Core Location Framework Reference

Data Management: Device Information

2010-05-11

Contents

Core Location Framework Reference

Framework	/System/Library/Frameworks/CoreLocation.framework
Header file directories	/System/Library/Frameworks/CoreLocation.framework/Headers
Companion guide	Location Awareness Programming Guide
Declared in	CLError.h CLErrorDomain.h CLHeading.h CLLocation.h CLLocationManager.h CLLocationManagerDelegate.h CLRegion.h

The Core Location framework lets you determine the current location or heading associated with a device. The framework uses the available hardware to determine the user's position and heading. You use the classes and protocols in this framework to configure and schedule the delivery of location and heading events. You can also use it to define geographic regions and monitor when the user crosses the boundaries of those regions.

Classes

PART I

Classes

CLHeading Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Availability	Available in iOS 3.0 and later.
Declared in	CLHeading.h

Overview

A `CLHeading` object contains heading data generated by a `CLLocationManager` object. The heading data consists of computed values for true and magnetic north. It also includes the raw data for the three-dimensional vector used to compute those values.

Typically, you do not create instances of this class yourself, nor do you subclass it. Instead, you receive instances of this class through the delegate assigned to the `CLLocationManager` object whose `startUpdatingHeading` (page 39) method you called.

Note: If you want heading objects to contain valid data for the `trueHeading` (page 11) property, your location manager object should also be configured to deliver location updates. You can start the delivery of these updates by calling the location manager object's `startUpdatingLocation` (page 40) method.

Tasks

Accessing the Heading Attributes

`magneticHeading` (page 10) *property*
The heading (measured in degrees) relative to magnetic north. (read-only)

`trueHeading` (page 11) *property*
The heading (measured in degrees) relative to true north. (read-only)

`headingAccuracy` (page 10) *property*
The maximum deviation (measured in degrees) between the reported heading and the true geomagnetic heading. (read-only)

`timestamp` (page 11) *property*
The time at which this heading was determined. (read-only)

`- description` (page 13)
    Returns the heading data in a formatted text string.

Accessing the Raw Heading Data

`x` (page 12) *property*
    The geomagnetic data (measured in microteslas) for the x-axis. (read-only)

`y` (page 12) *property*
    The geomagnetic data (measured in microteslas) for the y-axis. (read-only)

`z` (page 12) *property*
    The geomagnetic data (measured in microteslas) for the z-axis. (read-only)

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

headingAccuracy

The maximum deviation (measured in degrees) between the reported heading and the true geomagnetic heading. (read-only)

```
@property(readonly, nonatomic) CLLocationDirection headingAccuracy
```

Discussion
A positive value in this property represents the potential error between the value reported by the `magneticHeading` (page 10) property and the actual direction of magnetic north. Thus, the lower the value of this property, the more accurate the heading. A negative value means that the reported heading is invalid, which can occur when the device is uncalibrated or there is strong interference from local magnetic fields.

Availability
Available in iOS 3.0 and later.

Declared In
`CLHeading.h`

magneticHeading

The heading (measured in degrees) relative to magnetic north. (read-only)

```
@property(readonly, nonatomic) CLLocationDirection magneticHeading
```

Discussion
The value in this property represents the heading relative to the magnetic North Pole, which is different from the geographic North Pole. The value `0` means the device is pointed toward magnetic north, `90` means it is pointed east, `180` means it is pointed south, and so on. The value in this property should always be valid.

In iOS 3.x and earlier, the value in this property is always measured relative to the top of the device in a portrait orientation, regardless of the device's actual physical or interface orientation. In iOS 4.0 and later, the value is measured relative to the heading orientation specified by the location manager. For more information, see the `headingOrientation` (page 32) property in *CLLocationManager Class Reference*.

If the `headingAccuracy` property contains a negative value, the value in this property should be considered unreliable.

Availability
Available in iOS 3.0 and later.

See Also
- `@property headingAccuracy` (page 10)
- `@property trueHeading` (page 11)

Declared In
`CLHeading.h`

timestamp

The time at which this heading was determined. (read-only)

```
@property(readonly, nonatomic) NSDate *timestamp
```

Availability
Available in iOS 3.0 and later.

Declared In
`CLHeading.h`

trueHeading

The heading (measured in degrees) relative to true north. (read-only)

```
@property(readonly, nonatomic) CLLocationDirection trueHeading
```

Discussion
The value in this property represents the heading relative to the geographic North Pole. The value `0` means the device is pointed toward true north, `90` means it is pointed due east, `180` means it is pointed due south, and so on. A negative value indicates that the heading could not be determined.

In iOS 3.x and earlier, the value in this property is always measured relative to the top of the device in a portrait orientation, regardless of the device's actual physical or interface orientation. In iOS 4.0 and later, the value is measured relative to the heading orientation specified by the location manager. For more information, see the `headingOrientation` (page 32) property in *CLLocationManager Class Reference*.

Important: This property contains a valid value only if location updates are also enabled for the corresponding location manager object. Because the position of true north is different from the position of magnetic north on the Earth's surface, Core Location needs the current location of the device to compute the value of this property.

Availability
Available in iOS 3.0 and later.

See Also
`@property magneticHeading` (page 10)

Declared In
`CLHeading.h`

x

The geomagnetic data (measured in microteslas) for the x-axis. (read-only)

```
@property(readonly, nonatomic) CLHeadingComponentValue x
```

Discussion
The value reported by this property is normalized to the range -128 to +128. This value represents the x-axis deviation from the magnetic field lines being tracked by the device.

Availability
Available in iOS 3.0 and later.

Declared In
`CLHeading.h`

y

The geomagnetic data (measured in microteslas) for the y-axis. (read-only)

```
@property(readonly, nonatomic) CLHeadingComponentValue y
```

Discussion
The value reported by this property is normalized to the range -128 to +128. This value represents the y-axis deviation from the magnetic field lines being tracked by the device.

Availability
Available in iOS 3.0 and later.

Declared In
`CLHeading.h`

z

The geomagnetic data (measured in microteslas) for the z-axis. (read-only)

```
@property(readonly, nonatomic) CLHeadingComponentValue z
```

Discussion
The value reported by this property is normalized to the range -128 to +128. This value represents the z-axis deviation from the magnetic field lines being tracked by the device.

Availability
Available in iOS 3.0 and later.

Declared In
`CLHeading.h`

Instance Methods

description

Returns the heading data in a formatted text string.

```
- (NSString *)description
```

Return Value
A string of the form “magneticHeading <*magnetic*> trueHeading <*heading*> accuracy <*accuracy*> x <*x*> y <*y*> z <*z*> @ <*date-time*>” where <*magnetic*>, <*heading*>, <*accuracy*>, <*x*>, <*y*>, and <*z*> are formatted floating-point numbers and <*date-time*> is a formatted date string that includes date, time, and time zone information.

Availability
Available in iOS 3.0 and later.

Declared In
`CLHeading.h`

Constants

CLHeadingComponentValue

A type used to report magnetic differences reported by the onboard hardware.

```
typedef double CLHeadingComponentValue;
```

Availability
Available in iOS 3.0 and later.

Declared In
`CLHeading.h`

CLLocation Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/CoreLocation.framework
Availability	Available in iOS 2.0 and later.
Declared in	CLLocation.h

Overview

A `CLLocation` object represents the location data generated by a `CLLocationManager` object. This object incorporates the geographical coordinates and altitude of the device's location along with values indicating the accuracy of the measurements and when those measurements were made. In iOS, this class also reports information about the speed and heading in which the device is moving.

Typically, you use a `CLLocationManager` object to create instances of this class based on the last known location of the user's device. You can create instances yourself, however, if you want to cache custom location data or get the distance between two points.

This class is designed to be used as is and should not be subclassed.

Tasks

Initializing a Location Object

`– initWithLatitude:longitude:` (page 21)
Initializes and returns a location object with the specified latitude and longitude.

