of tremors during a one-minute interval.

```
class CMDyskineticSymptomResult
```

A result object that contains data about the likely presence of dyskinetic symptoms during a one-minute interval.

Fall detection

```
class CMFallDetectionManager
```

An object for managing fall detection events.

```
protocol CMFallDetectionDelegate
```

A delegate that receives information about fall detection events and authorization status changes.

```
class CMFallDetectionEvent
```

An object that contains data about a fall detection event.

NSFallDetectionUsageDescription

A message to the user that explains the app's request for permission to access fall detection event data.

Historical data

Access recorded motion events to help you analyze movement patterns.

```
class CMBatchedSensorManager
```

Common data

```
class CMLogItem
```

The base class for all motion-related data objects.