CLLocationManager Class Reference

Data Management: Device Information

2010-05-14

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

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	CLError.h CLErrorDomain.h CLHeading.h CLLocation.h 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 10) 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 19) 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 10) and `distanceFilter` (page 11) 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 17) 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 16) 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 13) 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 15) and `regionMonitoringEnabled` (page 15) 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 25) 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 18) 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 10) *property*
The delegate object you want to receive update events.

Determining the Availability of Services

`+ locationServicesEnabled` (page 14)
Returns a Boolean value indicating whether location services are enabled on the device.

`+ significantLocationChangeMonitoringAvailable` (page 16)
Returns a Boolean value indicating whether significant location change tracking is available.

`+ headingAvailable` (page 14)
Returns a Boolean value indicating whether the location manager is able to generate heading-related events.

`+ regionMonitoringAvailable` (page 15)
Returns a Boolean indicating whether region monitoring is supported on the current device.

`+ regionMonitoringEnabled` (page 15)
Returns a Boolean indicating whether region monitoring is currently enabled.

`headingAvailable` (page 25) *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 14) class method instead.)

`locationServicesEnabled` (page 25) *property* Deprecated in iOS 4.0
A Boolean value indicating whether location services are enabled on the device. (read-only) (Deprecated. Use the `locationServicesEnabled` (page 14) class method instead.)

Initiating Standard Location Updates

`- startUpdatingLocation` (page 19)
Starts the generation of updates that report the user's current location.

`- stopUpdatingLocation` (page 21)
Stops the generation of location updates.

`distanceFilter` (page 11) *property*
The minimum distance (measured in meters) a device must move laterally before an update event is generated.

`desiredAccuracy` (page 10) *property*
The desired accuracy of the location data.

Initiating Significant Location Updates

- `startMonitoringSignificantLocationChanges` (page 17)
  Starts the generation of updates based on significant location changes.
- `stopMonitoringSignificantLocationChanges` (page 20)
  Stops the delivery of location events based on significant location changes.

Initiating Heading Updates

- `startUpdatingHeading` (page 18)
  Starts the generation of updates that report the user's current heading.
- `stopUpdatingHeading` (page 20)
  Stops the generation of heading updates.
- `dismissHeadingCalibrationDisplay` (page 16)
  Dismisses the heading calibration view from the screen immediately.

`headingFilter` (page 11) *property*
The minimum angular change (measured in degrees) required to generate new heading events.

`headingOrientation` (page 12) *property*
The device orientation to use when computing heading values.

Initiating Region Monitoring

- `startMonitoringForRegion:desiredAccuracy:` (page 16)
  Starts monitoring the specified region for boundary crossings.
- `stopMonitoringForRegion:` (page 19)
  Stops monitoring the specified region.

`monitoredRegions` (page 13) *property*
The set of shared regions monitored by all location manager objects. (read-only)

`maximumRegionMonitoringDistance` (page 13) *property*
The largest boundary distance that can be assigned to a region. (read-only)

Getting Recently Retrieved Data

`location` (page 12) *property*
The most recently retrieved user location. (read-only)

`heading` (page 11) *property*
The most recently reported heading. (read-only)

Describing Your Application's Services to the User

`purpose` (page 13) *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` 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 21) 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`

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 22) 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 23), `CLDeviceOrientationFaceUp` (page 24), or `CLDeviceOrientationFaceDown` (page 24), 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 19)

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 22) 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 16) 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 15)

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 15)

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 13) 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 12) property also contains the current location even if location services are not enabled.

Region events are delivered to the `locationManager:didEnterRegion:` and `locationManager:didExitRegion:` methods of your delegate. If there is an error, the location manager calls the `locationManager:monitoringDidFailForRegion:withError:` 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:` 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 11) 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 12) 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:` method to respond to potential errors.

Availability
Available in iOS 4.0 and later.

See Also
`- stopMonitoringSignificantLocationChanges` (page 20)

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:` method of your delegate. If there is an error, the location manager calls the `locationManager:didFailWithError:` method of your delegate instead.

Availability
Available in iOS 3.0 and later.

See Also
- `stopUpdatingHeading` (page 20)
- `@property headingAvailable` (page 25)

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:` 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:` method to respond to potential errors.

Availability
Available in iOS 2.0 and later.

See Also
- `stopUpdatingLocation` (page 21)
- `@property locationServicesEnabled` (page 25)
- `@property distanceFilter` (page 11)

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 17)

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 18)

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 19)

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`.

Deprecated CLLocationManager Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in iOS 4.0

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 14) 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 18)

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 14) 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 19)

Declared In
`CLLocationManager.h`

Document Revision History

This table describes the changes to *CLLocationManager Class Reference*.

Date	Notes
2010-05-14	Updated to include symbols introduced in iOS 4.0.
2010-02-25	Updated for iOS 3.2.
2009-08-04	Updated the document to reflect the availability of the interfaces in Mac OS X v10.6.
2009-05-22	Updated for iOS 3.0.
2008-05-27	New document that describes the class for configuring the delivery of location-related events to an application.

Document Revision History