`– initWithCoordinate:altitude:horizontalAccuracy:verticalAccuracy:timestamp:` (page 20)
Initializes and returns a location object with the specified coordinate information.

Location Attributes

`coordinate` (page 17) *property*
  The geographical coordinate information. (read-only)

`altitude` (page 16) *property*
  The altitude measured in meters. (read-only)

`horizontalAccuracy` (page 18) *property*
  The radius of uncertainty for the location, measured in meters. (read-only)

`verticalAccuracy` (page 19) *property*
  The accuracy of the altitude value in meters. (read-only)

`timestamp` (page 18) *property*
  The time at which this location was determined. (read-only)

`– description` (page 19)
  Returns the location data in a formatted text string.

Measuring the Distance Between Coordinates

`– distanceFromLocation:` (page 19)
  Returns the distance (in meters) from the receiver's location to the specified location.

`– getDistanceFrom:` (page 20) Deprecated in iOS 3.2
  Returns the distance (in meters) from the receiver's location to the specified location. (Deprecated. Use the `distanceFromLocation:` (page 19) method instead.)

Getting Speed and Course Information

`speed` (page 18) *property*
  The instantaneous speed of the device in meters per second.

`course` (page 17) *property*
  The direction in which the device is traveling.

Properties

For more about Objective-C properties, see "Properties" in *The Objective-C Programming Language*.

altitude

The altitude measured in meters. (read-only)

```
@property(readonly, NS_NONATOMIC_IPHONEONLY) CLLocationDistance altitude
```

Discussion
Positive values indicate altitudes above sea level. Negative values indicate altitudes below sea level.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.0 and later.

See Also
`@property verticalAccuracy` (page 19)

Declared In
`CLLocation.h`

coordinate

The geographical coordinate information. (read-only)

```
@property(readonly, NS_NONATOMIC_IPHONEONLY) CLLocationCoordinate2D coordinate
```

Discussion
When running in the simulator, Core Location assigns a fixed set of coordinate values to this property. You must run your application on an iOS-based device to get real location values.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

course

The direction in which the device is traveling.

```
@property(readonly, NS_NONATOMIC_IPHONEONLY) CLLocationDirection course
```

Discussion
Course values are measured in degrees starting at due north and continuing clockwise around the compass. Thus, north is 0 degrees, east is 90 degrees, south is 180 degrees, and so on. Course values may not be available on all devices. A negative value indicates that the direction is invalid.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.2 and later.

Declared In
`CLLocation.h`

horizontalAccuracy

The radius of uncertainty for the location, measured in meters. (read-only)

```
@property(readonly, NS_NONATOMIC_IPHONEONLY) CLLocationAccuracy horizontalAccuracy
```

Discussion
The location's latitude and longitude identify the center of the circle, and this value indicates the radius of that circle. A negative value indicates that the location's latitude and longitude are invalid.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

speed

The instantaneous speed of the device in meters per second.

```
@property(readonly, NS_NONATOMIC_IPHONEONLY) CLLocationSpeed speed
```

Discussion
This value reflects the instantaneous speed of the device in the direction of its current heading. A negative value indicates an invalid speed. Because the actual speed can change many times between the delivery of subsequent location events, you should use this property for informational purposes only.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.2 and later.

Declared In
`CLLocation.h`

timestamp

The time at which this location was determined. (read-only)

```
@property(readonly, NS_NONATOMIC_IPHONEONLY) NSDate *timestamp
```

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

verticalAccuracy

The accuracy of the altitude value in meters. (read-only)

```
@property(readonly, NS_NONATOMIC_IPHONEONLY) CLLocationAccuracy verticalAccuracy
```

Discussion
The value in the `altitude` property could be plus or minus the value indicated by this property. A negative value indicates that the altitude value is invalid.

Determining the vertical accuracy requires a device with GPS capabilities. Thus, on some earlier iOS-based devices, this property always contains a negative value.

Special Considerations
In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.0 and later.

See Also
`@property altitude` (page 16)

Declared In
`CLLocation.h`

Instance Methods

description

Returns the location data in a formatted text string.

```
- (NSString *)description
```

Return Value
A string of the form "<<*latitude*>, <*longitude*>> +/- <*accuracy*>m (speed <*speed*> kph / heading <*heading*>) @ <*date-time*>", where <*latitude*>, <*longitude*>, <*accuracy*>, <*speed*>, and <*heading*> are formatted floating point numbers and <*date-time*> is a formatted date string that includes date, time, and time zone information.

Discussion
The returned string is intended for display purposes only.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

distanceFromLocation:

Returns the distance (in meters) from the receiver's location to the specified location.

```
- (CLLocationDistance)distanceFromLocation:(const CLLocation *)location
```

Parameters

`location`

The other location.

Return Value
The distance (in meters) between the two locations.

Discussion
This method measures the distance between the two locations by tracing a line between them that follows the curvature of the Earth. The resulting arc is a smooth curve and does not take into account specific altitude changes between the two locations.

Availability
Available in iOS 3.2 and later.

Declared In
`CLLocation.h`

getDistanceFrom:

Returns the distance (in meters) from the receiver's location to the specified location. (Deprecated in iOS 3.2. Use the `distanceFromLocation:` (page 19) method instead.)

```
- (CLLocationDistance)getDistanceFrom:(const CLLocation *)location
```

Parameters

`location`

The other location.

Return Value
The distance (in meters) between the two locations.

Discussion
This method measures the distance between the two locations by tracing a line between them that follows the curvature of the Earth. The resulting arc is a smooth curve and does not take into account specific altitude changes between the two locations.

Availability
Available in iOS 2.0 and later.

Deprecated in iOS 3.2.

Declared In
`CLLocation.h`

initWithCoordinate:altitude:horizontalAccuracy:verticalAccuracy:timestamp:

Initializes and returns a location object with the specified coordinate information.

```
- (id)initWithCoordinate:(CLLocationCoordinate2D)coordinate
    altitude:(CLLocationDistance)altitude
    horizontalAccuracy:(CLLocationAccuracy)hAccuracy
    verticalAccuracy:(CLLocationAccuracy)vAccuracy timestamp:(NSDate *)timestamp
```

Parameters

`coordinate`
: A coordinate structure containing the latitude and longitude values.

`altitude`
: The altitude value for the location.

`hAccuracy`
: The accuracy of the coordinate value. Specifying a negative number indicates that the coordinate value is invalid.

`vAccuracy`
: The accuracy of the altitude value. Specifying a negative number indicates that the altitude value is invalid.

`timestamp`
: The time to associate with the location object. Typically, you would set this to the current time.

Return Value
A location object initialized with the specified information.

Discussion
Typically, you acquire location objects from the location service, but you can use this method to create new location objects for other uses in your application.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

initWithLatitude:longitude:

Initializes and returns a location object with the specified latitude and longitude.

```
- (id)initWithLatitude:(CLLocationDegrees)latitude
    longitude:(CLLocationDegrees)longitude
```

Parameters

`latitude`
: The latitude of the coordinate point.

`longitude`
: The longitude of the coordinate point.

Return Value
A location object initialized with the specified coordinate point.

Discussion
Typically, you acquire location objects from the location service, but you can use this method to create new location objects for other uses in your application. When using this method, the other properties of the object are initialized to appropriate values. In particular, the `altitude` and `horizontalAccuracy` properties are set to 0, the `verticalAccuracy` property is set to -1 to indicate that the altitude value is invalid, and the `timestamp` property is set to the time at which the instance was initialized.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

Constants

CLLocationDegrees

Represents a latitude or longitude value specified in degrees.

```
typedef double CLLocationDegrees;
```

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

CLLocationCoordinate2D

A structure that contains a geographical coordinate using the WGS 84 reference frame.

```
typedef struct {
    CLLocationDegrees latitude;
    CLLocationDegrees longitude;
} CLLocationCoordinate2D;
```

Fields

`latitude`
: The latitude in degrees. Positive values indicate latitudes north of the equator. Negative values indicate latitudes south of the equator.

