# Core Telephony Framework Reference

**Networking & Internet**

**2010-03-15**

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



Simultaneously published in the United States and Canada.

# Contents

# Introduction

| | |
|---|---|
| **Framework** | /System/Library/Frameworks/CoreTelephony.framework |
| **Header file directories** | /System/Library/Frameworks/CoreTelephony.framework/Headers |
| **Declared in** | CTCall.h<br>CTCallCenter.h<br>CTCarrier.h<br>CTTelephonyNetworkInfo.h |

Use the Core Telephony framework to obtain information about a user's home cellular service provider—that is, the provider with whom the user has an account. Carriers can use this information to write applications that provide services only for their own subscribers. You can also use this framework to obtain information about current cellular calls.

A `CTCarrier` object gives you information about the user's cellular service provider, such as whether it allows use of VoIP (Voice over Internet Protocol) on its network. A `CTCall` object gives you information about a current call, including a unique identifier and state information—dialing, incoming, connected, or disconnected.

Introduction

# Classes

PART I

# Classes

# CTCall Class Reference

| | |
|---|---|
| **Inherits from** | NSObject |
| **Conforms to** | NSObject (NSObject) |
| **Framework** | /System/Library/Frameworks/CoreTelephony.framework/ |
| **Availability** | Available in iOS 4.0 and later. |
| **Declared in** | |

## Overview

Use a cellular call's `CTCall` object to obtain an identifier for the call and to determine the call's state.

## Tasks

### Obtaining Information About a Cellular Call

`callID` (page 9) *property*
A unique identifier for the cellular call. (read-only)

`callState` (page 10) *property*
The state of the cellular call. (read-only)

## Properties

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

### callID

A unique identifier for the cellular call. (read-only)

```
@property (nonatomic, readonly, retain) NSString *callID
```

**Discussion**
Use this value to differentiate multiple active cellular calls.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCall.h`

### callState

The state of the cellular call. (read-only)

```
@property (nonatomic, readonly, retain) NSString *callState
```

**Discussion**
A cellular call's initial state is either `CTCallStateDialing` or `CTCallStateIncoming`. When the call is fully established for all parties involved, the state transitions to `CTCallStateConnected`. When the call is terminated, the state transitions to `CTCallStateDisconnected`.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCall.h`

## Constants

### Cellular Call States

States of cellular calls; one of *dialing*, *incoming*, *connected*, or *disconnected*.

```
extern NSString const *CTCallStateDialing;
extern NSString const *CTCallStateIncoming;
extern NSString const *CTCallStateConnected;
extern NSString const *CTCallStateDisconnected;
```

**Constants**

`CTCallStateDialing`
The call state, before connection is established, when the user initiates the call.
Available in iOS 4.0 and later.
Declared in `CTCall.h`.

`CTCallStateIncoming`
The call state, before connection is established, when a call is incoming but not yet answered by the user.
Available in iOS 4.0 and later.
Declared in `CTCall.h`.

`CTCallStateConnected`
The call state when the call is fully established for all parties involved.
Available in iOS 4.0 and later.
Declared in `CTCall.h`.

`CTCallStateDisconnected`

The call state upon call termination.

Available in iOS 4.0 and later.

Declared in `CTCall.h`.

# CTCallCenter Class Reference

| | |
|---|---|
| **Inherits from** | NSObject |
| **Conforms to** | NSObject (NSObject) |
| **Framework** | /System/Library/Frameworks/CoreTelephony.framework |
| **Availability** | Available in iOS 4.0 and later. |
| **Declared in** | |

## Overview

Use the `CTCallCenter` class to obtain a list of current cellular calls, and to respond to state changes for calls such as from a dialing state to a connected state. Such state changes are known as *cellular call events*.

## Tasks

### Responding to Cellular Call Events

`callEventHandler` (page 13) *property*
Dispatched when a call changes state.

`currentCalls` (page 14) *property*
An array representing the cellular calls in progress. (read-only)

## Properties

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

### callEventHandler

Dispatched when a call changes state.

```
@property (nonatomic, copy) void (^callEventHandler)(CTCall*);
```

**Discussion**
This property's block object is dispatched on the default priority global dispatch queue when a call changes state. To handle such call events, define a handler block in your application and assign it to this property. You must implement the handler block to support being invoked from any context.

If your application is active when a call event takes place, the system dispatches the event to your handler immediately. However, call events can also take place while your application is suspended. While it is suspended, your application does not receive call events. When your application resumes the active state, it receives a single call event for each call that changed state—no matter how many state changes the call experienced while your application was suspended. The single call event sent to your handler, upon your application returning to the active state, describes the call's state at that time.

For example, suppose your application changes from the active to the suspended state while a call is in the connected state. Suppose also that while your application is suspended, the call disconnects. When your application returns to the active state, you get a cellular call event indicating that the call is disconnected.

Here is a more complex example. Suppose your application changes from the active to the suspended state after the user has initiated a call but before it connects (that is, your application suspends while the call is in the dialing state). Suppose further that, while your application is suspended, the call changes first to the connected state and then to the disconnected state. When your application returns to the active state, you get a single cellular call event indicating that the call is disconnected.

See `Cellular_Call_States` (page 10) for the various call states. For an explanation of application states, see "Understanding an Application's States and Transitions" in *iOS Application Programming Guide*.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCallCenter.h`

## currentCalls

An array representing the cellular calls in progress. (read-only)

```
@property (readonly, retain) NSSet *currentCalls
```

**Discussion**
An array containing a `CTCall` object for each cellular call in progress. If no calls are in progress, the value of this property is `nil`.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCallCenter.h`

# CTCarrier Class Reference

| | |
|---|---|
| **Inherits from** | NSObject |
| **Conforms to** | NSObject (NSObject) |
| **Framework** | /System/Library/Frameworks/CoreTelephony.framework/ |
| **Availability** | Available in iOS 4.0 and later. |
| **Declared in** | |

## Overview

Use the `CTCarrier` class to obtain information about the user's home cellular service provider, such as its unique identifier and whether or not it allows VoIP (Voice over Internet Protocol) calls on its network.