`longitude`
: The longitude in degrees. Measurements are relative to the zero meridian, with positive values extending east of the meridian and negative values extending west of the meridian.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

CLLocationAccuracy

Represents the accuracy of a coordinate value in meters.

```
typedef double CLLocationAccuracy;
```

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

Accuracy Constants

Constant values you can use to specify the accuracy of a location.

```
extern const CLLocationAccuracy kCLLocationAccuracyBestForNavigation;
extern const CLLocationAccuracy kCLLocationAccuracyBest;
extern const CLLocationAccuracy kCLLocationAccuracyNearestTenMeters;
extern const CLLocationAccuracy kCLLocationAccuracyHundredMeters;
extern const CLLocationAccuracy kCLLocationAccuracyKilometer;
extern const CLLocationAccuracy kCLLocationAccuracyThreeKilometers;
```

Constants

`kCLLocationAccuracyBestForNavigation`

Use the highest possible accuracy and combine it with additional sensor data. This level of accuracy is intended for use in navigation applications that require precise position information at all times and are intended to be used only while the device is plugged in.

Available in iOS 4.0 and later.

Declared in `CLLocation.h`.

`kCLLocationAccuracyBest`

Use the highest-level of accuracy.

Available in iOS 2.0 and later.

Declared in `CLLocation.h`.

`kCLLocationAccuracyNearestTenMeters`

Accurate to within ten meters of the desired target.

Available in iOS 2.0 and later.

Declared in `CLLocation.h`.

`kCLLocationAccuracyHundredMeters`

Accurate to within one hundred meters.

Available in iOS 2.0 and later.

Declared in `CLLocation.h`.

`kCLLocationAccuracyKilometer`

Accurate to the nearest kilometer.

Available in iOS 2.0 and later.

Declared in `CLLocation.h`.

`kCLLocationAccuracyThreeKilometers`

Accurate to the nearest three kilometers.

Available in iOS 2.0 and later.

Declared in `CLLocation.h`.

CLLocationSpeed

Represents the speed at which the device is moving in meters per second.

```
typedef double CLLocationSpeed;
```

Availability

Available in iOS 2.2 and later.

Declared In
`CLLocation.h`

CLLocationDirection

Represents a direction that is measured in degrees relative to true north.

```
typedef double CLLocationDirection;
```

Discussion
Direction values are measured in degrees starting at due north and continue clockwise around the compass. Thus, north is 0 degrees, east is 90 degrees, south is 180 degrees, and so on. A negative value indicates an invalid direction.

Availability
Available in iOS 2.2 and later.

Declared In
`CLLocation.h`

Specifying an Invalid Coordinate

Use this constant whenever you want to indicate that a coordinate is invalid.

```
const CLLocationCoordinate2D kCLLocationCoordinate2DInvalid
```

Constants
`kCLLocationCoordinate2DInvalid`
An invalid coordinate value.

Available in iOS 4.0 and later.

Declared in `CLLocation.h`.

CLLocationManager Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/CoreLocation.framework
Availability	Available in iOS 2.0 and later.
Declared in	CLLocationManager.h

Overview

The `CLLocationManager` class defines the interface for configuring the delivery of location- and heading-related events to your application. You use an instance of this class to establish the parameters that determine when location and heading events should be delivered and to start and stop the actual delivery of those events. You can also use a location manager object to retrieve the most recent location and heading data.

A location manager object provides support for the following location-related activities:

- Tracking large or small changes in the user's current location with a configurable degree of accuracy. (iOS and Mac OS X)
- Reporting heading changes from the onboard compass. (iOS only)
- Monitoring distinct regions of interest and generating location events when the user enters or leaves those regions. (iOS only)

Some location services require the presence of specific hardware on the given device. For example, heading information is available only for devices that contain a hardware compass. This class defines several methods that you can use to determine which services are currently available.

Important: In addition to hardware not being available, the user has the option of denying an application's access to location service data. During its initial uses by an application, the Core Location framework prompts the user to confirm that using the location service is acceptable. If the user denies the request, the `CLLocationManager` object reports an appropriate error to its delegate during future requests.

For the services you do use, you should configure any properties associated with that service accurately. The location manager object manages power aggressively by turning off hardware when it is not needed. For example, setting the desired accuracy for location events to one kilometer gives the location manager the flexibility to turn off GPS hardware and rely solely on the WiFi or cell radios. This can lead to significant power savings.

To configure and use a `CLLocationManager` object to deliver events:

1. Always check to see whether the desired services are available before starting any services and abandon the operation if they are not.
2. Create an instance of the `CLLocationManager` class.
3. Assign a custom object to the `delegate` (page 29) property. This object must conform to the `CLLocationManagerDelegate` protocol.
4. Configure any additional properties relevant to the desired service.
5. Call the appropriate start method to begin the delivery of events.

All location- and heading-related updates are delivered to the associated delegate object, which is a custom object that you provide. The delegate object must conform to the `CLLocationManagerDelegate` protocol and implement the appropriate methods. For information about the methods of this protocol, see *CLLocationManagerDelegate Protocol Reference*.

Getting the User's Current Location

There are two options for configuring location-related services:

- You can use the standard location services, which allow you to specify the desired accuracy of the location data and receive updates as the location changes. Standard location services are available in all versions of iOS and in Mac OS X 10.6 and later.
- You can request events for significant location changes only. This provides a more limited set of tracking options but offers tremendous power savings and the ability to receive location updates even if your application is not running. This service is available only in iOS 4.0 and later and requires a device with a cellular radio.

You start standard location services by calling the `startUpdatingLocation` (page 40) method. This service is most appropriate for applications that need more fine-grained control over the delivery of location events. Specifically, it takes into account the values in the `desiredAccuracy` (page 30) and `distanceFilter` (page 30) property to determine when to deliver new events. This is most appropriate for navigation applications or any application where high-precision location data or a regular stream of updates is required. However, these services typically also require the location-tracking hardware to be enabled for longer periods of time, which can result in higher power usage.

For applications that do not need a regular stream of location events, you should consider using the `startMonitoringSignificantLocationChanges` (page 38) method to start the delivery of events instead. This method is more appropriate for the majority of applications that just need an initial user location fix and need updates only when the user moves a significant distance. This interface delivers new events only when it detects changes to the device's associated cell towers, resulting in less frequent updates and significantly better power usage.

Regardless of which location service you use, location data is reported to your application via the location manager's associated delegate object. Because it can take several seconds to return an initial location, the location manager typically delivers the previously cached location data immediately and then delivers more up-to-date location data as it becomes available. Therefore it is always a good idea to check the timestamp of any location object before taking any actions. If both location services are enabled simultaneously, they deliver events using the same set of delegate methods.

Using Regions to Monitor Boundary Crossings

In iOS 4.0 and later, you can use the region monitoring service to define the boundaries for multiple geographical regions. After registering a region using the `startMonitoringForRegion:desiredAccuracy:` (page 37) method, the location manager tracks movement across the region's boundary and reports that movement to its delegate. You might use region monitoring to alert the user to approaching landmarks or to provide other relevant information. For example, upon approaching a dry cleaners, an application could notify the user to pick up any clothes that had been dropped off and are now ready.

The regions you register with the location manager persist between launches of your application. If a region crossing occurs while your application is not running, the system automatically wakes up your application (or relaunches it) in the background so that it can process the event. When relaunched, all of the regions you configured previously are made available in the `monitoredRegions` (page 34) property of any location manager objects you create.

The region monitoring service operates independently of any location services in use by your application. You may use it in conjunction with any of the other services. Region monitoring is not available in Mac OS X and is not supported on all devices. Use the `regionMonitoringAvailable` (page 36) and `regionMonitoringEnabled` (page 36) class methods to determine if region monitoring can be used.

Configuring Heading-Related Services

In iOS, a device with the appropriate hardware may also report heading information. When the value in the `headingAvailable` (page 31) property is `YES`, you can use a location manager object to retrieve heading information. To begin the delivery of heading-related events, assign a delegate to the location manager object and call the location manager's `startUpdatingHeading` (page 39) method. If location updates are also enabled, the location manager returns both the true heading and magnetic heading values. If location updates are not enabled, the location manager returns only the magnetic heading value. These features are not available in Mac OS X.

Tasks

Accessing the Delegate

`delegate` (page 29) *property*
The delegate object you want to receive update events.

Determining the Availability of Services

+ `locationServicesEnabled` (page 35)
    Returns a Boolean value indicating whether location services are enabled on the device.

+ `significantLocationChangeMonitoringAvailable` (page 37)
    Returns a Boolean value indicating whether significant location change tracking is available.

+ `headingAvailable` (page 35)
    Returns a Boolean value indicating whether the location manager is able to generate heading-related events.

+ `regionMonitoringAvailable` (page 36)
    Returns a Boolean indicating whether region monitoring is supported on the current device.

+ `regionMonitoringEnabled` (page 36)
    Returns a Boolean indicating whether region monitoring is currently enabled.

`locationServicesEnabled` (page 33) *property*
    A Boolean value indicating whether location services are enabled on the device. (read-only) (Deprecated. Use the `locationServicesEnabled` (page 35) class method instead.)

`headingAvailable` (page 31) *property* Deprecated in iOS 4.0
    A Boolean value indicating whether the location manager is able to generate heading-related events. (read-only) (Deprecated. Use the `headingAvailable` (page 35) class method instead.)

Initiating Standard Location Updates

- `startUpdatingLocation` (page 40)
    Starts the generation of updates that report the user's current location.

- `stopUpdatingLocation` (page 42)
    Stops the generation of location updates.

`distanceFilter` (page 30) *property*
    The minimum distance (measured in meters) a device must move laterally before an update event is generated.

`desiredAccuracy` (page 30) *property*
    The desired accuracy of the location data.

Initiating Significant Location Updates

- `startMonitoringSignificantLocationChanges` (page 38)
    Starts the generation of updates based on significant location changes.

- `stopMonitoringSignificantLocationChanges` (page 41)
    Stops the delivery of location events based on significant location changes.

Initiating Heading Updates

- `startUpdatingHeading` (page 39)
    Starts the generation of updates that report the user's current heading.

- `stopUpdatingHeading` (page 41)
  Stops the generation of heading updates.
- `dismissHeadingCalibrationDisplay` (page 37)
  Dismisses the heading calibration view from the screen immediately.

`headingFilter` (page 32) *property*
  The minimum angular change (measured in degrees) required to generate new heading events.

`headingOrientation` (page 32) *property*
  The device orientation to use when computing heading values.

Initiating Region Monitoring

- `startMonitoringForRegion:desiredAccuracy:` (page 37)
  Starts monitoring the specified region for boundary crossings.
- `stopMonitoringForRegion:` (page 40)
  Stops monitoring the specified region.

`monitoredRegions` (page 34) *property*
  The set of shared regions monitored by all location manager objects. (read-only)

`maximumRegionMonitoringDistance` (page 34) *property*
  The largest boundary distance that can be assigned to a region. (read-only)

Getting Recently Retrieved Data

`location` (page 32) *property*
  The most recently retrieved user location. (read-only)

`heading` (page 31) *property*
  The most recently reported heading. (read-only)

Describing Your Application’s Services to the User

`purpose` (page 34) *property*
  An application-provided string that describes the reason for using location services.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

delegate

The delegate object you want to receive update events.

```
@property(assign, nonatomic) id<CLLocationManagerDelegate> delegate
```

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocationManager.h`

desiredAccuracy

The desired accuracy of the location data.

```
@property(assign, nonatomic) CLLocationAccuracy desiredAccuracy
```

Discussion
The receiver does its best to achieve the requested accuracy; however, the actual accuracy is not guaranteed.

You should assign a value to this property that is appropriate for your usage scenario. In other words, if you need the current location only within a few kilometers, you should not specify `kCLLocationAccuracyBest` (page 23) for the accuracy. Determining a location with greater accuracy requires more time and more power.

When requesting high-accuracy location data, the initial event delivered by the location service may not have the accuracy you requested. The location service delivers the initial event as quickly as possible. It then continues to determine the location with the accuracy you requested and delivers additional events, as necessary, when that data is available.

The default value of this property is `kCLLocationAccuracyBest`.

This property is used only in conjunction with the standard location services and is not used when monitoring significant location changes.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocationManager.h`

distanceFilter

The minimum distance (measured in meters) a device must move laterally before an update event is generated.

```
@property(assign, nonatomic) CLLocationDistance distanceFilter
```

Discussion

This distance is measured relative to the previously delivered location. Use the value `kCLDistanceFilterNone` (page 42) to be notified of all movements. The default value of this property is `kCLDistanceFilterNone`.

This property is used only in conjunction with the standard location services and is not used when monitoring significant location changes.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability

Available in iOS 2.0 and later.

Declared In

`CLLocationManager.h`

heading

The most recently reported heading. (read-only)

```
@property(readonly, nonatomic) CLHeading *heading
```

Discussion

The value of this property is `nil` if heading updates have never been initiated.

Availability

Available in iOS 4.0 and later.

Declared In

`CLLocationManager.h`

headingAvailable

A Boolean value indicating whether the location manager is able to generate heading-related events. (read-only) (Deprecated in iOS 4.0. Use the `headingAvailable` (page 35) class method instead.)

```
@property(readonly, nonatomic) BOOL headingAvailable
```

Discussion

Heading data may not be available on all iOS-based devices. You should check the value of this property before asking the location manager to deliver heading-related events.

Availability

Available in iOS 3.0 and later.

Deprecated in iOS 4.0.

See Also

`– startUpdatingHeading` (page 39)

Declared In
`CLLocationManager.h`

headingFilter

The minimum angular change (measured in degrees) required to generate new heading events.

```
@property(assign, nonatomic) CLLocationDegrees headingFilter
```

Discussion
The angular distance is measured relative to the last delivered heading event. Use the value `kCLHeadingFilterNone` (page 43) to be notified of all movements. The default value of this property is `kCLHeadingFilterNone`.

Availability
Available in iOS 3.0 and later.

Declared In
`CLLocationManager.h`

headingOrientation

The device orientation to use when computing heading values.

```
@property(assign, nonatomic) CLDeviceOrientation headingOrientation
```

Discussion
When computing heading values, the location manager assumes that the top of the device in portrait mode represents due north (0 degrees) by default. For applications that run in other orientations, this may not always be the most convenient orientation. This property allows you to specify which device orientation you want the location manager to use as the reference point for due north.

Although you can set the value of this property to `CLDeviceOrientationUnknown` (page 44), `CLDeviceOrientationFaceUp` (page 45), or `CLDeviceOrientationFaceDown` (page 45), doing so has no effect on the orientation reference point. The original reference point is retained instead.

Changing the value in this property affects only those heading values reported after the change is made.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManager.h`

location

The most recently retrieved user location. (read-only)

```
@property(readonly, nonatomic) CLLocation *location
```

Discussion
The value of this property is `nil` if no location data has ever been retrieved.

In iOS 4.0 and later, this property may contain a more recent location object at launch time. Specifically, if significant location updates are running and your application is terminated, this property is updated with the most recent location data when your application is relaunched (and you create a new location manager object). This location data may be more recent than the last location event processed by your application.