## Tasks

### Obtaining Information About the Cellular Service Provider

`allowsVOIP` (page 16) *property*
Indicates if the carrier allows VoIP calls to be made on its network. (read-only)

`carrierName` (page 16) *property*
The name of the user's home cellular service provider. (read-only)

`isoCountryCode` (page 16) *property*
The ISO country code for the user's cellular service provider. (read-only)

`mobileCountryCode` (page 17) *property*
The mobile country code (MCC) for the user's cellular service provider. (read-only)

`mobileNetworkCode` (page 17) *property*
The mobile network code (MNC) for the user's cellular service provider. (read-only)

## Properties

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

## allowsVOIP

Indicates if the carrier allows VoIP calls to be made on its network. (read-only)

```
@property(nonatomic, readonly, assign) BOOL allowsVOIP
```

**Discussion**
A read-only Boolean value that is `YES` if the carrier allows VoIP calls to be made on its network, or `NO` if not.

The value for this property is `nil` if any of the following apply:

- The device is in Airplane mode.
- There is no SIM card in the device.
- The device is outside of cellular service range.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCarrier.h`

## carrierName

The name of the user's home cellular service provider. (read-only)

```
@property(nonatomic, readonly, retain) NSString *carrierName
```

**Discussion**
This string is provided by the carrier and formatted for presentation to the user. The value does not change if the user is roaming; it always represents the provider with whom the user has an account.

The value for this property is `nil` if any of the following apply:

- The device is in Airplane mode.
- There is no SIM card in the device.
- The device is outside of cellular service range.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCarrier.h`

## isoCountryCode

The ISO country code for the user's cellular service provider. (read-only)

```
@property(nonatomic, readonly, retain) NSString *isoCountryCode
```

**Discussion**
This property uses the ISO 3166-1 country code representation.

The value for this property is `nil` if any of the following apply:

- The device is in Airplane mode.
- There is no SIM card in the device.
- The device is outside of cellular service range.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCarrier.h`

## mobileCountryCode

The mobile country code (MCC) for the user's cellular service provider. (read-only)

```
@property(nonatomic, readonly, retain) NSString *mobileCountryCode
```

**Discussion**
A read-only `NSString` object that contains the numeric mobile country code for the user's cellular service provider. MCCs are defined by ITU-T Recommendation E.212, "List of Mobile Country or Geographical Area Codes." Typing this property as an `NSString` object, rather than a number type, ensures that leading zeroes in MCCs are respected.

The value for this property is `nil` if any of the following apply:

- The device is in Airplane mode.
- There is no SIM card in the device.
- The device is outside of cellular service range.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCarrier.h`

## mobileNetworkCode

The mobile network code (MNC) for the user's cellular service provider. (read-only)

```
@property(nonatomic, readonly, retain) NSString *mobileNetworkCode
```

**Discussion**
A read-only `NSString` object that represents the numeric mobile network code for the user's cellular service provider. Typing this property as an `NSString` object, rather than a number type, ensures that leading zeroes in MNCs are respected.

The value for this property is `nil` if any of the following apply:

- The device is in Airplane mode.

- There is no SIM card in the device.
- The device is outside of cellular service range.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTCarrier.h`

# CTTelephonyNetworkInfo Class Reference

| | |
|---|---|
| **Inherits from** | NSObject |
| **Conforms to** | NSObject (NSObject) |
| **Framework** | /System/Library/Frameworks/CoreTelephony.framework/ |
| **Availability** | Available in iOS 4.0 and later. |
| **Declared in** | |

## Overview

Use the `CTTelephonyNetworkInfo` class to respond to changes in the user's cellular service provider. This occurs, for example, if a user swaps the device's SIM card with one from another provider, while your application is running. This class also gives you access to the CTCarrier object, which contains information about the user's home cellular service provider.

## Tasks

### Obtaining Information About the Cellular Service Provider

`subscriberCellularProvider` (page 19) *property*
Information about the user's cellular service provider. (read-only)

`subscriberCellularProviderDidUpdateNotifier` (page 20) *property*
Dispatched when the user's cellular service provider information changes.

## Properties

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

### subscriberCellularProvider

Information about the user's cellular service provider. (read-only)

```
@property(readonly, retain) CTCarrier *subscriberCellularProvider
```

**Discussion**
A `CTCarrier` object that contains information about the user's home cellular service provider—that is, the provider with whom the user has an account. This information is available immediately after you allocate and initialize the `CTTelephonyNetworkInfo` object.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTTelephonyNetworkInfo.h`

## subscriberCellularProviderDidUpdateNotifier

Dispatched when the user's cellular service provider information changes.

```
@property(nonatomic, copy) void
    (^subscriberCellularProviderDidUpdateNotifier)(CTCarrier*);
```

**Discussion**
A block object that is dispatched on the default priority global dispatch queue when the user's cellular provider information changes. This occurs, for example, if a user swaps the device's SIM card with one from another provider, while your application is running.

To handle changes in cellular service provider information, define a block in your application and assign it to this property. The block must be implemented to support being called from any context.

**Availability**
Available in iOS 4.0 and later.

**Declared In**
`CTTelephonyNetworkInfo.h`

# Document Revision History

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

| Date | Notes |
| --- | --- |
| 2010-03-15 | New document that describes a framework for obtaining information about cellular calls and about the cellular service provider. |

Document Revision History