It is always a good idea to check the timestamp of the location stored in this property. If the receiver is currently gathering location data, but the minimum distance filter is large, the returned location might be relatively old. If it is, you can stop the receiver and start it again to force an update.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability

Available in iOS 2.0 and later.

See Also

`– startUpdatingLocation` (page 40)

Declared In

`CLLocationManager.h`

locationServicesEnabled

A Boolean value indicating whether location services are enabled on the device. (read-only) (Deprecated in iOS 4.0. Use the `locationServicesEnabled` (page 35) class method instead.)

```
@property(readonly, nonatomic) BOOL locationServicesEnabled
```

Discussion

The user can enable or disable location services from the Settings application by toggling the Location Services switch in General.

You should check this property before starting location updates to determine whether the user has location services enabled for the current device. If this property contains the value `NO` and you start location updates anyway, the Core Location framework prompts the user with a confirmation alert asking whether location services should be reenabled.

Special Considerations

In iOS, this property is declared as `nonatomic`. In Mac OS X, it is declared as `atomic`.

Availability

Available in iOS 2.0 and later.

Deprecated in iOS 4.0.

See Also

`– startUpdatingLocation` (page 40)

Declared In

`CLLocationManager.h`

maximumRegionMonitoringDistance

The largest boundary distance that can be assigned to a region. (read-only)

```
@property(readonly, nonatomic) CLLocationDistance maximumRegionMonitoringDistance
```

Discussion
This property defines the largest boundary distance allowed from a region's center point. Attempting to monitor a region with a distance larger than this value causes the location manager to send a `kCLErrorRegionMonitoringFailure` (page 43) error to the delegate.

If region monitoring is unavailable or not supported, the value in this property is `-1`.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManager.h`

monitoredRegions

The set of shared regions monitored by all location manager objects. (read-only)

```
@property(readonly, nonatomic) NSSet *monitoredRegions
```

Discussion
You cannot add regions to this property directly. Instead, you must register regions by calling the `startMonitoringForRegion:desiredAccuracy:` (page 37) method. The regions in this property are shared by all instances of the `CLLocationManager` class in your application.

The objects in this set may not necessarily be the same objects you specified at registration time. Only the region data itself is maintained by the system. Therefore, the only way to uniquely identify a registered region is using its `identifier` property.

The location manager persists region data between launches of your application. If your application is terminated and then relaunched, the contents of this property are repopulated with region objects that contain the previously registered data.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManager.h`

purpose

An application-provided string that describes the reason for using location services.

```
@property(copy, nonatomic) NSString *purpose
```

Discussion
If this property is not `nil` and the system needs to ask for the user's consent to use location services, it displays the provided string. You can use this string to explain why your application is using location services.

You must set the value of this property prior to starting any location services. Because the string is ultimately displayed to the user, you should always load it from a localized strings file.

Availability
Available in iOS 3.2 and later.

Declared In
`CLLocationManager.h`

Class Methods

headingAvailable

Returns a Boolean value indicating whether the location manager is able to generate heading-related events.

```
+ (BOOL)headingAvailable
```

Return Value
`YES` if heading data is available or `NO` if it is not.

Discussion
Heading data may not be available on all iOS-based devices. You should check the value returned by this method before asking the location manager to deliver heading-related events.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManager.h`

locationServicesEnabled

Returns a Boolean value indicating whether location services are enabled on the device.

```
+ (BOOL)locationServicesEnabled
```

Return Value
`YES` if location services are enabled or `NO` if they are not.

Discussion
The user can enable or disable location services from the Settings application by toggling the Location Services switch in General.

You should check the return value of this method before starting location updates to determine whether the user has location services enabled for the current device. If this method returns `NO` and you start location updates anyway, the Core Location framework prompts the user with a confirmation panel asking whether location services should be reenabled.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManager.h`

regionMonitoringAvailable

Returns a Boolean indicating whether region monitoring is supported on the current device.

```
+ (BOOL)regionMonitoringAvailable
```

Return Value
`YES` if region monitoring is available or `NO` if it is not.

Discussion
Support for region monitoring may not be available on all devices and models. You should check the value of this property before attempting to set up any regions or initiate region monitoring.

Even if region monitoring support is present on a device, it may still be unavailable because the user disabled it for the current application or for all applications.

Availability
Available in iOS 4.0 and later.

See Also
`+ regionMonitoringEnabled` (page 36)

Declared In
`CLLocationManager.h`

regionMonitoringEnabled

Returns a Boolean indicating whether region monitoring is currently enabled.

```
+ (BOOL)regionMonitoringEnabled
```

Return Value
`YES` if region monitoring is available and is currently enabled or `NO` if it is unavailable or not enabled.

Discussion
The user can enable or disable location services (including region monitoring) altogether from the Settings application by toggling the switch in Settings > General > Location Services.

You should check the return value of this method before starting region monitoring updates to determine if the user currently allows location services to be used at all. If this method returns `NO` and you start region monitoring updates anyway, the Core Location framework prompts the user with a confirmation panel asking whether location services should be reenabled.

This method does not check to see if region monitoring capabilities are actually supported by the device. Therefore, you should also check the return value of the `regionMonitoringAvailable` class method before attempting to start region monitoring services.

Availability
Available in iOS 4.0 and later.

See Also
`+ regionMonitoringAvailable` (page 36)

Declared In
`CLLocationManager.h`

significantLocationChangeMonitoringAvailable

Returns a Boolean value indicating whether significant location change tracking is available.

```
+ (BOOL)significantLocationChangeMonitoringAvailable
```

Return Value
`YES` if location change monitoring is available or `NO` if it is not.

Discussion
This method indicates whether the device is able to report updates based on significant location changes only. (This primarily involves detecting changes in the cell tower currently associated with the device.) This capability provides tremendous power savings for applications that want to track a user's approximate location and do not need highly accurate position information.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManager.h`

Instance Methods

dismissHeadingCalibrationDisplay

Dismisses the heading calibration view from the screen immediately.

```
- (void)dismissHeadingCalibrationDisplay
```

Discussion
Core Location uses the heading calibration alert to calibrate the available heading hardware as needed. The display of this view is automatic, assuming your delegate supports displaying the view at all. If the view is displayed, you can use this method to dismiss it after an appropriate amount of time to ensure that your application's user interface is not unduly disrupted.

Availability
Available in iOS 3.0 and later.

Declared In
`CLLocationManager.h`

startMonitoringForRegion:desiredAccuracy:

Starts monitoring the specified region for boundary crossings.

```
- (void)startMonitoringForRegion:(CLRegion
    *)regiondesiredAccuracy:(CLLocationAccuracy)accuracy
```

Parameters

region

The region object that defines the boundary to monitor. This parameter must not be `nil`.

accuracy

The distance past the border (measured in meters) at which to generate notifications. You can use this value to prevent the delivery of multiple notifications when the user is close to the border's edge.

Discussion

You must call this method separately for each region you want to monitor. If an existing region with the same identifier is already being monitored by the application, the old region is replaced by the new one. The regions you add using this method are shared by all location manager objects in your application and stored in the `monitoredRegions` (page 34) property.

If you begin monitoring a region and your application is subsequently terminated, the system automatically relaunches it into the background if the region boundary is crossed. In such a case, the options dictionary passed to the `application:didFinishLaunchingWithOptions:` method of your application delegate contains the `UIApplicationLaunchOptionsLocationKey` key to indicate that your application was launched because of a location-related event. In addition, creating a new location manager and assigning a delegate results in the delivery of the corresponding region messages. The newly created location manager's `location` (page 32) property also contains the current location even if location services are not enabled.

Region events are delivered to the `locationManager:didEnterRegion:` (page 54) and `locationManager:didExitRegion:` (page 54) methods of your delegate. If there is an error, the location manager calls the `locationManager:monitoringDidFailForRegion:withError:` (page 57) method of your delegate instead.

Availability

Available in iOS 4.0 and later.

Declared In

`CLLocationManager.h`

startMonitoringSignificantLocationChanges

Starts the generation of updates based on significant location changes.

```
- (void)startMonitoringSignificantLocationChanges
```

Discussion

This method initiates the delivery of location events asynchronously, returning shortly after you call it. Location events are delivered to your delegate's `locationManager:didUpdateToLocation:fromLocation:` (page 56) method. The first event to be delivered is usually the most recently cached location event (if any) but may be a newer event in some circumstances. Obtaining a current location fix may take several additional seconds, so be sure to check the timestamps on the location events in your delegate method.

After returning a current location fix, the receiver generates update events only when a significant change in the user's location is detected. For example, it might generate a new event when the device becomes associated with a different cell tower. It does not rely on the value in the `distanceFilter` (page 30) property to generate events. Calling this method several times in succession does not automatically result in new events being generated. Calling `stopMonitoringSignificantLocationChanges` in between, however, does cause a new initial event to be sent the next time you call this method.

If you start this service and your application is subsequently terminated, the system automatically relaunches the application into the background if a new event arrives. In such a case, the options dictionary passed to the `application:didFinishLaunchingWithOptions:` method of your application delegate contains the key `UIApplicationLaunchOptionsLocationKey` to indicate that your application was launched because of a location event. Upon relaunch, you must still configure a location manager object and call this method to continue receiving location events. When you restart location services, the current event is delivered to your delegate immediately. In addition, the `location` (page 32) property of your location manager object is populated with the most recent location object even before you start location services.

In addition to your delegate object implementing the `locationManager:didUpdateToLocation:fromLocation:` method, it should also implement the `locationManager:didFailWithError:` (page 55) method to respond to potential errors.

Availability
Available in iOS 4.0 and later.

See Also
`– stopMonitoringSignificantLocationChanges` (page 41)

Declared In
`CLLocationManager.h`

startUpdatingHeading

Starts the generation of updates that report the user's current heading.

```
- (void)startUpdatingHeading
```

Discussion
This method returns immediately. Calling this method when the receiver is stopped causes it to obtain an initial heading and notify your delegate. After that, the receiver generates update events when the value in the `headingFilter` property is exceeded.

Before calling this method, you should always check the `headingAvailable` property to see whether heading information is supported on the current device. If heading information is not supported, calling this method has no effect and does not result in the delivery of events to your delegate.

Calling this method several times in succession does not automatically result in new events being generated. Calling `stopUpdatingHeading` in between, however, does cause a new initial event to be sent the next time you call this method.

If you start this service and your application is suspended, the system stops the delivery of events until your application starts running again (either in the foreground or background). If your application is terminated, the delivery of new heading events stops altogether and must be restarted by your code when the application is relaunched.

Heading events are delivered to the `locationManager:didUpdateHeading:` (page 56) method of your delegate. If there is an error, the location manager calls the `locationManager:didFailWithError:` (page 55) method of your delegate instead.

Availability
Available in iOS 3.0 and later.

See Also
`- stopUpdatingHeading` (page 41)
`@property headingAvailable` (page 31)

Declared In
`CLLocationManager.h`

startUpdatingLocation

Starts the generation of updates that report the user's current location.

```
- (void)startUpdatingLocation
```

Discussion
This method returns immediately. Calling this method causes the location manager to obtain an initial location fix (which may take several seconds) and notify your delegate by calling its `locationManager:didUpdateToLocation:fromLocation:` (page 56) method. After that, the receiver generates update events primarily when the value in the `distanceFilter` property is exceeded. Updates may be delivered in other situations though. For example, the receiver may send another notification if the hardware gathers a more accurate location reading.

Calling this method several times in succession does not automatically result in new events being generated. Calling `stopUpdatingLocation` in between, however, does cause a new initial event to be sent the next time you call this method.

If you start this service and your application is suspended, the system stops the delivery of events until your application starts running again (either in the foreground or background). If your application is terminated, the delivery of new location events stops altogether. Therefore, if your application needs to receive location events while in the background, it must include the `UIBackgroundModes` key (with the `location` value) in its `Info.plist` file.

In addition to your delegate object implementing the `locationManager:didUpdateToLocation:fromLocation:` method, it should also implement the `locationManager:didFailWithError:` (page 55) method to respond to potential errors.

Availability
Available in iOS 2.0 and later.

See Also
`- stopUpdatingLocation` (page 42)
`@property locationServicesEnabled` (page 33)
`@property distanceFilter` (page 30)

Declared In
`CLLocationManager.h`

stopMonitoringForRegion:

Stops monitoring the specified region.

```
- (void)stopMonitoringForRegion:(CLRegion *)region
```

Parameters

`region`
The region object currently being monitored. This parameter must not be `nil`.

Discussion
If the specified region object is not currently being monitored, this method has no effect.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManager.h`

stopMonitoringSignificantLocationChanges

Stops the delivery of location events based on significant location changes.

```
- (void)stopMonitoringSignificantLocationChanges
```

Discussion
Use this method to stop the delivery of location events that was started using the `startMonitoringSignificantLocationChanges` method.

Availability
Available in iOS 4.0 and later.

See Also
`– startMonitoringSignificantLocationChanges` (page 38)

Declared In
`CLLocationManager.h`

stopUpdatingHeading

Stops the generation of heading updates.

```
- (void)stopUpdatingHeading
```

Discussion
You should call this method whenever your code no longer needs to receive heading-related events. Disabling event delivery gives the receiver the option of disabling the appropriate hardware (and thereby saving power) when no clients need location data. You can always restart the generation of heading updates by calling the `startUpdatingHeading` method again.

Availability
Available in iOS 3.0 and later.

See Also
`– startUpdatingHeading` (page 39)

Declared In
`CLLocationManager.h`

stopUpdatingLocation

Stops the generation of location updates.

```
- (void)stopUpdatingLocation
```

Discussion
You should call this method whenever your code no longer needs to receive location-related events. Disabling event delivery gives the receiver the option of disabling the appropriate hardware (and thereby saving power) when no clients need location data. You can always restart the generation of location updates by calling the `startUpdatingLocation` method again.

Availability
Available in iOS 2.0 and later.

See Also
`- startUpdatingLocation` (page 40)

Declared In
`CLLocationManager.h`

Constants

CLLocationDistance

A distance measurement (in meters) from an existing location.

```
typedef double CLLocationDistance;
```

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocation.h`

Distance Filter Value

This constant indicates the minimum distance required before an event is generated.

```
extern const CLLocationDistance kCLDistanceFilterNone;
```

Constants
`kCLDistanceFilterNone`

All movements are reported.

Available in iOS 2.0 and later.

Declared in `CLLocation.h`.

Heading Filter Value

This constant indicates the minimum heading change before an event is generated.

```
const CLLocationDegrees kCLHeadingFilterNone;
```

Constants

`kCLHeadingFilterNone`

All heading changes are reported.

Available in iOS 3.0 and later.

Declared in `CLHeading.h`.

CLError

Error codes returned by the location manager object.

```
typedef enum {
   kCLErrorLocationUnknown  = 0,
   kCLErrorDenied,
   kCLErrorNetwork,
   kCLErrorHeadingFailure,
   kCLErrorRegionMonitoringDenied,
   kCLErrorRegionMonitoringFailure,
   kCLErrorRegionMonitoringSetupDelayed
} CLError;
```

Constants

`kCLErrorLocationUnknown`

The location manager was unable to obtain a location value right now.

Available in iOS 2.0 and later.

Declared in `CLError.h`.

`kCLErrorDenied`

Access to the location service was denied by the user.

Available in iOS 2.0 and later.

Declared in `CLError.h`.

`kCLErrorNetwork`

The network was unavailable or a network error occurred.

Available in iOS 3.0 and later.

Declared in `CLError.h`.

`kCLErrorHeadingFailure`

The heading could not be determined.

Available in iOS 3.0 and later.

Declared in `CLError.h`.

`kCLErrorRegionMonitoringDenied`

Access to the region monitoring service was denied by the user.

Available in iOS 4.0 and later.

Declared in `CLError.h`.

`kCLErrorRegionMonitoringFailure`

A registered region cannot be monitored.

Available in iOS 4.0 and later.

Declared in `CLError.h`.

`kCLErrorRegionMonitoringSetupDelayed`

Core Location could not initialize the region monitoring feature immediately.

Available in iOS 4.0 and later.

Declared in `CLError.h`.

Discussion

Errors are delivered to the delegate using an `NSError` object.

kCLErrorDomain

The domain for Core Location errors.

```
extern NSString *const kCLErrorDomain;
```

Constants

`kCLErrorDomain`

The domain for Core Location errors. This value is used in the `NSError` class.

Available in iOS 2.0 and later.

Declared in `CLErrorDomain.h`.

CLDeviceOrientation

The physical orientation of the device.

```
typedef enum {
    CLDeviceOrientationUnknown = 0,
    CLDeviceOrientationPortrait,
    CLDeviceOrientationPortraitUpsideDown,
    CLDeviceOrientationLandscapeLeft,
    CLDeviceOrientationLandscapeRight,
    CLDeviceOrientationFaceUp,
    CLDeviceOrientationFaceDown
} CLDeviceOrientation;
```

Constants

`CLDeviceOrientationUnknown`

The orientation is currently not known.

Available in iOS 4.0 and later.

Declared in `CLLocationManager.h`.

`CLDeviceOrientationPortrait`

The device is in portrait mode, with the device held upright and the home button at the bottom.

Available in iOS 4.0 and later.

Declared in `CLLocationManager.h`.

`CLDeviceOrientationPortraitUpsideDown`

The device is in portrait mode but upside down, with the device held upright and the home button at the top.

Available in iOS 4.0 and later.

Declared in `CLLocationManager.h`.

`CLDeviceOrientationLandscapeLeft`
: The device is in landscape mode, with the device held upright and the home button on the right side.

  Available in iOS 4.0 and later.

  Declared in `CLLocationManager.h`.

`CLDeviceOrientationLandscapeRight`
: The device is in landscape mode, with the device held upright and the home button on the left side.

  Available in iOS 4.0 and later.

  Declared in `CLLocationManager.h`.

`CLDeviceOrientationFaceUp`
: The device is held parallel to the ground with the screen facing upwards.

  Available in iOS 4.0 and later.

  Declared in `CLLocationManager.h`.

`CLDeviceOrientationFaceDown`
: The device is held parallel to the ground with the screen facing downwards.

  Available in iOS 4.0 and later.

  Declared in `CLLocationManager.h`.

CLRegion Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/CoreLocation.framework
Availability	Available in iOS 4.0 and later.
Declared in	CLLocation.h

Overview

The `CLRegion` class defines a geographical area that can be tracked. When an instance of this class is registered with a `CLLocationManager` object, the location manager generates an appropriate event whenever the user crosses the boundaries of the defined area.

To use this class, create an instance of it and use the `startMonitoringForRegion:desiredAccuracy:` (page 37) method of a `CLLocationManager` object to begin monitoring it.

Tasks

Initializing a Circular Region

`– initCircularRegionWithCenter:radius:identifier:` (page 49)
Initializes and returns a region object defining a circular area.

Accessing a Region's Attributes

`identifier` (page 48) *property*
The identifier for the region object. (read-only)

`center` (page 48) *property*
The center point of the region. (read-only)

`radius` (page 48) *property* Deprecated in iOS 4.0
The radius (measured in meters) that defines the region's outer boundary. (read-only)

Hit-Testing in a Region

`– containsCoordinate:` (page 49)
Returns a Boolean value indicating whether the region contains the specified coordinate.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

center

The center point of the region. (read-only)

```
@property(readonly, nonatomic) CLLocationCoordinate2D center
```

Availability
Available in iOS 4.0 and later.

Declared In
`CLRegion.h`

identifier

The identifier for the region object. (read-only)

```
@property(readonly, nonatomic) NSString *identifier
```

Discussion
This is a value that you specify and can use to identify this region inside your application.

Availability
Available in iOS 4.0 and later.

Declared In
`CLRegion.h`

radius

The radius (measured in meters) that defines the region’s outer boundary. (read-only)

```
@property(readonly, nonatomic) CLLocationDistance radius
```

Availability
Available in iOS 4.0 and later.

Declared In
`CLRegion.h`

Instance Methods

containsCoordinate:

Returns a Boolean value indicating whether the region contains the specified coordinate.

```
- (BOOL)containsCoordinate:(CLLocationCoordinate2D)coordinate
```

Parameters

coordinate

The coordinate to test against the region.

Return Value

`YES` if the coordinate lies within the region's boundaries or `NO` if it does not.

Availability

Available in iOS 4.0 and later.

Declared In

`CLRegion.h`

initCircularRegionWithCenter:radius:identifier:

Initializes and returns a region object defining a circular area.

```
- (id)initCircularRegionWithCenter:(CLLocationCoordinate2D)center
    radius:(CLLocationDistance)radius identifier:(NSString *)identifier
```

Parameters

center

The center point of the region.

radius

The distance (measured in meters) from the center point that marks the boundary of the region.

identifier

A unique identifier to associate with the region object. You use this identifier to differentiate regions within your application. This value must not be `nil`.

Return Value

An initialized region object.

Availability

Available in iOS 4.0 and later.

Declared In

`CLRegion.h`

Protocols

CLLocationManagerDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Frameworks/CoreLocation.framework
Availability	Available in iOS 2.0 and later.
Declared in	CLLocationManagerDelegate.h

Overview

The `CLLocationManagerDelegate` protocol defines the methods used to receive location and heading updates from a `CLLocationManager` object. The methods of this protocol are optional.

Upon receiving a successful location or heading update, you can use the result to update your user interface or perform other actions. Similarly, if the location or heading could not be determined, you might want to stop updates for a short period of time and try again later. You can use the `stopUpdatingLocation` (page 42), `stopMonitoringSignificantLocationChanges` (page 41) `stopUpdatingHeading` (page 41), or `stopMonitoringForRegion:` (page 40) methods of `CLLocationManager` to stop location, heading, and region updates.

The methods of your delegate object are called from the thread in which you started the corresponding location services. That thread must itself have an active run loop, like the one found in your application's main thread.

Tasks

Responding to Location Events

`– locationManager:didUpdateToLocation:fromLocation:` (page 56)
Tells the delegate that a new location value is available.

`– locationManager:didFailWithError:` (page 55)
Tells the delegate that the location manager was unable to retrieve a location value.

Responding to Heading Events

`– locationManager:didUpdateHeading:` (page 56)
Tells the delegate that the location manager received updated heading information.

`– locationManagerShouldDisplayHeadingCalibration:` (page 57)
: Asks the delegate whether the heading calibration alert should be displayed.

Responding to Region Events

`– locationManager:didEnterRegion:` (page 54)
: Tells the delegate that the user entered the specified region.

`– locationManager:didExitRegion:` (page 54)
: Tells the delegate that the user left the specified region.

`– locationManager:monitoringDidFailForRegion:withError:` (page 57)
: Tells the delegate that a region monitoring error occurred.

Instance Methods

locationManager:didEnterRegion:

Tells the delegate that the user entered the specified region.

```
- (void)locationManager:(CLLocationManager *)managerdidEnterRegion:(CLRegion *)region
```

Parameters

manager
: The location manager object reporting the event.

region
: An object containing information about the region that was entered.

Discussion

Because regions are a shared application resource, every active location manager object delivers this message to its associated delegate. It does not matter which location manager actually registered the specified region. And if multiple location managers share a delegate object, that delegate receives the message multiple times.

The region object provided may not be the same one that was registered. As a result, you should never perform pointer-level comparisons to determine equality. Instead, use the region's identifier string to determine if your delegate should respond.

Availability

Available in iOS 4.0 and later.

Declared In

`CLLocationManagerDelegate.h`

locationManager:didExitRegion:

Tells the delegate that the user left the specified region.

```
- (void)locationManager:(CLLocationManager *)managerdidExitRegion:(CLRegion *)region
```

Parameters

manager
: The location manager object reporting the event.

region
: An object containing information about the region that was exited.

Discussion

Because regions are a shared application resource, every active location manager object delivers this message to its associated delegate. It does not matter which location manager actually registered the specified region. And if multiple location managers share a delegate object, that delegate receives the message multiple times.

The region object provided may not be the same one that was registered. As a result, you should never perform pointer-level comparisons to determine equality. Instead, use the region's identifier string to determine if your delegate should respond.

Availability

Available in iOS 4.0 and later.

Declared In

`CLLocationManagerDelegate.h`

locationManager:didFailWithError:

Tells the delegate that the location manager was unable to retrieve a location value.

```
- (void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError
    *)error
```

Parameters

manager
: The location manager object that was unable to retrieve the location.

error
: The error object containing the reason why the location or heading could not be retrieved.

Discussion

Implementation of this method is optional. You should implement this method, however.

If the location service is unable to retrieve a location fix right away, it reports a `kCLErrorLocationUnknown` (page 43) error and keeps trying. In such a situation, you can simply ignore the error and wait for a new event.

If the user denies your application's use of the location service, this method reports a `kCLErrorDenied` (page 43) error. Upon receiving such an error, you should stop the location service.

If a heading could not be determined because of strong interference from nearby magnetic fields, this method returns `kCLErrorHeadingFailure` (page 43).

Availability

Available in iOS 2.0 and later.

See Also

`CLError` (page 43)

Declared In
`CLLocationManagerDelegate.h`

locationManager:didUpdateHeading:

Tells the delegate that the location manager received updated heading information.

```
- (void)locationManager:(CLLocationManager *)manager didUpdateHeading:(CLHeading
    *)newHeading
```

Parameters

manager
: The location manager object that generated the update event.

newHeading
: The new heading data.

Discussion
Implementation of this method is optional but expected if you start heading updates using the `startUpdatingHeading` (page 39) method.

The location manager object calls this method after you initially start the heading service. Subsequent events are delivered when the previously reported value changes by more than the value specified in the `headingFilter` (page 32) property of the location manager object.

Availability
Available in iOS 3.0 and later.

Declared In
`CLLocationManagerDelegate.h`

locationManager:didUpdateToLocation:fromLocation:

Tells the delegate that a new location value is available.

```
- (void)locationManager:(CLLocationManager *)manager didUpdateToLocation:(CLLocation
    *)newLocation fromLocation:(CLLocation *)oldLocation
```

Parameters

manager
: The location manager object that generated the update event.

newLocation
: The new location data.

oldLocation
: The location data from the previous update. If this is the first update event delivered by this location manager, this parameter is `nil`.

Discussion
Implementation of this method is optional. You should implement this method, however.

By the time this message is delivered to your delegate, the new location data is also available directly from the `CLLocationManager` object. The *`newLocation`* parameter may contain the data that was cached from a previous usage of the location service. You can use the `timestamp` (page 18) property of the location object to determine how recent the location data is.

Availability
Available in iOS 2.0 and later.

Declared In
`CLLocationManagerDelegate.h`

locationManager:monitoringDidFailForRegion:withError:

Tells the delegate that a region monitoring error occurred.

```
- (void)locationManager:(CLLocationManager
    *)managermonitoringDidFailForRegion:(CLRegion *)regionwithError:(NSError *)error
```

Parameters

`manager`
The location manager object reporting the event.

`region`
The region for which the error occurred.

`error`
An error object containing the error code that indicates why region monitoring failed.

Discussion
If an error occurs while trying to monitor a given region, the location manager sends this message to its delegate. Region monitoring might fail because the region itself cannot be monitored or because there was a more general failure in configuring the region monitoring service.

Although implementation of this method is optional, it is recommended that you implement it if you use region monitoring in your application.

Availability
Available in iOS 4.0 and later.

Declared In
`CLLocationManagerDelegate.h`

locationManagerShouldDisplayHeadingCalibration:

Asks the delegate whether the heading calibration alert should be displayed.

```
- (BOOL)locationManagerShouldDisplayHeadingCalibration:(CLLocationManager *)manager
```

Parameters

`manager`
The location manager object coordinating the display of the heading calibration alert.

Return Value
`YES` if you want to allow the heading calibration alert to be displayed or `NO` if you do not.

Discussion

Core Location may call this method in an effort to calibrate the onboard hardware used to determine heading values. Typically, Core Location calls this method at the following times:

- The first time heading updates are ever requested
- When Core Location observes a significant change in magnitude or inclination of the observed magnetic field

If you return `YES` from this method, Core Location displays the heading calibration alert on top of the current window immediately. The calibration alert prompts the user to move the device in a particular pattern so that Core Location can distinguish between the Earth's magnetic field and any local magnetic fields. The alert remains visible until calibration is complete or until you explicitly dismiss it by calling the `dismissHeadingCalibrationDisplay` (page 37) method. In this latter case, you can use this method to set up a timer and dismiss the interface after a specified amount of time has elapsed.

Note: The calibration process is able to filter out only those magnetic fields that move with the device. To calibrate a device that is near other sources of magnetic interference, the user must either move the device away from the source or move the source in conjunction with the device during the calibration process.

If you return `NO` from this method or do not provide an implementation for it in your delegate, Core Location does not display the heading calibration alert. Even if the alert is not displayed, calibration can still occur naturally when any interfering magnetic fields move away from the device. However, if the device is unable to calibrate itself for any reason, the value in the `headingAccuracy` (page 10) property of any subsequent events will reflect the uncalibrated readings.

Availability

Available in iOS 3.0 and later.

Declared In

`CLLocationManagerDelegate.h`

Functions

Core Location Functions Reference

Declared in	CLLocation.h
Companion guide	Location Awareness Programming Guide

Overview

The Core Location framework provides functions to help you work with coordinate values.

Functions

CLLocationCoordinate2DIsValid

Returns a Boolean indicating whether the specified coordinate is valid.

```
BOOL CLLocationCoordinate2DIsValid(
        CLLocationCoordinate2D coord)
```

Parameters

`coord`

A coordinate containing latitude and longitude values.

Return Value

`YES` if the coordinate is valid or `NO` if it is not.

Discussion

A coordinate is considered invalid if it meets at least one of the following criteria:

- Its latitude is greater than 90 degrees or less than -90 degrees.
- Its longitude is greater than 180 degrees or less than -180 degrees.

Availability

Available in iOS 4.0 and later.

Declared In

`CLLocation.h`

CLLocationCoordinate2DMake

Formats a latitude and longitude value into a coordinate data structure format.

```
CLLocationCoordinate2D CLLocationCoordinate2DMake(
        CLLocationDegrees latitude,
        CLLocationDegrees longitude)
```

Parameters

`latitude`

The latitude for the new coordinate.

`longitude`

The longitude for the new coordinate.

Return Value

A coordinate structure encompassing the latitude and longitude values.

Availability

Available in iOS 4.0 and later.

Declared In

`CLLocation.h`

Document Revision History

This table describes the changes to *Core Location Framework Reference*.

Date	Notes
2010-05-11	Added classes and functions introduced in iOS 4.0.
2009-07-28	Updated the document to reflect the availability of the interfaces in Mac OS X v10.6.
2009-05-07	Added the CLHeading class.
2008-03-12	New document that describes the classes and protocols for configuring and scheduling the delivery of location-related events.